RSMG3 Report

Improving Public Key Certification Systems

Jiangshan Yu

Supervisor: Prof. Mark Ryan

25^{th} August.

School of Computer Science
University of Birmingham
Edgbaston, Birmingham, West Midlands, B15 2TT, UK.

Thesis Group Committee:

Dr. Eike Ritter
Dr. Dan Ghica
Dr. Vincent Cheval

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Abstract

This proposal motivates to enhance the security of current public key certification systems and their associated applications. In particular, we first review and analyse the literature. To address the identified security weakness, we present our on-going work and explain our future work. My recent and upcoming academic activities are also included in this proposal.

Contents

Chapter 1

Overview

Computer controlled communication networks provide an instant and effortless communication between people and devices even on opposite sides of the world. For example, cloud storage services (*e.g.* Dropbox, Microsoft SkyDrive, and Apple iCloud) simplify data sharing between different devices; online social networking services (*e.g.* Facebook, Twitter, and Google+) offer an attractive internet-based platform for client interconnections; and email services (*e.g.* Gmail, Hotmail, and Yahoo! Mail) provide extremely fast and 24/7 mail services with the advantage that clients can manage their mails anytime and anywhere in the world. However, two high-profile questions are arisen – "are your private data secure?" "can you trust your service providers with your private information?" [13, 33, 66]. The answer can be found from recent discoveries:

- in March 2011, *Comodo* was attacked and fake certificates[1] were issued for popular domains (*e.g.* Google, Yahoo, Skype, etc.) [58];
- in August 2011, Iranian government attacked Google users by offering fake certificates [62];
- in August 2011, *DigiNotar* issued 531 fake certificates for more than three hundred domains, including most of major Internet communications companies [71];
- in June 2013, the U.S. National Security Agency (NSA) contractor Edward Snowden leaked the existence of PRISM – a surveillance program from NSA since 2007 [8]. In PRISM, NSA colluded with global giant companies (such as Google, Facebook, Microsoft, Apple, and Yahoo!) to launch worldwide attacks to surveil American citizens and to spy on other countries [7, 12, 36].

Obviously, current systems have not properly protected client data (*i.e.* existing protocols are insufficient to guarantee confidentiality of client data); and service providers (*e.g.* DigiNotar) cannot be fully trusted (*i.e.* they have not properly protected client privacy and they have actively launched attacks). Several protocols (*e.g.* [32, 44, 67]) were developed in the literature to improve data confidentiality and client privacy. However,our analytical survey (in next chapter) shows that they still have security weaknesses. In particular, our research identified the following **problems**:

[1]Fake certificates can be used by attackers to launch man-in-the-middle (MITM) attacks – by doing this, attackers can silently intercept victims' exchanged messages, read them in clear text, and re-transfer them to the receiver.

- public key certification protocols – which are used to secure almost all internet communications – have security weaknesses;
- clients have privacy concerns on cloud computing protocols since their service providers are not fully trusted. However, it is difficult to formally verify such protocols because classical adversary models that we can find in the literature are not adapted – they assume that corrupted parties are either *honest but curious* (*i.e.* they launch no attacks which would leave any trace) or *malicious* (*i.e.* they launch any attack they want by using all of their resources). However, authorities and service providers are not honest but curious since they could violate protocol specifications; and they are not malicious since they do not want to launch attacks that would be caught – because they (such as NSA) cannot afford the press from negative public opinion and their business (such as Google) cannot afford the loss of reputation.
- Ryan proposed the concept of a new adversary model[2], which is called *malicious but cautious*, to present real-world service providers who will only launch attacks that would not leave readily verifiable evidence. However, this concept has not been formally defined and modelled yet, although the formalisation of this model is desired to verify existing and upcoming protocols.

This proposal focuses on solving the above problems. To achieve our research **goal**, we are attempting to:

- identify properties that a public key certification protocol should satisfy;
- evaluate existing public key certification protocols in the malicious but cautious (MbC) adversary model according to the properties we previously identified;
- propose a public key certification protocol that offers the identified properties;
- formalise identified properties, thereafter formally verify our proposed protocol in the MbC adversary model;
- apply the new protocol on existing communication protocols (*e.g.* the SSL protocol for secure web browsing and the S/MIME protocol for secure email services) in order to enhance their security.

In brief, our research aims to protect users' online communication and users' privacy against malicious but cautious parties (*e.g.* service providers and government agencies).

The rest of this report is organised as follows. Chapter 2 reviews and analyses related work. Chapter 3 states the idea on solving identified problems. Chapter 4 presents the timetable for future work and my academic activities are reported in the Chapter 5.

[2]This work has been presented by Prof. Mark Ryan in the *Security Seminar* at the University of Birmingham, April 2013.

Chapter 2

Analytical Literature Review

2.1 SSL Issues

Netscape Communications developed Secure Sockets Layer (SSL) protocol – which is the predecessor of Transport Layer Security (TLS) protocol [28, 75] – to provide a private and secure connection between two parties communicating over the Hypertext Transfer Protocol (HTTP) [57]. In particular, the design of SSL protocol aims to authenticate domain servers by using an X.509 certificate, to secure private messages by using symmetric key encryption, and to guarantee the message integrity by using message authentication codes (MAC).

Unfortunately, numerous efficient and practical attacks against SSL have been found. We classify the vulnerability of SSL into two categories: protocol weakness and trust model weakness.

2.1.1 Protocol Weakness

Past research shows that SSL protocol is flawed in a variety of ways due to mistakes from developers and users. On the developer side, some domain servers still support outdated cryptographic algorithms (*e.g.* single DES, RC2, RC4, RSA 512 bit, MD4, MD5) [45], although they are known to be insecure [31, 72, 73, 76, 77]. This enables algorithm downgrade attacks – by doing this, an attacker can actively downgrade the strength/version of the algorithms (*e.g.* key exchange algorithm and encryption algorithm) by influencing the algorithm negotiation during the TLS handshake, then break the secure connection by launching attacks on the outdated algorithm [25]. The most recent attack on TLS (with RC4) was presented by AlFardan and Paterson [17]. The misuse of cryptographic security parameter (*e.g.* using the same RSA modulus in different RSA cryptosystems) is another example that harms SSL security [46].

On the user side, they are likely to ignore the security indicators and warnings [61, 74, 81]. For example, some PC browsers have a green or grey lock icon at the address bar that indicates a HTTPS connection. Users can check the public key certificate they got from the TLS handshake by clicking this lock icon. However, most users do not check it and they have bad understandings about the security cues [61, 74]. Also, when an invalid certificate has been received, the client browser will display warnings to notice users that the security of this connection cannot be

guarantee. However, a research shows that more than half of the users clicked through the warning page [61]. So, technically detachable attacks (*e.g.* SSL spoofing attacks and SSL stripping attacks) will succeed with a good probability because of such careless behaviours.

SSL spoofing attack [34] attempts to use a well-positioned and well-disguised pop-up window to cover and forge the security indicator. This attack can be prevented by forcing pop-up windows into new tabs. To protect exchanged messages, most domain servers automatically offer services over HTTPS even if the received request is over HTTP. In SSL stripping attack, attackers normally block the HTTPS response from a server, and then impersonate this server to establish a HTTP connection with users. If users did not check the HTTPS:// in the URL, then the attacker can get client message in clear text. So, SSL stripping attacks [53] are technically detectable but it is unlikely for inattentive users to notice them. In addition, it is not hard to launch such attacks since popular SSL stripping tools (like *sslstrip* [47]) can be freely downloaded from the internet. Many existing solutions (*e.g.* ForceTLS [3], HSTS [39], HTTPS Everywhere [6]) are proposed as browser add-ons to prevent SSL stripping attacks by using a white list. The white list records a set of domain names which support HTTPS protocol. For each connection, such add-ons will enforce a HTTPS connection on the user side if the domain name is included in the white list. However, these browser add-ons only support a few popular web browser. For example, ForceTLS only supports Firefox; HTTPS Everywhere only supports Firefox, Chrome and Chromium; HSTS supports Firefox, Chrome, Chromium, and Opera. Also, the default white list is incomplete and it is hard for clients to manually extend the white list.

2.1.2 Trust Model Issues

In SSL protocol, an X.509 certificate [26] that binds a public key to an identity is used to authenticate domain servers. The certificate issuer is a trusted third party called *certificate authority* (CA). To certify the ownership of public keys, a CA needs to validate the identity of a domain owner by sending a validation email to an email address that can be only accessed by the domain administrator. To pass the validation, the applicant needs to provide the authentication token (or to click the validation link) that is contained in the validation email.

A client accepts a certificate if the certificate issuer is trusted by the client's web browser. Client web browsers initially trust a number of CAs, which are called *root CAs* (*e.g.* Mozilla Firefox browser initially trust 57 root CAs). Each root CA can empower many intermediate CAs and they are fully trusted. So, clients have to trust a large number (around 1500 [5]) of CAs and assume that they are behaving correctly. However, recent attacks [51, 58, 62, 71] show the weakness of such CA/browser trust model. In particular, a compromised CA or a CA who has been forced to issue fake certificates can break the security of SSL protocol. To enhance certificate security, several replacements and improvements have been proposed. We classify them in Table. 2.1 and discuss some popular protocols in the next section.

