Evaluating web PKIs

Jiangshan Yu*, Mark Ryan

***Abstract*—Certificate authorities serve as trusted parties to help secure web communications. They are a vital component for ensuring the security of cloud infrastructures and big data repositories. Unfortunately, recent attacks using mis-issued certificates show this model is severely broken.**

Much research has been done to enhance certificate management in order to create more secure and reliable cloud architectures. However, none of it has been widely adopted yet, and it is hard to judge which one is the winner.

This paper provides a survey with critical analysis on the existing proposals for managing public key certificates. We hope our evaluation framework would be helpful for future research on designing an alternative certificate management system to secure the internet.

***Index Terms*—Certificate management, certificate verification, SSL/TLS validation, SSL/TLS Authentication.**

I. Introduction

The term big data refers to the massive amounts of digital information generated from multiple sources, such as social media sites, climate monitoring sensors, digital pictures and videos, online payment records, smart phones, and IoT devices. Security and privacy issues are magnified by velocity, volume, and variety of big data, such as large-scale cloud infrastructures, diversity of data sources and formats, streaming nature of data acquisition and high volume inter-cloud migration.

Ensuring the security of big data requires encryption, and in particular, requires *public key encryption*. The producers of big data (sensors, websites, etc.) will have public keys, and the consumers may also have public keys. The main practical difficulty in deploying public key cryptography is to ensure that the correct public key is being used. *Public key infrastructure* (PKI) [1] is the name given to the policies and procedures needed to create, manage, and distribute public-key certificates in order to use public key encryption securely.

Putting good PKI in place would provide a range of benefits for big data. As mentioned above, it would allow public key encryption. But it also enables a range of other uses and applications of public keys, such as digital signatures, secure multi-party computation, privacy-preserving data mining, and proofs of possession and proofs of knowledge [2], [3], [4], [5].

To give an intuition of potential security and privacy applications of PKI in big data, we present some example scenarios. We then explain in a high level how PKI would help enable these applications.

- **Entity authenticity**: A web user needs to verify that a social network web server it is communicating with really is the correct one. To achieve this, the web server typically sends its public key to the user's web browser. The browser needs PKI to ensure that the public key being used really is the bank's public key [6], [7], [8].
- **Data authenticity**: A server needs to ensure that data it receives from a sensor is authentic. To achieve this, the sensor can digitally sign the data. The server requires PKI in order be sure to use the correct public key to verify the sensor's signature [9], [10].
- **Data confidentiality**: Data exchanged between two parties should be secure, namely only the sender and the recipient can read the data. To achieve this, the two parties need to establish a *session key* between them. If they each have a public key and PKI is in place, then there are several protocols [11], [12], [13] they can use to establish the session key.
- **Data integrity**: Any unintended changes to the exchanged data between two parties (for example, two embedded devices in an *ad hoc* network) should be prevented, or at least, detected. There are several ways to achieve this (for example, one can use digital signatures or message authentication codes [14]), but they all rely on PKI to manage the public keys of the devices.
- **Privacy-preserving data mining**: Suppose a data owner wants to subcontract the processing of customer data (e.g. social security numbers or health records) to a third party. This data is sensitive, and should not be divulged freely. A variety of cryptographic techniques can be used [15], [16], such as *fully homomorphic encryption* (which allows the third party to compute with ciphertexts) [17], [18], [19], [20], [21] and *multi-party computation* (which allows the third party to contribute to a computation without seeing all the plaintext data) [22], [23]. All of these techniques require the various parties to have public keys, and PKI is needed to ensure their correct management.
- **Verifiability of computational results**: When subcontracting data processing in the example above, another party may wish to verify that the processing has been done correctly. Once again, there are several techniques in cryptography that can achieve this. *Verifiable computing* is the name given to this field [24], [25], [26], [27]. It starts from the assumption that the parties have public keys and there is a PKI in place to manage them.

II. An overview of PKI

The purpose of PKI is to ensure that a party has an authentic copy of the genuine public key of another party. For example, when a user logs in to Facebook through her web browser, the web session will be secured by the public key of the Facebook server. If the user's web browser accepts

* Corresponding author.
Jiangshan Yu and Mark Ryan are with the School of Computer Science, University of Birmingham, UK. E-mail: {jxy223, m.d.ryan}@cs.bham.ac.uk

an inauthentic public key for Facebook created by an attacker, then the traffic (including log-in credentials) can be intercepted and manipulated by the attacker.

The authenticity of keys is assured at present by *certificate authorities* (CAs). In the given example, the browser is presented with a public key certificate for Facebook, which is intended to be unforgeable evidence that the given public key is the correct one for the web site. A public key certificate is a digital document declaring that the recorded subject owns the public key presented in the certificate. It contains a public key, the identity of the key owner, and a signature of an entity that has verified the certificate's contents are correct. In a typical PKI scheme, the signer is a trusted party called certificate authority (CA), usually a company (e.g. VeriSign and Comodo) which charges customers to issue certificates for them. The user's browser is pre-configured to accept certificates from certain known CAs. A typical installation of Firefox has about 100 root certificates in its database. Each root CA can empower many intermediate CAs. The EFF SSL observatory has observed more than 1500 CAs [28].

Unfortunately, CA model that has been in use on the web for the last 20 years is vulnerable to attacks. The main weakness of CA model is that CAs must be assumed to be trustworthy. If a CA is dishonest or compromised, it may issue certificates asserting the authenticity of fake keys; those keys could be created by an attacker or by the CA itself. In practice, the assumption of honesty does not scale up very well. As already mentioned, a browser typically has hundreds of CAs registered in it, and the user cannot be expected to have evaluated the trustworthiness and security of all of them. This fact has been exploited by attackers [29], [30], [31], [32], [33], [34]. In 2011, two CAs were compromised: Comodo [35] and DigiNotar [36]. In both cases, certificates for high-profile sites were illegitimately obtained (e.g. Google, Yahoo, Skype, etc.). In the second case, these certificates reportedly used in a *man in the middle* (MITM) attack [37]. See [38] for a survey on CA compromises.

Another problem with CA model is the *certificate revocation management*. The CA model it self does not provide any effective way for managing certificate revocation. In common practice, Certificate Revocation Lists (CRL) [39], [40], [41], On-line Certificate Status Protocol (OCSP), and certificate revocation trees [42], [43], [44] are used to handle certificate revocation. In order to remove the need for on-the-fly revocation checking, they mostly involve periodically pushing revocation lists to browsers. However, such solutions create a window during which the browser's revocation lists are out of date until the next push.

Several protocols are proposed to strengthen the current certificate management system. However, none of them has been widely adopted yet. On one side, this is partly because the replacement of a large scale system is challenging. On the other side, this is also because new proposals each have different sets of advantages and disadvantages, and it may not be clear what features are really needed in given applications.

