2006-11-11

Digital Receipts: A System to Detect the Compromise of Digital Certificates

Nathaniel Allen Seeley
Brigham Young University - Provo

BYU ScholarsArchive Citation

Seeley, Nathaniel Allen, "Digital Receipts: A System to Detect the Compromise of Digital Certificates" (2006). *Theses and Dissertations*. 1108.
https://scholarsarchive.byu.edu/etd/1108

DIGITAL RECEIPTS:
A SYSTEM TO DETECT THE COMPROMISE
OF DIGITAL CERTIFICATES

by

Nathaniel Seeley

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Brigham Young University

December 2006

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Nathaniel Seeley

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date Kent E. Seamons, Chair

Date Mark J. Clement

Date Eric G. Mercer

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Nathaniel Seeley in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

Kent E. Seamons
Chair, Graduate Committee

Accepted for the Department

Parris K. Egbert
Graduate Coordinator

Accepted for the College

Thomas W. Sederberg
Associate Dean, College of Physical and Mathematical Sciences

ABSTRACT

DIGITAL RECEIPTS:
A SYSTEM TO DETECT THE COMPROMISE
OF DIGITAL CERTIFICATES

Nathaniel Seeley

Department of Computer Science

Master of Science

The ease of copying digital materials creates difficulty in detecting the theft of digital certificates. Uneducated users frequently fail to protect their digital certificate keys by not encrypting them, storing them in insecure places, and using them unwisely. In addition, there is no way to prove that protocols involving certificates are completely secure. This thesis introduces a system to ameliorate these problems by detecting the compromise of digital certificates. It leverages dual logging messages sent via side channels to a trusted third party. This third party correlates these messages and automatically detects when an imposter presents a certificate based on the collected evidence.

ACKNOWLEDGMENTS

I would like to thank my family and friends for their support and encouragement and God for giving me the opportunity to make this work possible.

Table of Contents

List of Tables

List of Figures

Chapter 1 — Introduction

Knowledge of a secret, like a password, is a common form of authentication. Unlike driver's licenses and other physical credentials carried in one's wallet, a secret can be stolen and still remain in the owner's possession. Digital certificates, which are analogous to physical credentials, can be presented as evidence that the owner possesses certain attributes. Digital certificates are in use in systems like Transport Layer Security [7] and trust negotiation [18]. Proof of ownership of a digital certificate requires an associated private key. Since ownership of digital certificates involves secrets, it is difficult to detect when a digital certificate is stolen.

For that reason, this thesis presents an auditing system to detect when stolen digital certificates are used. This system utilizes dual logging via messages sent down side channels in order to detect compromise'. When an owner demonstrates knowledge of a secret to an entity (hereafter called a *verifier*), side channel messages are sent to an entity called the *Receipt Resolution Server* (RRS). The RRS analyzes these messages for discrepancies and notifies owners if suspicious activity is discovered. Although this thesis was designed specifically to protect the secrets associated with digital certificates, it could be adapted to protect other types of secrets as well.

1.1 Digital Certificate Theft

Detecting the theft of a driver's license or student ID is straightforward. If it is not in the owner's wallet, or if the wallet is missing, then there is a chance the credential has been compromised.

Detecting the theft of a credit card or credit card number is slightly different than detecting the theft of a driver's license, because using a credit card does not always involve presentation of the physical card. Ownership is assumed when one demonstrates knowledge of the number on the card, as is the case when one shops online or makes purchases over the phone. Thus, it is more difficult to detect the theft of a credit card number, since the number is a sequence of digits, not a physical object. This is one reason why cardholders receive itemized monthly statements and why credit card companies have fraud detection departments. If there are charges on the monthly statement that the credit card owner did not make there is a chance that someone has stolen the number.

As with a credit card, ownership of most digital certificates involves knowing a sequence of numbers. For example, an X.509 certificate contains a public key that corresponds to a private key which cannot be feasibly derived from the public key. Proving ownership of this type of digital certificate involves proving knowledge of the associated private key. Therefore, in order to avoid theft of an X.509 certificate one must maintain the secrecy of the private key. The system developed in this thesis focuses on protecting public/private key certificates.

There are two ways to steal these types of digital certificates. The first is learning the private key. The second is more complicated, involving the manipulation of a vulnerable certificate ownership protocol. In this second method, an attacker tricks the certificate owner into performing an action that will allow the thief to feign knowledge of the private key. For an example of a broken certificate ownership protocol, see Section 2.2.