Table 2.1: Taxonomy of existing solutions

Taxonomy	Existing Proposals
PGP adoption	MonkeySphere [2];
DNS adoption	DANE [40]; CAge (13') [41].
Difference observation	SSL Observatory [5]; Certificate Patrol [1]; Perspectives (08') [80];
	DoubleCheck (09') [18]; CertLock (10') [67];Convergence (11') [48];
	TACK (2012) [49].
Public log adoption	Sovereign Keys (12') [32]; Certificate Transparency (12') [44];
	AKI (13') [42]

2.1.3 Existing Proposals

DNS Adoption

Domain name system (DNS)-based authentication of named entities (DANE) [20, 40] secures connections between clients and domain servers by binding public keys to domain names. This binding is ensured by only allowing CAs to sign a certain scope of domains, and the scope can be verified by using DNS Security Extensions (DNSSEC) [79]. In DNSSEC, domain servers bind their names with a public key that is signed by their parent domain server and can be obtained through DNSSEC protocol. DANE adds a new format of resource record that is called TLSA resource record (TLSA RR) by adding certificate extensions. Clients can get an authentic copy of such resource record through TLSA query in DANE.

DANE improves the certificate security since the compromised signing key of an authority only harms its sub-domains. However, in DANE, a parent domain server can easily issue fake certificates for its sub-domains. In addition, victims or other parties do not have a way to readily detect such mis-behaviours.

In 2013, Kasten, Wustrow and Halderman proposed *CAge* [41] to restrict CA's signing scope based on DNS. Their research (based on the data observed and presented in [38]) shows that only a small number of CAs has signed certificates for top-level domains (TLDs). Based on this observation, *CAge* suggests to limit a CA's signing scope by only allowing a CA to issue certificates on a restricted set of top-level domains (TLDs). CAge limits the possibility of the CA launching a man-in-the-middle (MITM) attack, but cannot completely solve this problem or readily detect MITM attacks.

Difference Observation

In 2008, Wendlandt, Andersen and Perrig proposed a protocol called *Perspectives* [80]. It was proposed to improve *secure shell* (SSH)-style authentication security by asking different observers to detect inconsistent public keys. In 2009, however, Alicherry and Keromytis [18] pointed out that *Perspectives* has privacy issues since observers can get user browsing history, and new servers or new keys in *Perspectives* suffer from an unavailable period. They then proposed

DoubleCheck – a protocol that employs *Tor* [29] to overcome the weakness of *Perspectives*. However, the use of *Tor* adds additional time cost (up to 15 seconds [67]) for each certificate verification. In 2011, Marlinspike also proposed an improvement on *Perspectives* called *Convergence*. It solved privacy issues in *Perspectives* by using local cache and employing onion routing mechanism. In the same year, Soghoian and Stamm [67] introduced compelled certificate creation attack, which enables government agencies compelling CAs to issue fake certificates to surveil citizens. To prevent such attacks, they developed a new browser add-on which can detect the location difference between the certificate issuer and the domain server. In 2012, Marlinspike and Perrin proposed *trust assertions for certificate keys* (TACK) [49] – instead of trusting CAs, it "pins" the public part of self-generated certification keys to its subject. The private part of certification keys is used to certify the corresponded subject's TLS public key. TACK releases clients from having to trust CAs, but it has shortcomings. First, clients have to frequently pin the domain server's TACK public key. Second, a new TACK key has an unavailable period. Third, clients still have to trust a third party for pin sharing. However, it currently is a weak argument since the additional pin sharing protocol has not been specified yet.

All above mentioned protocols can only detect the difference of certificates but cannot distinguish attacks from authentic certificate updates. This reduces the usability since users are not quantified to make correct decisions. In addition, they cannot prevent attacks on captive portal – in the airport or hotel, a client needs to pay for accessing the internet. In this case, the client cannot run above protocols to check the certificate since the internet is not available.

Public Log Adoption

To tackle the problem of fake certificates issued by certificate authority, some of the existing protocols (*e.g.* [32, 42, 44]) record all certificates in a public auditable untrusted log. The log server is able to prove the correctness of its behaviour. Every interested party can verify the proof and monitor the public log to detect mis-behaviours. We list proofs that interested parties would require.

- *Proof of presence* proves that a certificate is included in a public log
- *Proof of extension* proves that the current public log is an extension of previous versions.
- *Proof of currency* proves that the public key of a subject is the latest one in the public log.
- *Proof of non-mis-issuance* proves that no mis-issued certificate is included in the log.
- *proof of absence* proves that no certificate in the log is for the given subject.

We now review some popular protocols that adopt public log.

Sovereign Keys In 2011, Electronic Frontier Foundation (EFF) started a project called *sovereign keys* (SK) [32]. This project proposed a public log based protocol to detect mis-issued certificates.

The basic idea is that domain owners register their certificates with a timeline server which maintains an append-only public log. The public log is linear and the certificates in the log are ordered by monotone increasing serial numbers and corresponded timestamps. The public log prevents any subsequent registrations for the same domain. For each secure connection with

a domain server, the browser needs to contact the timeline server to query the domain server's latest certificate that is included in the log. To reduce timeline servers' workload, distributed mirrors which have a copy of all timeline servers' database are employed.

Sovereign keys perhaps is the first protocol that uses a public auditable and append-only log to detect mis-behaviours. Unfortunately, sovereign keys still has security problems. First, it is vulnerable to "domain stealing attack" – a malicious CA could register sovereign keys for a domain if its genuine owner has not registered it yet, then this CA gets control of this domain's sovereign keys. Second, all proofs are $O(n)$ due to the linear log structure. In other words, to detect misbehaviours, clients or domain servers have to download the whole log and check all certificates in it. This is inefficient and impractical. Last, the default mirror has to suffer from heavy workflow since most clients will not change browser's default setting. In addition, the browser history will be leaked to mirrors since clients need to ask a mirror for each connection.

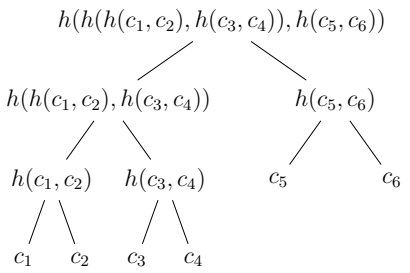
Certificate Transparency In 2012, Laurie, Kasper and Langley proposed *certificate transparency* (CT) [44] to detect mis-issued certificates. *Certificate transparency* enhances the efficiency of mis-behaviour detection by using an append-only Merkle tree [50] to implement the public log. Main entities in CT are listed.

1. *Log server* is the party who maintains a public log. The log server has a pair of signing key (sk_{log}) and verification key (pk_{log}).
2. *CT log* is an append-only data structure, which stores validated certificates at leaf notes of a Merkle tree [50] (see Figure. 2.1). Each certificate is added chronologically at the right by extending the tree. An example of the CT log is showed in Figure 2.1, in which, c_i is the i^{th} certificate and h is the cryptographic hash function. The relevant proofs are listed below:
    - to prove the *proof of presence*, the log provider only needs to provide one hash value of each layer (*i.e.* nodes of same depth in the tree). Hence, the size of this proof is $O(log\ n)$;
    - *proof of extension* is similar as the proof of presence and this proof is also $O(log\ n)$, where n is the amount of new appended entries;
    - *proof of currency* is $O(n)$ since the proof needs to show all entries added after the given certificate;
    - *proof of non-mis-issuance* and proof of absence are $O(n)$ since the proof needs to show all entries in the log.
3. *Monitor* checks the proof of non-mis-issuance on behalf of domain servers.
4. *Auditor* checks the proof of presence for a given certificate and checks the proof of extension.

The process of CT is reviewed as below.

1. Certificate authorities should submit all issued certificates to a public log server.
2. For received requests, a log server appends each valid certificate in the log and returns a Signed Certificate Timestamp (SCT) to the requester. SCT contains the time of submission and a signature on the certificate and the timestamp. It can be later used as evidence to

Figure 2.1: An example of CT log



prove that the log server has verified the certificate and will append it in the log within a specified time period[1].

3. For a given certificate, a client accepts it iff a valid SCT is provided together with the certificate.
4. If a client doubts the validity of a certificate, s/he can ask the auditor to check the proof of present.

Certificate transparency is a good way to detect unexpected behaviours from CAs. Compared with the sovereign keys, CT is more efficient on the proof of present and proof of extension. However, many security issues remain unsolved.

First, we identify two novel attacks on CT, namely *sacrifice attack* and *rogue "trusted" party attack*.

- In CT, log server accepts a certificate if the submitter is a legitimate certificate authority. In other words, an intermediate CA is able to convince log server to accept a fake certificate, but it will be caught by the monitor. However, sacrifice attack allows an attacker[2] to launch MITM attacks by issuing fake certificate but without being caught.