Clark and Van Oorschot [45] have presented an analysis on TLS mechanism and issues, by concerning themselves with crypto weakness and implementation issues of HTTPS, and trust issues of certificate management. However, they left the public log based certificate management systems out of the analysis. The use of public logs is now the dominant trend in managing web certificates. The main idea of public log based certificate management systems is to make certificate management transparent by using public audit-able logs to record all issued certificates. Clients will only accept a certificate if it is recorded in the log. Site owners can compare their own local record with the log to check whether a mis-issued certificate has been recorded in the log. This gives the site owners the ability to verify issued certificates for their sites, and make the certificate management transparent.

Kim et al. [46] have presented a comparison of web certificate management mainly based on the duration of compromise and duration of unavailability. The former shows, given the compromise of a domain's private key, how long the domain can be impersonated; and the later concerns the unavailability time period of a domain's certificate in a system.

The above two works are broad and they evaluate web certificate management systems from different perspectives. However, some important aspects are not considered in the existing work. For example, *offline verification* is one of the desired properties that have been left out from the above analyses. This property ensures that internet users can verify a received certificate without having to communicate with other parties. This is extremely useful when a user needs to connect from a captive portal in an airport or in a hotel, since the user's device cannot make other connections before they paid for the internet connection. In addition, this property also reduces the communication cost for certificate verification, as the verifier is not required to have extra connections for verifying a certificate.

Another important property not considered by the existing works is *trust agility* [47] — it allows users to freely make decisions on which certificate management service provider they wish to trust, for establishing secure communications with domain servers. In particular, we discovered a new aspect of trust agility, namely independence of trust. It requires that one or more service providers cannot not influence another service provider's service. It is in particular useful in the scenario where there exists a set of service providers, and users need to put their trust in a subset of these service providers for certificate management. If a system does not offer this feature, then it means that even if the set of service providers chosen by a user is trustworthy, a malicious service provider that is not trusted by the user can still influence the certificate verification result, and put the user in the risk of accepting fake certificates. Since the independence of trust is more strict, it is possible that a system offers the generic trust agility, but it does not offer independence of trust. In this case, users are free to make their trust decisions, but servers that are not trusted by the user are still able to affect the certificate management services delivered to the user.

One more example of desired properties that is not considered by the existing evaluation frameworks is called *anti-oligopoly*. It is proposed in [48] as a foundational property. This property observes that the present-day certificate authority model requires a fixed set of global certificate authorities

to be known to every browser, which implies an oligopoly. In fact, the current set of CAs trusted by the browser is dominated by U.S. organisations. This means that U.S. government agencies are likely to be able to control these organisations. This cannot be considered satisfactory in the presence of mutual distrust between nations regarding cybersecurity and citizen surveillance, and also trade sanctions which may prevent the USA offering services (such as CA services) to certain other countries.

To help research in securing the web certificate management, we classify 16 prominent proposals into different categories, and provide a qualitative analysis on selected proposals based on 15 criteria.

III. Desired Features and security concerns

To evaluate different systems in a systemic way, we list the desired features and security concerns for web certificate management systems.

1) Trust
   - *Trust agility* [47] allows users to freely decide which entities they want to trust for confirming public key information of domain servers, and to revise their decision at any time.

     In particular, we observed one aspect of the trust agility that has not been discovered in the literature, namely the independence of trust. It requires that the trust relations between service providers will not influence the trust relations between clients and the service providers they trust. In other words, one or more service providers cannot not influence another service provider's service to its clients.
   - *Free of trusted parties* is the property says that no party is required to be trusted for certificate issuance and revocation. For example, a certificate authority in the CA model is required to be trusted by all browsers, so the CA model does not provide *free of trusted parties*. This property is the strongest one in all trust-related features.
   - *Verifiable trusted parties* is the property says that the behaviour of trusted parties is transparent to and can be efficiently verified by users.
   - *Anti-oligopoly* [48] is the feature that prevents the monopoly or oligopoly of certificate management services. To achieve this, the trust on any service provider (e.g. CAs) should be minimised, and the system should support self-issued certificates.
2) Availability
   - *Offline verification* is a feature such that in a system, clients can verify a given key or certificate without having to connect to other parties.

     This feature is desired when a user needs to connect from a captive portal — a login page or payment page — before using the Internet. The use of captive portal is very common in public places, for example, airports or hotels. When a user is presented a captive portal, the user cannot establish a connection with any party to check the obtained public key as no internet is available. In addition, this feature also reduces the communication cost and network latency, as it does not require additional connections.
   - *Built-in key revocation* requires the system to have its own mechanism to effectively manage certificate revocation, rather than relaying on existing revocation protocols (*e.g.* certificate revocation list (CRL) or on-line certificate status protocol (OCSP)).

     The current certificate revocation management protocols (e.g. CRL and OCSP) have different limitations and cannot offer satisfactory services. So it is necessary for systems to have an integrated revocation mechanism to effectively manage certificate revocations.
   - *Scalability* is the property enabling a system to handle increasing real world workload. It is important that a system is capable to support enrolment from existing and potential future HTTPS servers.
   - *Multiple certificate support* says that the certificate verification system allows a domain to have multiple certificates. The fact that many sites have multiple certificates emphasises the importance of this feature.
   - *Timely key verification* says that the period from the time a domain owner establishes a key and the time a user can verify the key is short.

     This is a feature that has not been prominent in the literature. This feature is useful when a domain server updates its certificates. A system that does not offer this feature would cause the problem that the newly issued certificate cannot be verified and will not be accepted by web browsers within a short time period after the certificate issuance. This reduces the availability.
3) Security
   - *First connection protection* is the feature that protects the first connection between two communication parties.

     This is useful to prevent attacks on 'trust on first use'-based systems. In addition, it is likely to be the first connection when a user connects to a captive portal. So the system should protect users' first connection to a domain server.
   - *Denial of service (DoS) attack protection* is the security guarantee that prevents attacks on the key verification infrastructure in order to denial the verification services.

     This feature is useful to prevent attacks that attempt to block the verification servers to stop users verifying the received certificates.
   - *Use of mis-issued certificate prevention* measures whether the system can prevent MITM attacks launched by an attacker with mis-issued certificates. In other words, even if an attacker has obtained a mis-issued certificate, web browsers should still not accept this certificate. This gives users extra security guarantee against compromised CAs.
   - *Use of mis-issued certificate detection* measures

whether the system provides features allowing one to detect MITM attacks launched by using mis-issued certificates.

This is a weaker security guarantee, as it can only detect attacks rather than prevent attacks. However, CAs are business, and they are willing to maintain their reputation to keep their customers. So they might not launch attacks if their attacks will be detected. So this feature still offers some sensible security guarantee.

- *Provably secure* measures if the security of a given system is formally verified.

  It is well-known that security protocols are notoriously difficult to get right, and the only way to avoid this is with systematic verification.

4) Usability
   - *No user involvement* is a feature related to usability, such that the key verification result and the decision of accepting or rejecting a certificate do not need the extra involvement of users.

     This is an important feature to have, as users are not qualified to make decisions on the browser warnings, and they will likely to click through security warnings [49].

5) Privacy
   - *Protecting browsing history* says that the system does not leak users browsing history to other parties. In a PKI, if a user needs to ask another party to verify a received certificate, then the user's browsing activity is leaked to the verification party, as the subject of the to be verified certificate would very likely to be the website that the user is going to visit.

IV. Existing Proposals

Several protocols are proposed to strengthen the current certificate management system. According to the principles of each design, we classify leading certificate management systems into three categories, namely *difference observation*, *scope restriction*, and *certificate management transparency*.

A. Classic

CA-based certificate management system is the current deployed PKI. It is highly usable and scalable. Unfortunately, it requires users to fully trust all certificate authorities, and the trust cannot be modified without sacrificing users' ability to securely communicate with some domains securely. As a result, it does not provide trust agility, implies an oligopoly (on CA), and cannot easily prevent nor detect MITM attacks when a CA is compromised.

B. Difference observation

Difference observation is a concept aiming to detect untrustworthy CAs, by enabling a browser to verify if the received certificates are different from those that other people are being offered [50], [51], [52], [53], [47].

1) Perspectives: In 2008, Wendlandt, Andersen and Perrig implemented a Firefox addon, called *Perspectives* [50]. It is proposed to improve the security of trust-on-first-use authentication by asking different observers (a.k.a. notary servers) to detect inconsistent public keys of the same server. In Perspectives, observers are decentralised and independent. Each observer stores all observed keys or certificates with corresponding timestamps, and periodically checks updates and revocations. When a client wants to make a secure connection with a domain server, the client requests the server's public key from the server and from multiple observers, then compares the received keys. If the obtained public keys are consistent, the client considers the public key is trustworthy and uses this key to establish a secure connection. Otherwise, it might indicate that an attacker has launched man-in-the-middle (MITM) attack by offering a different public key to the client. So the client needs to make a decision on whether to use the obtained key or not.

- **Strength**
  Perspectives makes MITM attacks using mis-issued certificates difficult to launch without being detected, as an attacker would have to additionally intercept all connections between observers and the victim. In addition, it provides trust agility as users can choose which observer they want to use for certificate verification. Moreover, since it supports self-signed certificates, and does not require a fixed set of observers, it provides anti-oligopoly.
- **Weakness**
  With Perspectives, if a server has multiple public keys or certificates, then clients will likely get a warning of receiving inconsistent public keys. This is due to the fact that a client might receive two different genuine certificates of the same domain from the domain server and an observer. In addition, a new public key or a new server will suffer an unavailability period in the system. Since observers periodically check new public keys and revocations, the latest information about new public keys and revocations will not be immediately available from the observers. So, Perspectives does not offer timely key verification. Also, when a browser receives the latest genuine one from the server, and the revoked one from observers, then the browser will show a pop-up window warning the user that two different keys are observed, although what the server provided is a valid certificate. Such faulty warnings reduce usability of the system. Moreover, if two different certificates are detected, then the user needs to make a decision on whether to continue the connection. However, users are not qualified to make such a decision and they are likely to click through the warnings [49]. Furthermore, any observer can learn a user's browsing history when the user requests verification on a certificate. Last, it does not work when a user needs to connect from a captive portal, as no internet is available for connecting to an observer.

2) DoubleCheck: In 2009, Alicherry and Keromytis [51] proposed *DoubleCheck* to solve the issue of leaking user browsing history, and the issue that new keys might suffer an

TABLE I
EXISTING PROMINENT PROPOSALS

Category	Existing Proposals
Classic	CA-based certificate management system;
Difference observation	Perspectives ('08) [50]; DoubleCheck ('09) [51];
	Convergence ('11) [47]; Certificate Patrol ('11) [52]; CertLock ('11) [53]; TACK ('12) [54].
Scope restriction	Public key pinning ('11) [55]; DANE ('11) [56];CAge ('13) [57].
Certificate management transparency	Sovereign Keys ('12) [58]; Certificate Transparency ('12) [59];
	AKI ('13) [46]; CIRT('14) [60]; ARPKI ('14) [61]; DTKI ('14) [48]

unavailable period in Perspectives. The main idea is to query the certificate from a target server twice: once through a TLS connection, and once through *Tor* [62].

- **Strength**
  Compared to Perspectives, it additionally protects user browsing history, and new keys does not suffer an unavailability period. Moreover, it can be deployed without requiring any new infrastructure.
- **Weakness**
  The use of *Tor* adds extra time cost (up to 15 seconds [53]) for each certificate verification. In addition, a use is likely to get a warning when a server has multiple certificates. Also, when a warning is given, a user will need to make a decision on which certificate to trust, and they are likely to click through the warning. Moreover, it will not work when a user needs to connect from a captive portal.

3) Convergence: Marlinspike proposed *Convergence* [47], a Firefox addon and an improvement on *Perspectives*, in Black Hat 2011. In Convergence, to protect users' browsing history, instead of directly communicating with notary servers (i.e. observers), users randomly choose one notary server to pass the client request to other notary servers, through an onion routing like mechanism. So the intermediate notary server does not know what a requester is requesting, and the end notary server does not know who is the requester. In addition, to reduce the number of connections a user has to make, users store verified certificates in their browser cache and only query notary servers when they received a different one. Moreover, rather than querying the certificate of a domain server from a notary server, users send the certificate received from the server to notary servers. The notary server will request a certificate from the domain server if the received certificate does not match the notary's cache.

- **Strength**
  As an improvement of Perspectives, it additionally supports timely key verification, does not require user to make decisions on which certificate to trust, and it protects user privacy. Moreover, it offers offline verification if the site has been visited before.
- **Weakness**
  Similar to Perspectives, Convergence does not support multiple certificates, and does not protect users when they are connected to a captive portal.

4) Certificate Patrol: Certificate Patrol [52] is another Firefox add-on for managing web certificates. It monitors and stores all SSL certificates a browser has obtained. Since the validity period of a certificate is fairly long, it is unlikely a certificate being changed in a short time. So, when a different certificate is observed, it is possible that one of them is a misissued certificate used by attackers. With Certificate Patrol, if the newly received certificate is different from the previously stored certificate of the same domain, the browser will display to the user the difference between the two certificates, and the user needs to make a decision on whether to trust the newly received one.

- **Strength**
  It is a lightweight tool to protect user browsing history, and to offer an extra layer of security – it can help users to detect any change of the previously received certificate.
- **Weakness**
  This addon will not work if a domain has multiple certificates, and it requires users to make decisions. In addition, it does not protect user's first connection to a website nor protect user connection from a captive portal.

5) CertLock: CertLock [53] is a Firefox addon for monitoring CAs' location. In particular, it observes the country of the CA who issued the received certificate. On the detection that two CAs from different countries have issued certificates for the same site, the browser will display a warning to the user.

- **Strength**
  CertLock can help users to detect attacks in some specific scenario. For example, a site authorised certificate authority CA_1 in country A to issue certificate for its domain. A malicious government agency in country B wants to intercept the communication between users and the site. The malicious government agency can compel a certificate authority CA_2 located in country B to issue

fake certificates for the site, then uses this mis-issued certificate to launch MITM attacks. CertLock can help users to detect such attacks.