  To launch sacrifice attack, the attacker generates temporary trust of chains (*e.g.* Figure 2.2). In which, each non-root node[3] is an intermediate CA. Then, the attacker orders one intermediate CA (*e.g.* $CA_{L3,311}$) to sacrificially launch attacks by issuing fake certificates. The log server will accept the fake certificate that will be valid for a period[4] before it has been detected. $CA_{L3,311}$ will be caught and be removed from the trust list after be detected, but the real attacker will not be blamed since there is currently no clear way in CT to prevent this attack.

- A rogue "trusted" party is an organisation (*e.g.* a university) who could be trusted by a certain range of people (*e.g.* the employees and students). This small rogue party maintains its own log. This organisation can put itself in the browser's trusted CA/log list since it provides and controls computers to employees/students. In this case, maybe the monitor

[1]This time period is called "Maximum Merge Delay (MMD)"

[2]This attacker could be (a) the government who forces a CA to do so; (b) hackers who comprise one or more CAs' signing key; and (c) a malicious but cautious root certificate authority

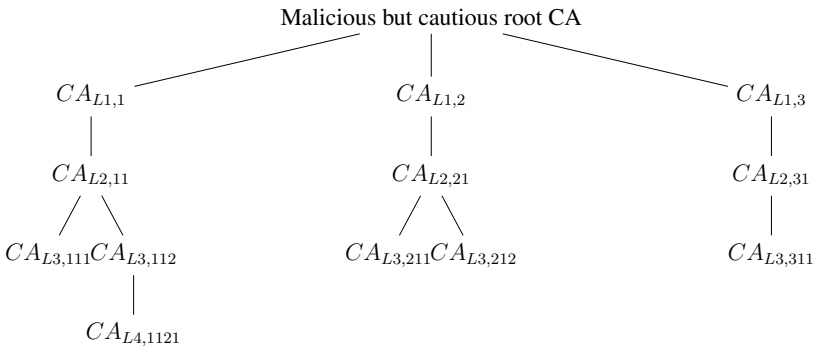
[3]The label of each node indicates the location of a node in the chain. For example, $CA_{L3,211}$ is a node at level 3 of a chain with parent node $CA_{L2,21}$ and grandparent node $CA_{L1,2}$

[4]The length of this period equals to Maximum Merge Delay (MMD, see [44] for detail) plus the time period needed for fake certificate detection.

is not even aware of the existence of such organisations. Hence, the monitor could not check the organisation's log. So, the organisation can play MITM attack and record its misbehaviours in the log, but without being detected.

In addition, the proof of absence, proof of currency and proof of non-mis-issuance are all inefficient ($O(n)$). Furthermore, checking the proof of presence is optional. This opens a window for a certificate authority, which colludes with a log server, to issue a fake certificate that will not be recorded in the log. Such fake certificate could then be used for a man in the middle attack.

Figure 2.2: An example of temporary trust chains



Certificate Issuance and Revocation Transparency Ryan [59] proposed an improvement on certificate transparency, called *certificate issuance and revocation transparency* (CT+), to enhance the efficiency of proofs: instead of using one Merkle tree to maintain the log, Ryan proposes to use two Merkle trees. The first tree is called "ChronTree" – which structure is as same as the structure of log in certificate transparency. The second tree is called "LexTree" – which is organised as a binary search tree storing lex-data at every leaf and non-leaf node in lexicographic order of entity name (see Figure 2.3[5]). Lex-data contains a user's identity and a list of associated certificates. Hence, the users can easily find all the public keys of an entity.

Insert and revoke actions are required to be done together on both trees. The size of each proof is presented in Table 2.2, in which, the proof of consistency proves that the ChronTree and the LexTree are presenting the same set of data. Table 2.2 shows that the proof of consistency

Table 2.2: Cost of proofs

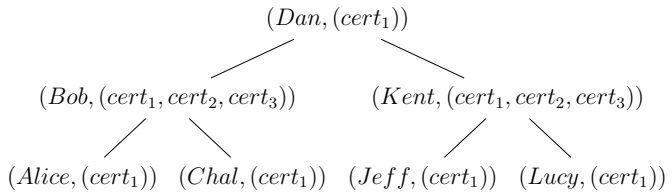
Proof of	ChronTree	LexTree
presence	$O(log\ n)$	$O(log\ n)$
currency	$O(n)$	$O(log\ n)$
mis-issuance	$O(n)$	$O(log\ n)$
absence	$O(n)$	$O(log\ n)$
extension	$O(log\ n)$	$O(n)$
consistency	$O(n)$	

[5]The hash values stored at leaf and non-leaf nodes are not shown in the figure.

is $O(n)$. To solve this problem, two ways are concerned. The first way is to introduce a new party called "auditor", who monitors larger proofs that the log server maintains data structures consistently. The second way is to ask users to collaborate together. Each of collaborated users randomly checks that some leaves in the ChronTree are also in the LexTree.

Thanks to this new log structure, CT+ improves the efficiency of misbehaviour detection. However, CT+ is still vulnerable to sacrifice attack and rogue "trusted" party attack. Moreover, the new log structure is not sufficient to ensure interesting properties that we will discuss in the next chapter.

Figure 2.3: LexTree



Accountable Key Infrastructure In 2013, Kim, Huang, Perrig, Jackson, and Gligor proposed *accountable key infrastructure* (AKI) [42] – a new public log based protocol that tries to reduce the level of trust in certificate authorities. The public log in AKI is a binary hash tree which stores entries at leaf nodes in lexicographic order. Each entry is an AKI certificate; expired or revoked AKI certificates in the log will be updated or purged. AKI enhances the efficiency of currency proof, absence proof and non-mis-issuance proof. However, the extension proof is $O(n)$ and AKI is vulnerable to domain stealing attacks and rogue "trusted" party attack.

The public log-based protocols do not themselves prevent mis-issuance, but they ensure that interested parties (particularly domain owners) have the ability to detect mis-behaviours. However, as discussed above, existing schemes have various weaknesses. Also, to the best of our knowledge, surprisingly, none of above existing protocols has been formally verified. One possible reason that we are going to discuss is that the existing well-defined adversary models cannot properly present the real-world adversary that these protocols need to face.

2.1.4 Adversary Model

Adversary model defines internal and external adversaries which could attack a system. Classically, two main categories of adversaries have been considered:

- *Malicious* adversaries are active attackers – they can launch any attack they want regardless of protocol instructions. Security in the presence of malicious adversaries provides the strongest security guarantee. However, it is difficult to achieve such security level and the cost of such protocols is normally high.
- *Honest but curious* adversaries are passive attackers who follow protocol instructions, but attempt to learn information by analysing eavesdropped messages exchanged among participants. In other words, honest but curious adversaries do not launch attacks which would

leave any trace (whether verifiable or not). Protocols that are secure in this adversary model provide a security guarantee for systems that all participants in which trust each other. However, it is insufficient for most systems.

Recent discoveries identified that some trusted parties (*e.g.* certificate authorities) actively launched attacks [25, 67, 68, 71] and some trusted parties (*e.g.* email service providers) handed clients' private data over to government agencies [7, 12, 23, 36, 62]. Classical adversary models cannot properly model such corrupted trusted parties: honest but curious adversary model is too weak – they do actively launch attacks and they do expose client private data (that they should protect) to others; and malicious adversary model is too strong – they only launch attacks that would not be caught since they do not want to lose their reputation. So, a new adversary model that can detect mis-behaviours from corrupted trusted parties is desired.

To the best of our knowledge, the first idea of mis-behaviour detection was mentioned by Franklin and Yung [35] in 1992. Later, Canetti and Ostrovsky proposed the *honest-looking* adversary [24] in 1999; and Hazay and Lindell formally presented the *covert* adversary [37] in 2008. All of them are proposed for multi-party computation (MPC), which is the case such that mutually distrustful parties need to perform a joint computation on their private inputs. A secure MPC requires that the adversary – which could be a single attacker or colluded attackers – can learn nothing about honest parties' inputs. Recently, Ryan presented the concept of *malicious but cautious* adversary[6] for cloud computing services.

The concept of these adversary models are similar – they are proposed for the adversary who attempts to cheat, but does not want to be caught since they cannot afford the cost of losing their reputation. However, they have different focuses. The focus of the passive/active mis-behaviour detection [35], the honest-looking adversary model, and the covert adversary model is on multi-party computation protocols – the corrupted parties are untrusted. However, the focus of malicious but cautious adversary model is on service providers – users need to trust them for their services (*e.g.* to use the public key certification service, users need to trust certificate authorities). We also concern following differences:

- malicious but cautious adversary model concerns that client data are transferred in an insecure communication channel, while other adversary models (*w.r.t* [35], [37], and [24]) assume a secure communication channel between participants;
- honest-looking assumes the willingness to be caught; covert adversary model does not assume the willingness to be caught since they require higher probability for mis-behaviour detection. Both of them detect mis-behaviours when the cheating occurs. However, malicious but cautious adversary model provides permanent evidence – every mis-behaviour is recorded; everyone can view and verify evidence of mis-behaviours after treating occurred.
- service providers in malicious but cautious adversary model can prove that they did not mis-behave, while participants in other adversary models cannot.
- covert adversary model places the onus of mis-behaviour detection on the protocol since maybe participants do not have the ability to detect mis-behaviours; while participants in the malicious but cautious adversary model can detect mis-behaviours if they want.