- **Weakness**
  CertLock won't be able to detect attacks using fake certificates that are issued by CAs in the same country. In addition, a false warning will be displayed if a site has switched from a CA in country A to a CA in country B. In addition, it still relies on the CA trust model, so it does not offer trust agility nor anti-oligopoly. Also, it cannot protect user's first connection and cannot protect a user who is connected to a captive portal.

6) TACK: In 2012, Marlinspike and Perrin proposed *trust assertions for certificate keys* (TACK) [54] to remove the need of trusting CAs. In TACK, a domain server generates a TACK private/public key pair, and uses the TACK private key to certify its TLS public keys. After a client observes a consistent TACK public key of a domain multiple times, it pins the public key to the domain name, and trusts this "pin" for a period, and accepts the public key if it is certified by the private key corresponding to the observed TACK public key. If a certificate becomes compromised and the observed information has not been pinned, then the client must delete the observed TACK information and re-start the observation process. To be scalable, TACK will need an online pin store, where users can share their observed pins. However, the problem of how to design a secure pin store for users to share their observations, while prevent attackers to spoof or poison the store, remains unsolved.

- **Strength**
  TACK removes the need of CA, offers trust agility, does not require users to any trusted party[1], and provides anti-oligopoly. Once local observations are built, TACK allows offline verification, supports multiple certificates.
- **Weakness**
  Since TACK relies on visit patterns by clients to pin the domain's public key, the first several connections to a domain server will not be protected, and every new TACK key pair or new domain suffers an initial unavailability period. In addition, the revoked key will still be accepted by the client if the client still trusts its previous observation.

  To be scalable, TACK requires an on-line store to share TACK keys observed by different clients. The use of such on-line stores make TACK difficult to provide the independence of trust required by trust agility. Because a client Alice, might choose to trust some stores or clients for the TACK keys they observed. However, the store or clients trusted by Alice might put their trust on other stores and clients. This transitive trust relation could effect Alice's trust option and Alice's observation on the TACK keys. Currently, it is hard to judge whether TACK offers the independence of trust required by trust agility, as the online store is not designed yet.

[1]Here, we only consider the TACK without having an online pin store

C. Scope restriction

Scope restriction is the concept aiming to reduce the power of CAs by restricting the domain scope that a CA can vouch for.

1) Public key pinning (PKP): Public key pinning (a.k.a. certificate pinning) is a mechanism for domain servers to specify which CAs are authorised to certify public keys for a given domain. Langley et al. implemented it in Google Chrome [63].

Scalability is a main challenge for key pinning, due to the need of pre-knowledge of the mapping between each domain server and CAs. Public key pinning extension for HTTP [64] addresses the scalability challenge by allowing a domain server to declare the authorised CAs for its sites in an HTTP header.

- **Strength**
  As PKP is a way to restrict CAs' power by specifying which CAs are authorised for a given website, it protects user communications against attackers who have mis-issued certificates from CAs that are not authorised for the victim. In addition, PKP allows a website to have multiple certificates, does support offline verification, and is scalable with the PKP extension for HTTP.
- **Weakness**
  The weakness of PKP is that it cannot completely protect all user connections. For example, it cannot protect when a user does not have a pin of a website, which is generally the case for the first connection. Also, it cannot protect the connection when the pin is expired in the user browser. Moreover, it cannot effectively detect attacks when a CA has mis-issued certificates for the domains that the CA is pinned for. Furthermore, it does not offer trust agility nor anti-oligopoly.

2) DANE: Domain name system (DNS)-based authentication of named entities (DANE) [65], [66] binds the public key information to a domain name by using DNS Security Extensions (DNSSEC). More specifically, DANE enables a domain server to certify its public keys by storing the public keys in its DNS records. This DNS record is valid only if it is correctly signed as specified in DNSSEC [67]. So, the parent domain servers are the authority of their child domains. In other words, only the parent domain can certify public keys of its child domains. In this way, DANE limits the damage of dishonest or compromised authorities.

- **Strength**
  Compared to PKP, DANE is highly scalable since it is based on DNSSEC. In addition, it can protect a user even when the user connects from a captive portal.
- **Weakness**
  The security of DANE strongly relies on the trustworthiness of parent domains according to the DNS hierarchy. As a result, ICANN, top-level domains (TLDs), and second-level domains (SLDs) become to be very powerful and fully trusted CAs. So, DANE does not provide trust agility and anti-oligopoly. In addition, domain servers cannot choose which CA they want to get service from, as they have to get their keys to be certified by their parent domain.

3) CAge: In 2013, Kasten, Wustrow and Halderman proposed *CAge* [57] to restrict the scope of domains that a CA can certify public keys for. According to the data observed in [68], they show that only a small number of CAs have signed certificates for TLDs. Based on this observation, *CAge* suggests to limit a CA's certification scope by only allowing a CA to issue certificates on a restricted set of TLDs. CAge limits the scale of MITM attacks using mis-issued certificates, but cannot completely solve this problem.

- **Strength**
  As all systems in the category of scope restriction, CAge reduces the damage from a compromised CA by limiting the set of domains that a CA can vouch for.
- **Weakness**
  Since CAge is still based on the CA trust model although with restrictions on a CA's ability, it does not offer trust agility and anti-oligopoly. In addition, domain servers have less flexibility to choose which CA they want to use, because only a subset of CAs will be eligible for certifying keys for given domains.

D. Certificate management transparency

Certificate management transparency is the concept aiming to make CA's behaviour transparent. The basic idea is to use a publicly visible log to record issued certificates. So interested parties can check the log to detect any mis-issued certificates.

1) Sovereign Keys: Sovereign Keys (SK) [69] aims to get rid of browser certificate warnings, by allowing domain owners to establish a long term ("sovereign") key and by providing a mechanism by which a browser can hard-fail if it doesn't succeed in establishing security via that key. A sovereign key is a long-term key used to cross-sign operational TLS keys, and it is stored in an append-only log on a "timeline server", which is abundantly mirrored.

When a browser connects to a website, it sends a query to a mirror of the "timeline server" to check if the site has a sovereign key. If the site does have a sovereign key, then the browser only accepts a certificate for this site if the certificate is issued by CAs and is cross-signed by the sovereign key. If the certificate is not cross-signed, then rather than emit certificate warnings, the browser will try to find a way to make a sovereign key connection to the site. There are several ways to establish a connection without having a cross-signed certificate. The strongest way is to compute a hash of the sovereign key, and use that as the .onion address of the Tor hidden service which allows the secure connection. Weaker ways include stapling to the sovereign key and trying to connect through other means such as proxy and VPN, until the browser gets a verified connection.

- **Strength**
  SK introduces the first public log based PKI. It eliminates browser certificate warnings, reduces the trust put on CAs, allow a site to have multiple certificates, and prevents attacks from an attacker who compromised CAs.
- **Weakness**
  Sovereign Keys doesn't have an efficient way for the timeline server and mirrors to prove their correct behaviour. The only way for verifying it is to download an verify the entire log. So internet users and domain owners have to trust mirrors of time-line servers. Additionally, it doesn't provide any mechanism for key revocation, either of TLS keys or sovereign keys. If a domain owner loses the sovereign private key, they lose the ability to switch to new TLS keys, and may even lose control of their domain, until the sovereign key expires. Another security concern is that if a site does not have a sovereign key yet, then a determined attacker could register his own sovereign key for the site and intercept secure connections made to the site.