[6]This work has been presented by Mark Ryan in the *Security Seminar* at the University of Birmingham, April 2013.

- the strongest formulation of covert adversary model requires that the protocol can detect cheating, and if the cheating has been detected, then the adversary learns nothing. With service providers, however, it is too hard to be achieved. While the focus of malicious but cautious adversary model is on recording evidence of all behaviours into a publicly auditable and verifiable log.

Thus, the above discussed adversary models have different focuses, different security requirements, and different security guarantees. The comparison also shows that the malicious but cautious adversary is more suitable for presenting corrupted service providers.

2.2 Email Security

Email services have been widely used in recent a few decades. The basic properties of email services are integrity and availability. Integrity guarantees that what a receiver received is what a sender sent. Availability enables users to get access to their email services at anytime and anywhere. One desired property is confidentiality which requires the email is only accessible by the sender and receiver, but current practical email services cannot perfectly achieve this requirement yet. Other properties and services – such as spam, phishing mail, and Trojan detection – are also desired for higher requirements.

Simple mail transfer protocol (SMTP) was proposed in 1982 [55] as an email standard. The current version was updated in 2008 [43]. With SMTP, an attacker can easily impersonate as someone else to send an email. This is because some SMTP servers do not check the authenticity of senders. To solve this problem, SMTPS has been introduced to secure the transport layer of SMTP by using SSL/TLS [28]. However, SMTPS does not support end-to-end encryption.

To achieve end-to-end encryption, secure/multipurpose Internet mail extensions (S/MIME) [56] has been introduced as a standard to adopt public key encryption and cryptographic signature on the MIME format email data. With S/MIME, a sender can sign an email by using her signing key and encrypt an email by using the receiver's public key. The receiver can decrypt the email by using her secret key and verify the identity of the sender by checking the signature. Thus, the sender authentication and end-to-end encryption are achieved. However, S/MIME has inherent security concerns since certificate authorities (CAs) have been involved. CAs offer two types of services for user registration. The first type is that a CA generates both a private key and a certificate of public key for users, for example, *Comodo* individual email certificate service. Since users' secret keys are generated by a CA, thus this CA has the power to decrypt all users' emails. The second type is that users generate secret keys by themselves, and ask certificates from a CA by uploading their public keys. This secures users' emails in some sense. However, any CA could play MITM attacks by providing a fake certificate and impersonate as the sender/receiver.

To solve trust issues of certificate authority, a web of trust based mechanism called "Pretty Good Privacy (PGP)" has been described in 1992 and specified in OpenPGP [65]. The main idea is that a user creates his/her own pair of keys, then spreads the public key to a set of users who could certify the public key by signing it. To check the validity of a public key, instead of trusting CAs, users verify all signatures on this public key and determine whether to trust it. Thus, users in PGP achieve authenticity and end-to-end encryption by maintaining and sharing a trust web in their group. However, since PGP users are desired to understand some cryptographic primitives

and how PGP works, the PGP has not been widely used [15].

The concept of identity-based encryption (IBE) was invented by Shamir in 1984 [64] to simplify certificate management in e-mail systems. The basic idea is that the public key in IBE could be an arbitrary string – such as a domain name, an organisation name, and an email address. Hence, with IBE, instead of searching certificates from servers in lightweight directory access protocol (LDAP) [63], a sender can encrypt a message by using receiver's email address as her public key. The corresponding secret key is generated by an identity provider who therefore has all users' secrets. The construction of IBE remained as an open problem until a pairing-based IBE scheme was proposed by Dan Boneh in 2001 [22]. However, the key escrow problem has not been solved.

Certificateless encryption (CLE) was proposed in 2003 [16] to solve the key escrow problem in IBE. In IBE, a user's secret key is generated by an identity provider who has to be fully trusted. So, the identity provider knows all users' secret keys. While in CLE, a user's secret key consists of two parts, the first part is generated by a key generation centre (KGC) and the second part is a random value generated by the user. Thus, a user's secret key is private against KGC. A survey of CLE and its security models has been provided in [27].

In another direction, the idea of using an email gateway to manage emails for an organisation is introduced. For outgoing emails, the gateway can sign them with organisation's signing key and encrypts them with receivers' public keys. For incoming emails, the gateway checks the signature and/or decrypts mails for receivers. In addition, the gateway is also expected to detect spams, Trojans and malware. The advantage of using email gateway is that users are protected from outside attacks and users do not need to manage keys for email signing and/or encryption. The disadvantage is that users have to trust the email gateway.

2.3 Internet of Things Security

Internet of things (IoT) [4] is the idea of connectivity for anything at anytime and anywhere. It aims to build a world-wide network of interconnected objects that are uniquely addressable with communication protocols [19]. The initiative idea derives from a perspective of thing-oriented applications, such as radio-frequency identification (RFID) tags. The Auto-ID Center at MIT [69] has a big contribution on spreading the concept of IoT [11].

The idea of connecting everything to the internet will play a leading role in the near future by involving billions of objects, such as food packages, furniture, paper documents, and personal belongings. The Research Councils UK (RCUK) considered 50 billion connected objects of IoT in 2020 with hundreds of billions of pounds a year potential added value [9]. The US National Intelligence Council (NIC) listed Internet of things (IoT) as one of the six "Disruptive Civil Technologies" which have potential huge impacts on US national power out to 2025 [69].

In the meantime, IoT creates new concerns – the huge number of connected objects, the inestimable size of shared data, and the enormous potential profit of IoT markets rise the security and privacy concerns to an unprecedented level [10, 70, 78]. The data security is becoming a top priority concern when we are moving towards to the IoT. For example, the spread of viruses, worms, Trojans, etc. among the connected objectives in IoT is hard to be controlled [30]. Also, the insecure IoT system is more dangerous. For example, attackers can control the direction

of moving vehicles, can change the operation of medical machines, can turn off the alarming systems of organisations, and even worse, maybe they can influence a country's infrastructure – such as smart grid, gas pipelines, power station, and water systems. The privacy is another increasingly critical issue [10, 52, 70, 78]. For example, connected shoes, shirts, glasses, and furniture could leak the owner's buying habits; the RFID tag could leak your personal location since you must be not far from the RFID reader; and all IoT service systems could be used as surveillance systems.

We summarise properties for IoT systems.

- Authenticity. It requires both client authentication and server authentication. Client authentication guarantees that a client/device has the authority to use associated data and services; server authentication guarantees that the device is communicating with the correct server.
- Confidentiality. It guarantees that anyone other than the communicating parties can learn nothing about the content exchanged data.
- Integrity. It guarantees that what a receiver received is what a sender sent.
- Anonymity. Devices and exchanged data must be somehow associated to an identity. The anonymity protects client privacy (*e.g. individual's location and travel routes*) by hiding the link between devices and clients.

Intuitively, some existing protocols could be applied in IoT to offer the properties. For example, the secure multiparty computation (MPC) or Tor might be adopted to protect user privacy; the public key certification mechanism might be applied to guarantee authenticity; the TLS might be helpful for providing integrity and confidentiality. However, the security and privacy concerns in IoT with further questions – such as who should and how to monitor the monitor? How to protect clients from surveillance systems? How to balance the client privacy and public safety protection? Who can and how to use the collected data? – will be a potential direction for future research.

Chapter 3

On-going and Future Work Statement

This proposal aims to enhance the security and reliability of public key certification. As discussed in Section 2.1.4, we consider malicious but cautious adversaries – who only launch attacks that would not leave readily verifiable evidence. With the perspective of malicious but cautious adversaries in mind, we started from identifying desired properties for public key certification protocols. Based on the identified property, we are evaluating existing public key certification protocols. In particular, we identified attacks on and security weakness of popular protocols. To address identified problems, we explain our idea on designing a new certification model that satisfies desired properties. This model can be applied to secure the SSL protocol, to provide end-to-end encryption email services, and to secure IoT services.

3.1 Evaluation on Public Key Certification Models

Regarding the perspective of malicious but cautious adversary model, we identified common properties and novel properties.

Common properties are properties that have been concerned by existing protocols and evaluations [25, 54]. We list some of them.

1. *Offline verification.* It allows clients to verify a certificate without requiring additional online verification. For example, with the current certificate authority/browser trust model, clients only need to verify the signature on a certificate. So, it satisfies offline verification. In contrast, with *sovereign keys* [32], clients have to make additional online queries to verify certificates.
2. *Built-in key revocation.* This feature allows that a domain server to revoke a key without requiring an additional protocol (*e.g.* certificate revocation list (CRL) or online certificate status protocol (OCSP)).
3. *Independent trust.* It requires that one or more compromised trusted parties will not influence an entity which did not choose to trust them. For example, current certificate authority/browser trust model does not satisfy this feature since that any compromised CA can issue certificates for any domain server, and the certificates can be accepted by web browsers without any warning.