2) Certificate transparency: Certificate transparency (CT) [59] is proposed by Google aiming to allow domain owners to efficiently detect mis-issued certificates, by making certificate issuance transparent.

The basic idea is to use public audit-able logs to record all issued certificates. In this way, interested parties can monitor the log to verify all of CAs' behaviour. To enforce CAs to publish all issued certificates into the log, web browsers only accept certificates if a verifiable evidence is provided to prove that the certificate is present in the log.

In more detail, domain owners request from the log maintainer signed confirmations saying that their certificates are included in the log, and then they can provide this confirmation together with the corresponding certificate to web browsers. Browsers only accept a certificate if both the certificate and the signed confirmation are valid. Browsers also need to periodically verify received signed confirmation against the public log to check if the certificate is indeed being inserted in the log.

To reduce the trust put on CAs and log maintainers, CT uses an append-only log which is organised as an append-only Merkle tree. In the tree, data items (i.e. certificates or references to certificates) are stored left-to-right in chronological order at the leaves, and added by extending the tree to the right. This structure enables the log maintainer to provide two types of verifiable cryptographic proofs: (a) a proof that the log contains a given certificate, and (b) a proof that a snapshot of the log is an extension of another snapshot (*i.e.*, only appends have taken place between the two snapshot). The time and size for proof generation and verification are logarithmic in the number of certificates recorded in the log. To ensure the log maintainer is behaving correctly, CT requires monitors to check the consistency of logs.

- **Strength**
  Since CA's behaviour is transparent, CT does not require users to blindly trust CAs, i.e. the behaviour of CAs are verifiable. This makes CT to offer trust agility. In addition, CT enables domain owners to readily detect any mis-issued certificates.
- **Weakness**
  A main weakness of CT is that users still have to trust "monitors" for verifying the behaviour of logs. In addition, CT does not provide an efficient scheme for key revocation. Also, CT does not provide anti-oligopoly,

because although the set of log servers are not fixed, it doesn't have any method to allocate different domains to different logs. In CT, when a domain owner wants to check whether mis-issued certificates are recorded in logs, he needs to contact all existing logs, and download all certificates in each of the logs, because there is no way to prove to the domain owner that no certificates for his domain is in the log, or to prove that the log maintainer has showed all certificates in the log for his domain to him. Thus, to be able to detect fake certificates, CT has to keep a very small number of log maintainers. This prevents new log providers being flexibly created, creating an oligopoly. Another limitation is that CT can only detect mis-issued certificates, rather than prevent attacks that use mis-issued certificates.

3) Accountable key infrastructure: Accountable key infrastructure (AKI) [46] also uses public logs to make certificate management more transparent.

Similar to SK, AKI allows domain owners to define their own security policy by specifying several additional attributes of a certificate, such as which CA and log maintainer a domain owner wants to get services from, what is the minimum number of CA signatures to validate a certificate for her domain, etc. To obtain a certificate, a domain owner contacts at least a minimum number of CAs that she wishes to trust based on the policy, and to cross sign her public key with her security policy. Then she requests log maintainers to update her certificate, and expects a signed proof that the certificate is recorded in the log. Clients only accept a certificate if the certificate satisfies defined security policy, and is currently recorded in the log.

To be able to manage key revocations, AKI stores only the current valid certificates of domains in a public log. The log is organised as a hash tree, where certificates stored in leaves ordered lexicographically.

To detect mis-behaviours, AKI uses the "checks-and-balances" idea that allows parties to monitor each other's behaviour. So AKI limits the requirement to trust any party. Moreover, AKI prevents attacks that use fake certificates rather than merely detecting such attacks (as in CT).

- **Strength**
  AKI extends the previous architectures in several ways. First, it allows multiple CAs to sign a single certificate. Additionally, the domain can specify in its certificate which CAs and logs are allowed to attest to the certificate's authenticity. These features provide resilience against a certificate signed by a compromised or unauthorised CA. AKI can also handle key loss or compromise through cool-off periods. For example, if a domain loses its private key and registers a new certificate not signed by its old private key, the new certificate will be subject to a cool-off period (e.g., three days) during which the certificate is publicly visible but not usable. This ensures that even if an adversary obtains and registers a fake certificate, the domain has the opportunity to contact the CAs and logs to resolve the issue.
- **Weakness**
  To ensure that any log server can provide a proof for a domain's certificate, AKI logs maintain a globally consistent view of the entries that they have for a given domain name. This applies for every certificate operation (registration, update, and revocation), meaning that even frequent certificate updates (such as in the case of short-lived certificates) are subject to successful log synchronisation. In addition, AKI requires that each domain name only has one active and valid certificate associate with it at any given time. Moreover, AKI needs to rely on third parties called validators to ensure that the log is maintained without improper modifications, and to assume that CAs, public log maintainers, and validators do not collude together.

4) Certificate issuance and revocation transparency: Certificate issuance and revocation transparency (CIRT) [60] improves certificate transparency by providing transparent key revocation, and reducing reliance on trusted parties.

To provide an effective way for certificate revocation, CIRT proposes a new log structure that consists of two tree structures presenting the same set of data. The first tree is called a ChronTree, which is an append-only Merkle tree (as in CT) ordered chronologically. The second tree is called LexTree, which is a Merkle tree ordered lexicographically by the subject of the certificate. The ChronTree stores in the leaves a pair (C, h), where C is a certificate appended in the ChronTree, and h is the hash root value of the LexTree in which the last inserted data is C. The LexTree stores $h(d_i)$ in every node for some i, where d is an ordered list of certificates that has the same subject. The last element in the list is the current valid certificate of the subject.

This log structure enables the log maintainer to provide efficient proofs that (A) some data is present in the log, (B) any data having a given attribute (e.g. an identity) is absent from the log, (C) some data is the latest valid one in the log, and (D) the current log is extended from a previous version.

Loosely speaking, by proving a proof that a certificate C is the last element in an ordered list d, and $h(d)$ is present in the LexTree of the log, a verifier is ensured that C is the currently valid certificate, i.e. not revoked. Due to the use of two different trees presenting the same set of data, it is crucial to ensure that the data presented by the two trees are consistent. To verify the consistency of the two trees, CIRT distributes the monitoring role among user browsers. To do so, each user browser verifies if a randomly selected certificate stored in the ChronTree is also in the LexTree. If the number of such random verification is big enough, then the consistency between the two trees is likely to be verified.

- **Strength**
  CIRT provides a solution for managing both certificate issuance and revocation by using a new log structure, and reduces reliance on trusted parties by using user side random verifications. It also allows a domain to have multiple certificates, and to update keys timely. In addition, similar to all other systems in certificate management transparency category, it does not need users to be involved.

- **Weakness**
  A weakness of CIRT is that it can only detect attacks that use fake certificates; it cannot prevent them. Also, since CIRT was proposed for email applications, it does not support the multiplicity of log maintainers that would be required for web certificates.

5) Attack Resilient Public-Key Infrastructure: Attack Resilient Public-Key Infrastructure (ARPKI) [61] is an improvement on AKI. In ARPKI, a client can designate n service providers (e.g. CAs and log maintainers), and only needs to contact one CA to register her certificate. Each of the designated service providers will monitor the behaviour of other designated service providers. As a result, ARPKI prevents attacks even when $n-1$ service providers are colluding together, whereas in AKI, an adversary who successfully compromises two out of three designated service providers can successfully launch attacks [61].

- **Strength**
  ARPKI is the first formally verified log-based PKI system. Its security properties are proved by using a protocol verification tool called Tamarin prover [70]. The verification uses several abstractions during modelling. For example, they represent its underlying log structure (a Merkle tree) as a list.
- **Weakness**
  The weakness of ARPKI is that all n designated service providers have to be involved in all the processes (i.e. certificate registration, confirmation, and update), which would cause considerable extra latencies and the delay of client connections.

6) Distributed Transparent Key Infrastructure: Distributed Transparent Key Infrastructure (DTKI) [48] is an improvement on CIRT [60].

In DTKI, each domain owner has two types of keys — TLS keys and a master key. Browsers only accept a certificate if it is issued by the corresponding master key. Both master keys and TLS keys are recorded in public auditable logs. So dishonest CAs and log maintainers cannot issue fake certificates for a domain unless they have the corresponding master key.

To support multiple logs, DTKI has two different types of log maintainers, namely certificate log maintainers (CLMs) and the mapping log maintainer (MLM). Each CLM maintains a database of all certificates for a particular set of domains for which it is responsible. DTKI does not fix the set of certificate logs, but the MLM is unique. The MLM maintains association between certificate logs and the domains they are responsible for. So an internet user can query the MLM for the association, and query the corresponding CLM for the certificate of the target domain. All log maintainers are not required to be trusted, as they behave transparently, and can provide the same set of efficient proofs as in CIRT. Similar to CIRT, the consistency between logs are verified by user side random verification, and the behaviours of all trusted parties are efficiently verifiable.

The complex data structure in systems like CIRT and DTKI prevents them to use the verification method proposed for ARKPI, as the method requires abstractions on the underlying log data structure. To bridge this gap, in DTKI, the underlying log structure of DTKI is formalised, and the properties of log structures are formally proved. Then, the security property of the protocol is proved by using Tarmarin prover.

- **Strength**
  It is the first system which allows all parties to collude together, while is still able to prevent MITM attacks using mis-issued certificates rather than merely detect them. It formalises its underlying log data structure, and formally proved its security properties also by using Tamarin prover. DTKI requires a single global mapping log server, which might imply a monopoly. However, the mapping log is a lightweight governing party, is not required to be trusted, is not involved in every day communications, and it does not directly manage certificates for sites. So, DTKI provides anti-oligopoly at some level.
- **Weakness**
  The weakness of DTKI is that it adds extra latency compared to other systems, its complexity make it difficult to be deployed, and it suffers the same problem of SK, namely a domain owner who loses the sovereign key also loses the ability to switch to new operational keys until the SK expires.

V. Observations

Based on the above analysis, we observed the advantage and weakness in each category. This section discusses the observations based on different perspectives, i.e. on the property perspective and on the system perspective. In addition, this section summarises our observation regarding to the leading proposals in Table II.

A. Property Perspective

We summarise our observations on different system categories according to the perspective of identified properties.

Trust agility
The current CA model does not provide this feature, since any compromised CA can issue valid certificates for any domain server. Similarly, systems in the category of *Scope restriction* also do not provide this feature, because they merely restrict the set of domains that CAs can issue certificates for. Most systems in *difference observation* offer this feature, as any one can be a notary server, and users can select which notary servers they want to trust, and any notary server will not be influenced by other notary servers.

Anti-oligopoly
Systems in the category of *difference observation* normally provide this feature, as the number of observers are not fixed. In addition, the certificate verification result relies on the out-of-band checking through a different path. So, as far as the observed certificates are the same one, the client will accept it. Thus, the role of CA is minimised.

TABLE II
EVALUATION OF PROPOSALS

Desired Features	Classic	Difference observation			Scope restriction		Certificate management transparency		
	CA-based	Perspectives	Convergence	TACK	PKP	DANE	CT	ARPKI	DTKI
Trust									
Trust agility	×	√	√	√[2]	×	×	√	√	√
Free of trusted parties	×	×	×	√[2]	×	×	×	×	×
Verifiable trusted parties	×	×	×	×	×	×	√	√	√
Anti-oligopoly	×	√	√	√	×	×	×	×	√
Availability									
Offline verification	√	×	⊗[1]	√	√	√	√	√	×
Built-in key revocation	×	×	×	×	×	×	×	√	√
Scalability	√	√	√	×[2]	√	√	√	√	√
Multiple certificate support	√	×	×	√	√	√	√	×	√
Timely key verification	√	×	√	×	√	√	√	√	√
Security									
First connection protection	√	√	√	×	√	√	√	√	√
DOS attack protection	√	×	⊗[3]	√[2]	√	√	√	√	√
Use of mis-issued certificate prevention	×	√	√	√	⊗[4]	⊗[4]	×	√	√
Use of mis-issued certificate detection	×	√	√	√	⊗[4]	⊗[4]	√	√	√
Provably secure	×	×	×	×	×	×	×	√	√
Usability									
No user involvement	√	×	√	×	√	√	√	√	√
Privacy									
Protecting browsing history	√	×	√	√[2]	√	√	√	√	√

√ – The subject offers this feature.
⊗ – The subject offers this feature but with other concerns.
× – The subject does not offer this feature.

[1] This feature is satisfied if and only if the received public key/certificate can be found in the local cache.
[2] We consider the case without using an on-line crowed-sourced pin store. If an online pin store is used, then the result might be different depending on how the store is designed. (The pin store has not been proposed yet.)
[3] This feature is satisfied if and only if the received public key certificate can be found in the local cache, and the subject of the certificate has not updated its certificate.
[4] This feature is satisfied if the malicious CA is not authorised for the victim domain.

Offline verification
In the current CA model, clients only need to verify the validity of the received certificate. So, it satisfies offline verification. Systems in the category of *scope restriction* also provide this feature, as the way they work is similar to the current CA model, but with some restrictions.

Most systems in the category of *certificate management transparency* offer this feature as well, because in these systems the proofs to be verified about a certificate are provided together with the certificate. In contrast, most systems in the difference observation category don't offer this feature, because with these systems, clients have to make additional connections to verify the certificates they obtained.

Built-in key revocation
Most systems in the category of *difference observation* and *scope restriction* do not provide this feature. Most systems in the category of *certificate management transparency* do offer this feature. For example, CIRT proposed a way to manage certificate revocation by using an advanced log structure; and AKI and ARPKI manage certificate revocation by only recording the latest certificates of domains in their logs.