4. *Scalability.* This feature enables a system to handle increasing workload or increasing number of users.
5. *domain server changing protection.* This protects a new domain server from its previous owner. In other words, the previous owner should not have the ability to influence the key/certificate of the domain. Note that owners of famous domains (*e.g.* Google.com, Facebook.com) are rarely to be changed. However, the owner of small domains could be changed in every a few years.
6. *Usability.* It enables clients to use services easily. For example, Certificate Patrol [1] alerts and asks clients to a make decision when different certificates are detected. However, there are too many false alarms and normally clients will click through them [25]. That means, clients do not want to make decisions and they are not qualified to make correct decisions.

We also concern following novel and desirable properties.

- *Country neutrality* is a property that the public key identification system should not be controlled or influenced by any single country. In particular, two aspects are concerned.
    1. Authorities (*e.g.* root CAs, timeline servers, and log providers) should not be dominated by a single country. *E.g.* current CA model is American-dominated.
    2. Government agencies that have compelled authorities in one country should not be able to use fake certificates without being readily detected.

  In the presence of cyber-security tensions between nations, it would be better to have country-neutrality technology. However, to the best of our knowledge, all existing protocols have not considered it yet.
- *Canonical signer* is the property that enables anyone to easily identify parties that are authorised to establish key authenticity. However, in existing protocols, a user has no way of knowing which CAs are authorised to issue a certificate for a particular domain.
- *Anti-monopoly.* This property allows any entity to freely remove their trust from any party, while without influencing the function of internet services. A counter example is the current certificate-based model: if you choose not to trust a CA, then you will receive security warnings for all websites which public keys have been signed by this CA, since the current certificate-based model does not have a backup plan for this case.
- Resistance to *sacrifice attacks* is in particular for systems that face attackers (*e.g.* service providers) which do not launch attacks that would leave readily verifiable evidence of the attack. In sacrifice attacks, an attacker can create temporary identities which can be sacrificed and replaced with new ones. To avoid to be caught in a system, the attacker orders this created identity to sacrifice and launch attacks for him. It becomes to an important threat for systems if an attacker can efficiently generate and order as many expendable identities as he wants to launch attacks, while not being detected or blamed. For example, a root CA can create many intermediate CAs in different level of a CA trust chain that anchors on the root CA, and then order some of intermediate CAs to launch attacks, to be caught, and to be blamed.
- Resistance to *rogue "trusted" party attack.* shares the same idea as man-in-the-middle (MITM) attacks, but attackers here are small organisations (*e.g.* universities) – they are

trusted by a certain group of users (*e.g.* students and employees) and they are maintaining their own public logs. Most systems that use public log can readily detect misbehaviours; however, it is possible that a rogue party did misbehave and did record misbehaviours in the log, but without being detected. Because the public key owner did not be aware of the existence of such rogue parties.

The Table 3.1 summarises our on-going evaluations.

Table 3.1: Evaluation of proposals

Desired Features	Existing Systems										
	X.509-PKI	Perspectives	Convergence	DoubleCheck	TACK[1]	CAge	DANE	Sovereign Keys[2]	CT	AKI[2]	CT+
Off-line verification	√	×	⊗[3]	×	⊗[3]	×	√	×	⊗	×	×
Build-in Key revocation	×	×	×	×	×	×	√	√	⊗	√	√
Independent trust	×	×	×	×	√	×	√	√	×	√	×
Distributed trust (log synchronisation)	—	×	×	—	⊗[4]	—	—	√	×	×	×
Scalability	√	√	√	√	√	√	√	×[5]	√	×	√
Self-signed certificate supporting	×	√	√	×	√	×	√	√	×	×	×
domain server changing protection	×	√	√	√	√	×	√	×	×	×	×
Usability	√	×	√	×	√	√	√	√	√	×	√
Seamless Key Changing	√	×	×	√	×	√	√	√	√	×	√
Resistance to MITM Attack[6]	×	⊗[7]	⊗[7]	√	√	×	×	√	⊗[8]	√	⊗[8]
Resistance to DDOS Attack	√	√	√	√	√	√	√	×	√	×	×
Country neutrality	×	×	×	√	√	×	×	×	×	×	×
Anti-monopoly	×	√	√	×	√	×	×	×	×	√	×
Canonical signer	×	×	×	×	×	×	⊗	×	×	×	×
Resistance to Sacrifice Attack	×	√	√	√	√	×	√	⊗[9]	×	√	×
Resistance to Rogue "trusted-party" attack	×	√	√	√	√	√	√	√	×	√	×

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

[1] TACK may be vulnerable to cache poisoning attack.
[2] The subject is vulnerable to domain stealing attack.
[3] The subject requires online updating for the public key/certificate that cannot be found in the local cache.
[4] This depends on the additional protocol for pin sharing.
[5] The default mirror will be a bottle-neck since clients are rarely changing their browser setting.
[6] In order to distinguish MITM attack from sacrifice attack, here, we only concerns the attacker itself plays MITM attack.
[7] The attacker needs to intercept all communications of a victim.
[8] An attacker who directly plays MITM attacks will be detected, however, s/he can avoid to be caught by launching MITM attacks through Sacrifice attack.
[9] The subject is only vulnerable to this attack if the CA system is involved. The employee of DANE can prevent this attack.

3.2 DTKI - A New Public Key Infrastructure

We first briefly explain our idea on achieving some of properties that are hard to be satisfied. To achieve country neutrality, one possible idea is that clients check the hash value of a public log by asking other several distributed public logs. If the hash value can be confirmed by different logs that are located in and controlled by different countries, then clients accept certificates that is included in the log (which hash value is confirmed). In addition, any misbehaviour will be readily detected and it is undeniable since all logs are cryptographic signed. To achieve canonical signer, we are concerning to design a DNS-like certification model.

Probably the idea of cross-signed certificates could help us to achieve anti-monopoly. For example, in the case that clients have a list of certificate authorities in their web browser; and clients want to remove a few certificate authorities. Then it is possible that a domain owner asks different authorities to sign the same public key. Clients accept this certificate if a fixed minimum number of signatures are valid. However, another concern is that the cost of certification is rising since a domain server needs to pay for more than one certification services. Also, the trade off between the number of certificates a domain needs to apply and the usability of such protocols requires more considerations and research.

Resistance to sacrifice attack could be satisfied in two ways. The first way is making the protocol accountable. In other words, we should have a clear way to judge who we should blame if cheating is occurred. The second way is to make attackers difficult to create intermediated trustworthy and expendable identities.

Resistance to rogue "trusted" party attack could be satisfied by offering distributed trust. Our idea is to integrate all logs, and distribute the full records. In this way, a participant can ensure the checking result by only checking one mirror of centralised records. With this idea, misbehaviours from a rogue MbC party will be detected.

Based on above thoughts, we proposed a new public key infrastructure called "DNS-based transparent key infrastructure (DTKI)". The basic idea of our protocol is that the parent domain servers, currently used in DNS, will record public keys of their child domain servers in a public monitorable and auditable log. Moreover, internet users will be able to check the public key of a domain by querying distributed log servers that all maintain the same log. We are also confident that our new system could be easily adopted by the current DNS.

3.2.1 Main Entities

- A **Second level domain (SLD)** is a subdomain of a top level domain (TLD). In DTKI, an SLD may or may not maintain a CT+ log. For example, ".co.uk"[1] does need a log for its sub-domains (*e.g.* example.co.uk), but "example.com"[2] does not[3].
- A **Top level domain (TLD)** is one of the highest level domains in DNS. In DKTI, each TLD maintains a CT+ log named $Tlog$ for subdomains. $Tlog$ records second level domain

[1] In this case, ".uk" is the TLD and ".co" is the SLD.

[2] In this case, ".com" is the TLD and ".example" is the SLD.

[3] We assume that such SLD will behave correctly for its sub-domains. This assumption is based on the fact that such domains share profit and reputation with sub-domains, *e.g.* "example.com" should not cheat or attack "mail.example.com".

(SLD) names and their public key information.

- The **Authority log server (ALS)** is a unique and independent log server. It maintains an authority log ($Alog$) and keeps a full copy of all logs from TLDs and SLDs. There is only one $Alog$ in the DNS. $Alog$ records a list of authentic TLD/SLD log information (*e.g.* (subject name, the hash root value of the log, the time that this version of log was created)) that has been signed by the corresponded TLD or SLD.
- **Mirrors** are servers which maintain a full copy of $Alog$. In other words, mirrors are distributed $Alog$. There are lots of mirrors. Different mirrors can be identified by their unique mirror identity (MID) that is a concatenation of country code and serial number. Mirrors should register with the country code TLD (ccTLD) according to their locations.

Before we describe protocol details, we provide a high-level overview: Alice is a client of example.com that is owned by Bob. Now Alice wants to send sensitive data (*e.g.* ID, password, personal information) to Bob's domain that supports a secure protocol (*e.g.* HTTPS). Alice gets Bob's IP address through DNS, checks the latest hash value of the log with mirrors that are located in different countries, and asks Bob's encryption key from one mirror. Alice accepts this encryption key if the mirror can provide a valid currency proof. If Bob's domain does not support a secure protocol and does not have an encrypt key, then the mirror should send the proof of absence to Alice.