Multiple certificate support
The current CA model offers this feature. Systems in the category of *difference observation* generally don't provide this feature. Because when clients see different certificates of the same website from different paths or observers, a warning will be displayed to clients even if the received certificates are all genuine. Systems in the category of *scope restriction* and *certificate management transparency* provide this feature.

Timely key verification
Systems in *difference observation* are likely to not provide this feature, as the observers might not be always up to date with all domains.

First connection protection
Systems such as Certificate Patrol, Certlock, and TACK in *difference observation* do not provide this feature, because they verify the certificate based on what has been observed in the previous connections.

Denial of service (DoS) attack protection
The CA model offers this feature. All systems in the category of *scope restriction* and most systems in the category of *certificate management transparency* provide this feature as well. However, some systems in *difference observation* require out-of-band observation, so they will not provide this feature, as the verification server can be blocked.

Use of mis-issued certificate prevention
All systems in *difference observation*, and some systems in the category of *certificate management transparency*, provide this feature. In contrast, systems in the category of *scope restriction* do not provide this feature if the mis-issued certificate is issued by a CA who is authorised for the victim domain. For example, DANE cannot prevent MITM attacks when the fake certificate used by an attacker is issued by the parent domain of the victim domain.

Use of mis-issued certificate detection
All systems in *difference observation* and in *certificate management transparency* provide this feature.

B. System Perspective

As shown in the table, systems in the category of *difference observation* provide better trust-related features. However, they can have difficulties to provide a better availability, because the observer might not have the latest update, the systems in general do not provide an effective key revocation management, and they require user involvement to make decisions. Moreover, they can suffer from DoS attacks on the observers.

Systems in the category of *scope restriction* provide better usability and availability. However, they have only restricted the power of each trusted parties, but internet users still need to trust them. This can limit the damage from attacks launched by malicious CAs, but cannot completely solve the problem.

Systems in the category of *certificate management transparency* provide better security and availability. However, anti-monopoly might be a problem for these systems. DTKI shows the possibility of providing anti-monopoly, but it still needs lightweight governing party. It is desired to provide a fully distributed system and still be able to remove the need of trusted parties.

All proposed systems have different advantages compared to the current PKI, and they are trying to solve different problems. However, none of them is satisfactory of being a replacement of the current PKI, as the authors were not concerned with all desired features while designing web PKI alternatives.

VI. Conclusion

The production, processing and consumption of big data requires that all the agents involved in those operations can authenticate each other reliably. Unfortunately, authentication of servers and services on the Internet is a surprisingly hard problem. The classical CA model that we have been relying upon for 20 years is no longer adequate.

We reviewed solutions proposed, and divided them into three categories. Of the three categories, the last one, based on transparent public logs, has had the most success in the real world. Unfortunately, the version of this idea being implemented by Google doesn't handle revocation, and doesn't properly allow a multiplicity of log maintainers. We have reviewed solutions that do allow these additional features, and we think that those solutions are the most comprehensive so far.

We hope our evaluation framework will help the ongoing research on web certificate management alternatives.

References

[1] P. Yee, "Updates to the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 6818 (Proposed Standard), Internet Engineering Task Force, Jan. 2013.

[2] W. Mao, *Modern cryptography: theory and practice*, ser. HP Professional Series. Prentice Hall PTR, 2004. [Online]. Available: http://books.google.com/books?id=H42WQgAACAAJ

[3] N. Ferguson and B. Schneier, *Practical cryptography*. Wiley, 2003.

[4] B. Schneier, *Applied cryptography - protocols, algorithms, and source code in C (2. ed.)*. Wiley, 1996.

[5] N. P. Smart, *Cryptography Made Simple*, ser. Information Security and Cryptography. Springer, 2016. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-21936-3

[6] E. Rescorla, "HTTP over TLS," RFC 2818 (Informational), Internet Engineering Task Force, May 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2818.txt

[7] T. Dierks and E. Rescorla, "The transport layer security (TLS) protocol version 1.1," RFC 4346 (Proposed Standard), Internet Engineering Task Force, Apr. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4346.txt

[8] S. Turner and T. Polk, "Prohibiting secure sockets layer (SSL) version 2.0," RFC 6176 (Proposed Standard), Internet Engineering Task Force, Mar. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6176.txt

[9] J. L. Massey, "Cryptographya selective survey," *Digital Communications*, vol. 85, pp. 3–25, 1986.

[10] A. Aziz and W. Diffie, "Privacy and authentication for wireless local area networks," *IEEE Personal Commun.*, vol. 1, no. 1, pp. 25–31, 1994.

[11] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, November 1976.

[12] W. Diffie, P. C. van Oorschot, and M. J. Wiener, "Authentication and authenticated key exchanges," *Des. Codes Cryptography*, vol. 2, no. 2, pp. 107–125, 1992. [Online]. Available: http://dx.doi.org/10.1007/BF00124891

[13] H. H. Kilinc and T. Yanik, "A survey of SIP authentication and key agreement schemes," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 2, pp. 1005–1023, 2014.

[14] S. Turner and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms," RFC 6151 (Informational), Internet Engineering Task Force, Mar. 2011.

[15] C. Dwork and K. Nissim, "Privacy-preserving datamining on vertically partitioned databases," in *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, 2004, pp. 528–544.

[16] B. Pinkas, "Cryptographic techniques for privacy-preserving data mining," *SIGKDD Explorations*, vol. 4, no. 2, pp. 12–19, 2002.

[17] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, 2010, pp. 24–43.

[18] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, 2009.

[19] C. Fontaine and F. Galand, "A survey of homomorphic encryption for nonspecialists," *EURASIP Journal on Information Security*, vol. 2007, no. 1, pp. 1–10, 2007.

[20] V. Vaikuntanathan, "Computing blindfolded: New developments in fully homomorphic encryption," in *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, 2011, pp. 5–16.

[21] C. Fontaine and F. Galand, "A survey of homomorphic encryption for nonspecialists," *EURASIP J. Information Security*, vol. 2007, 2007.

[22] W. Du and M. J. Atallah, "Secure multi-party computation problems and their applications: a review and open problems," in *Proceedings of the New Security Paradigms Workshop 2001, Cloudcroft, New Mexico, USA, September 10-13, 2001*, 2001, pp. 13–22.

[23] S. Goldwasser, "Multi-party computations: Past and present," in *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing, Santa Barbara, California, USA, August 21-24, 1997*, 1997, pp. 1–6.

[24] K. Yang, X. Jia, and K. Ren, "Secure and verifiable policy update outsourcing for big data access control in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 12, pp. 3461–3470, 2015.

[25] Q. Zheng and S. Xu, "Verifiable delegated set intersection operations on outsourced encrypted data," in *2015 IEEE International Conference on Cloud Engineering, IC2E 2015, Tempe, AZ, USA, March 9-13, 2015*, 2015, pp. 175–184.

[26] Q. Zheng, S. Xu, and G. Ateniese, "VABKS: verifiable attribute-based keyword search over outsourced encrypted data," in *2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014*, 2014, pp. 522–530.