3.2.2 Registration Process

We assume both Alice and Bob have an authentic copy of the verification key of .com and ALS. The process is as follows:

- Bob creates a pair of keys (EK_{ex}, DK_{ex}) for example.com as the TLS encryption and decryption key, respectively. Then Bob requests to register public keys for example.com by sending $(reg, example.com, EK_{ex})$ to .com.
- After receiving the request from Bob, .com sends a message that contains the received request and an authentication token, to the administrator of example.com through a private channel.
- If Bob can provide the authentication token, then .com accepts the request and sends the confirmation back to Bob. The confirmation is $Sig_{SK_{.com}}(example.com, EK_{ex}, TS_{conf})$, where $Sig_{SK_{.com}}$ is the signing function with the signing key $SK_{.com}$; and TS_{conf} is the timestamp which indicates the confirmation time.
- .com updates its log periodically with interval $it_{.com}$, and submits the update evidence ($UE_{.com} = Sig_{SK_{.com}}(h(log_{.com}), TS_{.com})$) and new added records since last update to ALS.
- For each request, ALS checks update evidence, then adds new records into $log_{.com}$, and checks whether the hash root value of this log is as the same as what Bob sent. This process also ensures the proof of extension and the proof of consistency. If these two hash values are exactly the same, ALS signs the hash root value of updated log together with a timestamp TS'_{conf}, then sends the signature to this TLD as evidence of acceptance.

- ALS updates $Alog$ periodically with interval it_{Alog}. Each mirror checks the update evidence ($Sig_{SK_{ALS}}(h(Alog), TS_{Alog})$), and downloads new updates if update evidence are valid. After added new records into logs, mirrors should check whether the hash root value of the updated log is as the same as what ALS presented. This process also ensures the proof of extension and the proof of consistency.

Bob can query fixed number of mirrors that are located in different countries for the latest hash root value and update evidence. Then Bob checks the proof of non-mis-issuance of $log_{.com}$ with one mirror. The new key should be available in mirrors if the current time $t > TS_{conf} + it_{.com} + it_{Alog}$. Otherwise, Bob can claim that .com misbehaved by providing the signed confirmation. .com could defence it by providing the update confirmation that was given by ALS with the proof of present that proves the key is appear in the log which has been accepted by ALS. If so, ALS mis-behaved.

3.2.3 Public Key Querying Process

Alice asks IP address of example.com through DNS as normal. In the meantime, she makes conversions as below:

1. Alice $\rightarrow$ mirrors: Alice sends request message m_1 to randomly selected fixed number of mirrors that are located in different countries to query update evidence of $Alog$. Recall that update evidence is the signature on hash value of a log and the timestamp of update time.
2. Mirrors $\rightarrow$ Alice: mirrors answers a signed $m_2 = (h(Alog), UE_{Alog})$, where UE_{Alog} is update evidence of $Alog$.
3. Alice $\rightarrow$ mirror: if received answers with the latest valid update evidence from mirrors are the same, Alice sends a request message m_3 to one mirror to query $EK_{.com}$.
4. Mirror $\rightarrow$ Alice: the mirror sends the signed $m_4 = (h(log_{.com}), TS_{.com}, UE_{log_{.com}}, Proof_p, EK_{ex}, Proof_c)$ to Alice, where $UE_{.com} = Sig_{SK_{.com}}(h(log_{.com}), TS_{.com})$; $Proof_p$ is the proof of presence that proves $h(log_{.com})$ is in the $h(Alog)$; $Proof_c$ is the proof of currency for EK_{ex}. If a request domain name is not in the log, then a proof of absence for this domain is required.
5. Alice validates the $h(log_{.com})$ by checking the signature and the proof of presence (w.r.t. $Proof_p$), then validates EK_{ex} by checking the proof of currency. Alice accepts the encryption key of example.com if proofs are valid.
6. In the secure protocol handshake (*e.g.* TLS), an encryption key that has not been included in the log can still be used iff a fresh confirmation that was issued by its parent domain is provided. Here, freshness means that the current time t satisfies $t < TS_{conf} + it_{.com} + it_{Alog}$, where TS_{conf} is the timestamp in the signed confirmation. This key will be cached and it will be verified at the next time.

3.2.4 Monitoring Process

- Each TLD periodically checks update evidence of $Alog$ from mirrors in different locations to verify whether the latest logs in different mirrors are consistent; if it does consist,

each TLD checks the proof of presence of $h(log)$, where $h(log)$ is the log that the TLD maintains. SLDs which maintain its own log should run similar checking process.

- For each valid update request from TLDs and SLDs, ALS checks the proof of consistency for new added entries since last update. TLDs and interested SLDs or other parties can also maintain a mirror to check the proof of consistency. Note that popular SLDs (*e.g.* google.com, facebook.com and ebay.com) and even some sub-domains of SLD (*e.g.* ebay.co.uk) are recommended to maintain a mirror and to check consistency proof; it is reasonable that other small and unpopular SLDs choose to do not monitor it. Internet users neither need to check the extension proof nor need to check consistency proof since other parties will do it, and the checking result is ensured over the world because of distributed mirrors.

3.2.5 Key Revocation

The process of key revocation/update is similar as the key registration process. For example, if Bob wants to revoke the encryption key of example.com and to certify a new one, Bob needs to send the request to .com. After validating the request, .com adds Bob's new key into $log_{.com}$ and sends a signed confirmation. The CT+ log ensures the revocation process since extension proof and currency proof are $O(log_n)$.

3.2.6 Another Thought

The proposed DTKI does not satisfy offline verification. An earlier idea to overcome it is adopting public log based DANE-like protocol. In practice, a client gets the public key and proofs directly from an additional extended DNS resource record. The new format of DNS record contains (domain name, VK, EK, Valid period, Proofs). In which, the VK is verification key and EK is encryption key; Valid period contains start time and end time; Proofs contain (1.) the update evidence of the log; (2.) proof of currency or proof of absence. To verify the current hash value of logs that is presented in the resource record, a client needs to cache the hash value of each log that is included in the currency/absence proof, and periodically check the extension proof with different mirrors. This is left as a future work.

3.3 Potential Applications

This certification model can be applied to enhance the security of most protocols that requires certification system. For example, it can be applied to the SSL protocol for secure web browsing; it can be applied to S/MIME protocol for secure email services; and it can be applied to secure potential IoT services.

3.3.1 SSL Enhancement

The adoption of our certification model will solve current certificate trust issues in SSL since all mis-issued certificates in our model are visible, verifiable, accountable, and undeniable. Also,

the adoption of our model in web browsing service can prevent SSL stripping attacks. This is because that (a.) all web certificates will be recorded in the distributed log; (b) the log supports efficient absence proof. So, the SSL stripping attack can be prevented if clients only connect to the server over HTTP if a valid absence proof is provided. This is as similar as white list based approach (*e.g.* HTTPS Everywhere), while there are some differences. First, public log provides a universal and complete white list that everyone can use it through our protocol (either online or offline); while for normal white list, different users have different incomplete white lists in their web browsers. Second, our approach does not have trust-on-first-use security concern, while the white list based approach does. Third, for usability, users with our approach do not need to add a HTTPS URL manually, while some white list based solutions do require manually operation.

3.3.2 Secure Email

This model could also be applied to end-to-end encryption email services by adopting our certification model into S/MIME protocol. However, two problems require further consideration. The first one is the distributed log will leak the email address to systems which send spam mails. Perhaps this problem can be solved by randomly stuffing invalid email addresses and certificates in the public log. By doing this, an attacker is unable to distinguish a valid email address from invalid ones. The second one is that spam filter will not work since all emails are encrypted and private against email service providers. These two problems are left as future work.

3.3.3 Secure Internet of Things

In the near future, billions of devices in IoT systems need public keys to securely their communications. To protect client privacy and data confidentality, end-to-end encryption might be achieved by adopting our certification model. However, the problem like how to design IoT protocols that satisfy desired properties (*e.g.* authenticity, confidentiality, integrity, and anonymity) is remained and left as future research.

3.4 Formal Verification

In the past research, trusted parties are assumed to be honest and launch no attacks on protocols. However, as discussed in the previous chapter, this assumption does not accurately represent the real environment. Currently, perhaps some of existing protocols are secure against malicious but cautious parties; however, to the best of our knowledge, no formal verification, which concerns malicious but cautious parties, has been done yet.

To formally verify protocols against malicious but cautious parties, we will start from verifying our proposed protocol. We will first formalise the employed public log and its relevant proofs. The formalisation of identified properties will be considered afterward. Based on the exploration, we will formally verify and evaluate existing protocols.

3.5 Evaluation of the work

We consider following contributions:

- the identified and formalised novel properties will arouse researchers' attentions for their upcoming protocols;
- the evaluation of existing proposals will provide a clear idea on their advantages and security weaknesses;
- the new public key infrastructure that is provably secure and satisfies identified properties;
- the proposed certification model has potential applications – such as applications for secure SSL, for secure email services, and for secure IoT services;
- perhaps our formal verification is the first work on formally verifying protocols in the malicious but cautious model. The verification framework might be applied to verify other protocols which the malicious but cautious adversary is involved.

Above works require formal method (*e.g.* applied pi calculus [14, 60]) to formalise properties and require verification tools (*e.g.* ProVerif [21]) to verify protocols. However, the challenge is how to model the public log and the behaviour of a malicious but cautious party in the verification tool. Also, if the selected tool (*e.g.* ProVerif) is insufficient to implement the verification, then we need to extend it or to find another tool to verify protocols. This will be a big challenge.

Chapter 4

Plan of future work and timetable

This chapter presents a rough research plan for the following two years.

Table 4.1: Research timeline

Time \ Task	Rough Plan of Problem Solving	Goal & Expected Result
Present - 09/2013	Informally evaluate the existing proposals and design a new certification model. (w.r.t. **Work 3.1, 3.2**)	It would be the first part of our work that targets to IEEE S&P 2014.
09/2013 - 11/2013	Formalise and verify our proposed model. (w.r.t. **Work 3.4**)	It would be the second part of our work that targets to IEEE S&P 2014.
11/2013 - 02/2014	Complete the offline verification protocol mentioned at the end of **Work3.2**	This work aims to provide a secure protocol that satisfies offline verification.
11/2013 - 02/2014	Design a protocol that prevents SSL stripping attack by using DNS and public log. (w.r.t. **Work 3.3.1**) (To do this work, we might corporate with other academics from University of Hanover.)	Write down a paper that targets to USENIX Security 2014.
02/2014 - 11/2014	Discover a way to solve encrypted spam email problem. (w.r.t. **Work 3.3.2**)	Try to solve the email spam issue.
02/2014 - 11/2014	Identify and solve problems in IoT data sharing and identify linking. (w.r.t. **Work 3.3.3**)	Try to solve a part of security and privacy issues in IoT.
11/2014 - 05/2015	Finish the work of spam email protection and the work of IoT security and privacy protection. (w.r.t. **Work 3.3.2, Work 3.3.3**)	Two papers target to ACM CCS 2015
05/2015 - 09/2015	Thesis writing.	Write down my Ph.D thesis

Chapter 5

Academic Activities

The academic activities I have been involved so far are listed.

1. 9^{th} ***CryptoForma*** *Meeting*, University of Surrey, November 2012.
2. *Second Workshop on Formal Methods and Tools for Security (**FMATS 2**)*, Microsoft Research, Cambridge, February 2013.
3. *Verifiable Voting Schemes Workshop, from Theory to Practice(**VIVO**)*, University of Luxembourg, Luxembourg, March 2013.
4. *Midlands Graduate School: Mathematical Foundations of Computing Science (**MGS 2013**)*, one week coursework, Leicester, April 2013.
5. 10^{th} ***CryptoForma*** *Meeting*, Microsoft Research, Cambridge, April 2013.
6. *13th International School on Foundations of Security Analysis and Design **(FOSAD 2013)***, University Residential Center of Bertinoro, Italy, September 2013.
7. **Invited reviewer**:
   IEEE Transactions on Information Forensics and Security (**IEEE TIFS**), April 2013.
8. **Sub-reviewer**:
   - *the 18th Australasian Conference on Information Security and Privacy* (**ACISP 2013**);
   - *the 18th European Symposium on Research in Computer Security* (**ESORICS 2013**);
   - *the 8th International DPM Workshop on Data Privacy Management* (**DPM 2013**).

Talks I gave are listed as below:

1. *"Secure Cloud Service"*, Research Skill Module, University of Birmingham, January, 2013.
2. *"A Generic Framework for Three-Factor Authentication"*, Security Seminar [CompSec-Sem], University of Birmingham, Birmingham, March, 2013.
3. *"DTKI: A DNS-Based Protocol for Transparent Key Management"*, Certificate Transparency Hack Day, Google, London, August, 2013.

I will present my work in the following upcoming events:

1. **FOSAD**, Italy, September, 2013.
2. *The 3rd International **CryptoForma** workshop at ESORICS*, London, September 2013.

Chapter 6

Bibliography

[1] Certificate patrol. URL http://patrol.psyced.org. 2.1, 6

[2] The Monkeysphere project. URL http://web.monkeysphere.info. 2.1

[3] ForceTLS and strict-transport-security. URL http://forcetls.sidstamm.com. 2.1.1

[4] Architecting the internet of things. Auto-ID Labs. URL http://www.autoidlabs.org. 2.3

[5] The EFF SSL Observatory. URL https://www.eff.org/observatory. 2.1.2, 2.1

[6] HTTPS Everywhere. EFF Blog, 2011. URL https://www.eff.org/https-everywhere. 2.1.1

[7] NSA slides explain the PRISM data-collection program. The Washington post, June 2013. URL http://www.washingtonpost.com/wp-srv/special/politics/prism-collection-documents/. 1, 2.1.4

[8] PRISM (surveillance program). Wikipedia, 2013. URL http://en.wikipedia.org/wiki/PRISM_(surveillance_program). 1

[9] Research in the wild - internet of things. EPSRC call for proposal, 2013. 2.3

[10] A roadmap for interdisciplinary research in the IoT. IoT Special Interest Group, 2013. 2.3

[11] Internet of things. Wikipedia, 2013. URL http://en.wikipedia.org/wiki/Internet_of_Things. 2.3

[12] The latest news and comment on the US national security agency. The Guardian, 2013. URL http://www.guardian.co.uk/world/nsa. 1, 2.1.4

[13] Judge Napolitano: Can you trust Google and other companies with your private information? YouTube, June 2013. URL http://www.youtube.com/watch?v=SbioAcZhos8. 1

[14] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *POPL*, pages 104–115, 2001. 3.5

[15] Ben Adida, Susan Hohenberger, and Ronald L. Rivest. Lightweight encryption for email. In *Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing*

Unwanted Traffic on the Internet Workshop, SRUTI'05, pages 13–13, Berkeley, CA, USA, 2005. USENIX Association. 2.2

[16] Sattam S. Al-Riyami and Kenneth G. Paterson. Certificateless public key cryptography. In *ASIACRYPT*, pages 452–473, 2003. 2.2

[17] Nadhem J. AlFardan and Kenneth G. Paterson. Lucky thirteen: Breaking the TLS and DTLS record protocols. In *IEEE Symposium on Security and Privacy*, pages 526–540, 2013. 2.1.1

[18] Mansoor Alicherry and Angelos D. Keromytis. Doublecheck: Multi-path verification against man-in-the-middle attacks. In *ISCC*, pages 557–563, 2009. 2.1, 2.1.3

[19] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787–2805, 2010. 2.3

[20] R. Barnes. Use Cases and Requirements for DNS-Based Authentication of Named Entities (DANE). RFC 6394 (Informational), October 2011. URL `http://www.ietf.org/rfc/rfc6394.txt`. 2.1.3

[21] Bruno Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *CSFW*, pages 82–96, 2001. 3.5

[22] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, pages 213–229, 2001. 2.2

[23] Bianca Bosker. Twitter to censor tweets in some countries. The Huffington Post, January 2012. URL `http://www.huffingtonpost.com/2012/01/26/twitter-to-censor-tweets-in-some-countries_n_1235116.html`. 2.1.4

[24] Ran Canetti and Rafail Ostrovsky. Secure computation with honest-looking parties: What if nobody is truly honest? (extended abstract). In *STOC*, pages 255–264, 1999. 2.1.4

[25] Jeremy Clark and Paul C. van Oorschot. SoK: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements. In *IEEE Symposium on Security and Privacy*, pages 511–525, 2013. 2.1.1, 2.1.4, 3.1, 6

[26] 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), May 2008. URL `http://www.ietf.org/rfc/rfc5280.txt`. Updated by RFC 6818. 2.1.2

[27] Alexander W. Dent. A survey of certificateless encryption schemes and security models. *Int. J. Inf. Sec.*, 7(5):349–377, 2008. 2.2

[28] T. Dierks and E. Rescorla. The transport layer security (TLS) protocol version 1.2. RFC 5246 (Proposed Standard), August 2008. URL `http://www.ietf.org/rfc/rfc5246.txt`. Updated by RFCs 5746, 5878, 6176. 2.1, 2.2

[29] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *In Proceedings of the 13th USENIX Security Symposium*, pages 303–320, 2004. 2.1.3

[30] MT Dlamini, MM Eloff, and JHP Eloff. Internet of things: emerging and future scenarios from an information security perspective. In *Southern Africa Telecommunication Networks and Applications Conference*, 2009. 2.3

[31] Hans Dobbertin. Cryptanalysis of MD4. *J. Cryptology*, 11(4):253–271, 1998. 2.1.1

[32] P. Eckersley. Internet-draft: Sovereign key cryptography for internet domains. 2012. 1, 2.1, 2.1.3, 2.1.3, 1

[33] Dan Farber. Can you trust the NSA, the internet giants, or your IT department? Cnet news, June 2013. URL http://news.cnet.com/8301-13578_3-57589180-38/can-you-trust-the-nsa-the-internet-giants-or-your-it-department/. 1

[34] Edward W Felten, Dirk Balfanz, Drew Dean, and Dan S Wallach. Web spoofing: An internet con game. *Software World*, 28(2):6–8, 1997. 2.1.1

[35] Matthew K. Franklin and Moti Yung. Communication complexity of secure computation (extended abstract). In *STOC*, pages 699–710, 1992. 2.1.4