[27] J. Wang, X. Chen, X. Huang, I. You, and Y. Xiang, "Verifiable auditing for outsourced database in cloud computing," *IEEE Trans. Computers*, vol. 64, no. 11, pp. 3293–3303, 2015.

[28] "The EFF SSL Observatory." [Online]. Available: https://www.eff.org/observatory

[29] P. Eckersley, "Iranian hackers obtain fraudulent HTTPS certificates: How close to a web security meltdown did we get?" Electronic Frontier Foundation, 2011. [Online]. Available: https://www.eff.org/deeplinks/2011/03/iranian-hackers-obtain-fraudulent-https

[30] J. Leyden, "Trustwave admits crafting SSL snooping certificate: Allowing bosses to spy on staff was wrong, says security biz," The Register, 2012. [Online]. Available: www.theregister.co.uk/2012/02/09/tustwave_disavows_mitm_digital_cert

[31] "MS01-017: Erroneous Verisign-issued digital certificates pose spoofing hazard," Microsoft Support. [Online]. Available: http://support.microsoft.com/kb/293818

[32] P. Roberts, "Phony SSL certificates issued for Google, Yahoo, Skype, others," Threat Post, March 2011. [Online]. Available: http://threatpost.com/phony-ssl-certificates-issued-google-yahoo-skype-others-032311

[33] T. Sterling, "Second firm warns of concern after dutch hack," Yahoo! News, September 2011. [Online]. Available: http://news.yahoo.com/second-firm-warns-concern-dutch-hack-215940770.html

[34] L. O. M. N. Falliere and E. Chien, "W32.stuxnet dossier. Technical report, symantec corporation," 2011.

[35] J. Appelbaum, "Detecting certificate authority compromises and web browser collusion," Tor Blog, 2011.

[36] "Black tulip report of the investigation into the DigiNotar certificate authority breach," Fox-IT (Tech. Report), 2012.

[37] C. Arthur, "Rogue web certificate could have been used to attack Iran dissidents," The Guardian, 2011.

[38] A. Niemann and J. Brendel, "A survey on CA compromises," 2013.

[39] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 5280 (Proposed Standard), Internet Engineering Task Force, May 2008, updated by RFC 6818. [Online]. Available: http://www.ietf.org/rfc/rfc5280.txt

[40] R. L. Rivest, "Can we eliminate certificate revocation lists?" in *Financial Cryptography*. Springer, 1998, pp. 178–183.

[41] A. Langley, "Revocation checking and Chrome's CRL," *ImperialViolet* (blog), 2012.

[42] P. Kocher, "A quick introduction to certificate revocation trees," 1998, unpublished work.

[43] K. Nissim and M. Naor, "Certificate revocation and certificate update." in *USENIX Security*. Citeseer, 1998.

[44] B. Laurie and E. Kasper, "Revocation transparency," September 2012, google Research.

[45] J. Clark and P. C. van Oorschot, "SoK: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements," in *IEEE Symposium on Security and Privacy*, 2013, pp. 511–525.

[46] T. H.-J. Kim, L.-S. Huang, A. Perrig, C. Jackson, and V. Gligor, "Accountable key infrastructure (AKI): A proposal for a public-key validation infrastructure," in *the 22nd International World Wide Web Conference (WWW 2013)*, 2013.

[47] M. Marlinspike, "SSL and the future of authenticity," in *Black Hat, USA*, 2011.

[48] J. Yu, V. Cheval, and M. Ryan, "DTKI: a new formalized PKI with no trusted parties," *CoRR*, vol. abs/1408.1023, 2014. [Online]. Available: http://arxiv.org/abs/1408.1023

[49] D. Akhawe and A. P. Felt, "Alice in warningland: A large-scale field study of browser security warning effectiveness." in *Usenix Security*, 2013, pp. 257–272.

[50] D. Wendlandt, D. G. Andersen, and A. Perrig, "*Perspectives:* improving SSH-style host authentication with multi-path probing," in *USENIX Annual Technical Conference*, 2008, pp. 321–334.

[51] M. Alicherry and A. D. Keromytis, "Doublecheck: Multi-path verification against man-in-the-middle attacks," in *ISCC*, 2009, pp. 557–563.

[52] "Certificate patrol." [Online]. Available: http://patrol.psyced.org

[53] C. Soghoian and S. Stamm, "Certified lies: Detecting and defeating government interception attacks against SSL," in *Financial Cryptography*, 2011, pp. 250–259.

[54] M. Marlinspike and T. Perrin, "Internet-draft: Trust assertions for certificate keys (TACK)," 2012.

[55] "Public key pinning," http://www.imperialviolet.org/2011/05/04/pinning.html, May, 2011.

[56] R. L. Barnes, "Dane: Taking tls authentication to the next level using dnssec," *IETF journal, October*, 2011.

[57] J. Kasten, E. Wustrow, and J. A. Halderman, "CAge: Taming certicate authorities by inferring restricted scopes," in *Financial Cryptography*, 2013.

[58] P. Eckersley, "Internet-draft: Sovereign key cryptography for internet domains," 2012.

[59] B. Laurie, A. Langley, and E. Kasper, "Certificate Transparency," RFC 6962 (Experimental), Internet Engineering Task Force, 2013.

[60] M. Ryan, "Enhanced certificate transparency and end-to-end encrypted mail," in *NDSS*, 2014.

[61] T. H. Kim, P. Gupta, J. Han, E. Owusu, J. I. Hong, A. Perrig, and D. Gao, "ARPKI: attack resilient public-key infrastructure," in *ACM CCS*, 2014.

[62] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *In Proceedings of the 13th USENIX Security Symposium*, 2004, pp. 303–320.

[63] A. Langley, "Public-key pinning," *ImperialViolet* (blog), 2011.

[64] C. Evansm, C. Palmer, and R. Sleevi, "Internet-draft: Public key pinning extension for http," October 2014, draft 21.

[65] R. Barnes, "Use Cases and Requirements for DNS-Based Authentication of Named Entities (DANE)," RFC 6394 (Informational), Internet Engineering Task Force, Oct. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6394.txt

[66] P. Hoffman and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA," RFC 6698 (Proposed Standard), Internet Engineering Task Force, Aug. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6698.txt

[67] S. Weiler and D. Blacka, "Clarifications and Implementation Notes for DNS Security (DNSSEC)," RFC 6840 (Proposed Standard), Internet Engineering Task Force, Feb. 2013. [Online]. Available: http://www.ietf.org/rfc/rfc6840.txt

[68] N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman, "Mining your Ps and Qs: detection of widespread weak keys in network devices," in *Proceedings of the 21st USENIX conference on Security symposium*, ser. Security'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 35–35. [Online]. Available: http://dl.acm.org/citation.cfm?id=2362793.2362828

[69] P. Eckersley, "Sovereign key cryptography for internet domains," Internet Draft, 2012.

[70] S. Meier, B. Schmidt, C. Cremers, and D. A. Basin, "The TAMARIN prover for the symbolic analysis of security protocols," in *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, 2013, pp. 696–701.