[36] Barton Gellman and Laura Poitras. U.S., British intelligence mining data from nine U.S. internet companies in broad secret program. The Washington post, June 2013. URL http://www.washingtonpost.com/investigations/us-intelligence-mining-data-from-nine-us-internet-companies-in-broad-se 2013/06/06/3a0c0da8-cebf-11e2-8845-d970ccb04497_story.html. 1, 2.1.4

[37] Carmit Hazay and Yehuda Lindell. Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In *TCC*, pages 155–175, 2008. 2.1.4

[38] Nadia Heninger, Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. Mining your Ps and Qs: detection of widespread weak keys in network devices. In *Proceedings of the 21st USENIX conference on Security symposium*, Security'12, pages 35–35, Berkeley, CA, USA, 2012. USENIX Association. URL http://dl.acm.org/citation.cfm?id=2362793.2362828. 2.1.3

[39] J. Hodges, C. Jackson, and A. Barth. HTTP Strict Transport Security (HSTS). RFC 6797 (Proposed Standard), nov 2012. 2.1.1

[40] P. Hoffman and J. Schlyter. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698 (Proposed Standard), August 2012. URL http://www.ietf.org/rfc/rfc6698.txt. 2.1, 2.1.3

[41] J. Kasten, E. Wustrow, and J. A. Halderman. CAge: Taming certicate authorities by inferring restricted scopes. In *Financial Cryptography*, 2013. 2.1, 2.1.3

[42] Tiffany Hyun-Jin Kim, Lin-Shung Huang, Adrian Perrig, Collin Jackson, and Virgil Gligor. Accountable key infrastructure (AKI): A proposal for a public-key validation infrastructure. In *the 22nd International World Wide Web Conference (WWW 2013)*, 2013. 2.1, 2.1.3, 2.1.3

[43] J. Klensin. Simple mail transfer protocol. RFC 5321 (Draft Standard), October 2008. URL http://www.ietf.org/rfc/rfc5321.txt. 2.2

[44] B. Laurie, A. Langley, and E. Kasper. Certificate Transparency. RFC 6962 (Experimental), 2013. 1, 2.1, 2.1.3, 2.1.3, 4

[45] Homin K. Lee, Tal Malkin, and Erich M. Nahum. Cryptographic strength of SSL/TLS servers: current and recent practices. In *Internet Measurement Comference*, pages 83–92, 2007. 2.1.1

[46] Arjen K. Lenstra, James P. Hughes, Maxime Augier, Joppe W. Bos, Thorsten Kleinjung, and Christophe Wachter. Public keys. In *CRYPTO'12*, pages 626–642, 2012. 2.1.1

[47] M. Marlinspike. More tricks for defeating SSL in practice. In *Black Hat, USA*, 2009. 2.1.1

[48] M. Marlinspike. SSL and the future of authenticity. In *Black Hat, USA*, 2011. 2.1

[49] M. Marlinspike and T. Perrin. Internet-draft: Trust assertions for certificate keys (TACK). 2012. 2.1, 2.1.3

[50] Ralph C. Merkle. A digital signature based on a conventional encryption function. In *CRYPTO*, pages 369–378, 1987. 2.1.3, 2

[51] L. O. Murchu N. Falliere and E. Chien. W32.stuxnet dossier. technical report, symantec corporation, 2011. 2.1.2

[52] Vladimir Oleshchuk. Internet of things and privacy preserving technologies. In *IEEE Wireless VITAE*, pages 336–340, 2009. 2.3

[53] Alberto Ornaghi and Marco Valleri. Man in the middle attacks demos. In *Black Hat, USA*, 2009. 2.1.1

[54] Henning Perl, Sascha Fahl, Michael Brenner, and Matthew Smith. A qualitative comparison of SSL validation alternatives. 2013. 3.1

[55] J. Postel. Simple mail transfer protocol. RFC 821 (INTERNET STANDARD), August 1982. URL `http://www.ietf.org/rfc/rfc821.txt`. Obsoleted by RFC 2821. 2.2

[56] B. Ramsdell and S. Turner. Secure/multipurpose internet mail extensions (S/MIME) version 3.2 message specification. RFC 5751 (Proposed Standard), January 2010. URL `http://www.ietf.org/rfc/rfc5751.txt`. 2.2

[57] Eric Rescorla. *SSL and TLS: designing and building secure systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001. ISBN 0-201-61598-3. 2.1

[58] Paul Roberts. Phony SSL certificates issued for Google, Yahoo, Skype, others. Threat Post, March 2011. URL `http://threatpost.com/phony-ssl-certificates-issued-google-yahoo-skype-others-032311`. 1, 2.1.2

[59] Mark Ryan. Certificate issuance and revocation transparency. submitted for publication, 2013. 2.1.3

[60] Mark Ryan and Ben Smyth. The applied pi calculus. In Véronique Cortier and Steve Kremer, editors, *Formal Models and Techniques for Analyzing Security Protocols*. IOS Press, 2011. 3.5

[61] Stuart E. Schechter, Rachna Dhamija, Andy Ozment, and Ian Fischer. The emperor's new

security indicators. In *IEEE Symposium on Security and Privacy*, pages 51–65, 2007. 2.1.1

[62] Seth Schoen and Eva Galperin. Iranian man-in-the-middle attack against Google demonstrates dangerous weakness of certificate authorities. EFF Blog, August 2011. URL https://www.eff.org/deeplinks/2011/08/iranian-man-middle-attack-against-google. 1, 2.1.2, 2.1.4

[63] J. Sermersheim. Lightweight directory access protocol (LDAP): The protocol. RFC 4511 (Proposed Standard), June 2006. URL http://www.ietf.org/rfc/rfc4511.txt. 2.2

[64] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984. 2.2

[65] D. Shaw. The Camellia Cipher in OpenPGP. RFC 5581 (Informational), June 2009. URL http://www.ietf.org/rfc/rfc5581.txt. 2.2

[66] Ray Shaw. Can you trust Google any more (or any less)? ITWire, April 2013. URL http://www.itwire.com/opinion-and-analysis/shaw-thing/59425-can-you-trust-google-any-more-or-any-less. 1

[67] Christopher Soghoian and Sid Stamm. Certified lies: Detecting and defeating government interception attacks against SSL. In *Financial Cryptography*, pages 250–259, 2011. 1, 2.1, 2.1.3, 2.1.4

[68] Christopher Soghoian and Sid Stamm. Certified lies: Detecting and defeating government interception attacks against SSL (short paper). In *Financial Cryptography*, pages 250–259, 2011. 2.1.4

[69] Bruce Sterling. Disruptive civil technologies – six technologies with potential impacts on US interests out to 2025. National Intelligence Council Report, 2008. 2.3

[70] Bruce Sterling. More about cybersecurity for the internet of things, 2013. URL http://www.wired.com/beyond_the_beyond/2013/07/more-about-cybersecurity-for-the-internet-of-things/. 2.3

[71] Toby Sterling. Second firm warns of concern after dutch hack. Yahoo! News, September 2011. URL http://news.yahoo.com/second-firm-warns-concern-dutch-hack-215940770.html. 1, 2.1.2, 2.1.4

[72] Marc Stevens, Arjen K. Lenstra, and Benne de Weger. Chosen-prefix collisions for MD5 and colliding X.509 certificates for different identities. In *EUROCRYPT*, pages 1–22, 2007. 2.1.1

[73] Marc Stevens, Alexander Sotirov, Jacob Appelbaum, Arjen K. Lenstra, David Molnar, Dag Arne Osvik, and Benne de Weger. Short chosen-prefix collisions for md5 and the creation of a rogue ca certificate. In *CRYPTO*, pages 55–69, 2009. 2.1.1

[74] Joshua Sunshine, Serge Egelman, Hazim Almuhimedi, Neha Atri, and Lorrie Faith Cranor. Crying wolf: An empirical study of SSL warning effectiveness. In *USENIX Security Symposium*, pages 399–416, 2009. 2.1.1

[75] S. Turner and T. Polk. Prohibiting secure sockets layer (SSL) version 2.0. RFC 6176

(Proposed Standard), March 2011. URL http://www.ietf.org/rfc/rfc6176.txt. 2.1

[76] Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In *EUROCRYPT*, pages 19–35, 2005. 2.1.1

[77] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In *CRYPTO*, pages 17–36, 2005. 2.1.1

[78] Rolf H Weber. Internet of things–new security and privacy challenges. *Computer Law & Security Review*, 26(1):23–30, 2010. 2.3

[79] S. Weiler and D. Blacka. Clarifications and Implementation Notes for DNS Security (DNSSEC). RFC 6840 (Proposed Standard), February 2013. URL http://www.ietf.org/rfc/rfc6840.txt. 2.1.3

[80] Dan Wendlandt, David G. Andersen, and Adrian Perrig. *Perspectives:* improving SSH-style host authentication with multi-path probing. In *USENIX Annual Technical Conference*, pages 321–334, 2008. 2.1, 2.1.3

[81] Tara Whalen and Kori M. Inkpen. Gathering evidence: use of visual security cues in web browsers. In *Graphics Interface*, pages 137–144, 2005. 2.1.1