Compromise of a physical or digital certificate can lead to identity theft, a growing problem in the United States today. Identity theft occurs when someone steals enough personal information about a victim to impersonate them in some context. The FTC reports that identity theft is escalating, with incidents in over 246,000 households in the United States in 2004 [1]. Internet fraud losses totaled over

$265,000,000 for that same year. As the use of digital certificates increases, it is probable that identity theft with digital certificates will also increase. This thesis research explores the problem of digital identity theft and presents a system to detect the compromise of a digital certificate.

1.2 Private Key Compromise

Frequently, digital certificate private keys are exposed through the actions of uneducated or careless owners, resulting in the potential for theft. Whitten and Tygar claim that the vast majority of computer security failures come from user error [19]. Peter Gutmann, a security expert at the University of Auckland, discusses specific ways in which user error makes the private keys of digital certificates vulnerable. Gutmann, who helped author the internationally recognized PGP and Cryptlib products, is a respected security researcher who has broken various noteworthy encryption systems, including the DES encryption used in Norton's Diskreet and the password file encryption used in Windows 3.1 and Windows 95.

Gutmann [9] claims that for practicality and usability reasons, system administrators often unwisely spread their private keys around numerous machines and applications on the network, making them more likely to be stolen. For instance, most system administrators make backup tapes of company data. These tapes often include the company private key, since it would be costly for the key to be lost. The backup tapes are often improperly secured, presenting opportunities for malicious individuals to steal private keys. Gutmann gives an example of a company whose backup tapes (containing a private key) were extensively scattered to various employees and assorted locations. These tapes were frequently transported by insecure methods, such as the back seat of a car. Gutmann jokes, "the only way to securely delete the [private] encryption key being used to protect large amounts of long-term sensitive data would have been to carpet-bomb the city."

In addition to the problems caused by carelessness, curiosity can also cause unnecessary exposure of certificate private keys. For example, many applications allow the private key to be exported as plaintext, rather than in a format that requires a password to decrypt. When asking why anyone would want to make their private key so vulnerable, Gutmann receives bizarre responses such as, "I don't know, I just want to do it."

At other times, the private key is made vulnerable when users misunderstand how certificates work. For example, one company distributed the private key of the company's root certificate to various company computers. They made the outrageous mistake of thinking that doing so would alleviate the error messages in web browsers and mail programs that the company's certificate was not trusted.

Although uneducated users have a higher risk of having their private keys compromised, even shrewd users and security minded companies are not completely safe. Attackers are inventive and unpredictable; they may even be insiders in the company whose certificates they attempt to steal. In situations like these, a method of detecting digital certificate theft is highly desirable. The digital receipts system is designed to detect compromise by noticing when an attacker attempts to use a certificate without the owner's knowledge.

1.3 Proving Protocol Security

An attacker does not necessarily need to know a digital certificate's private key to pretend ownership. Attackers can trick legitimate owners into disclosing material that the attacker can later present as proof of ownership. This is accomplished by manipulating insecure certificate ownership protocols. Designing a secure certificate ownership protocol is incredibly hard. Whitfield Diffie writes, "The design of cryptographic protocols in general, and authentication protocols in particular, is extremely error prone" [8]. Ross Anderson compares writing such protocols to programming Satan's computer, and claims that even very short protocols have turned out to contain security flaws which were not discovered for over a decade [3]. No one has yet been successful in finding an airtight method for proving that a digital certificate ownership protocol cannot be broken.

Some researchers have attempted to prove the security of these protocols by using formal methods. These approaches fall into four categories [14]. The first models a protocol using tools not specifically engineered for the analysis of protocols. The second method utilizes expert systems to analyze different scenarios and draw conclusions about the protocol's security. The third uses logics developed specifically for the analysis of knowledge and belief. Burrows, Abadi, and Needham pioneered this approach with *BAN* logic [6]. BAN logic assumes authentication is a function of integrity and freshness and uses logic to evaluate both of these attributes throughout the protocol. However, BAN logic cannot prove a protocol is secure; it can only present an organized way to reason about authentication and detect flaws. The fourth method models the protocols in an algebraic system and analyzes the reachability of certain states. Of these four approaches, the most popular is the third.

However, even specifically designed logics cannot prove security. According to Anderson, "quite a few protocols which had been 'proved' secure have been successfully attacked" [3]. Additionally, Rubin and Honeyman claim [13] that:

> ...there is no technique known for proving that a protocol is secure. The reason for this may be that security itself is not sufficiently well defined. We can prove that a protocol is correct, or that it meets its

> specification. We can even prove that under various assumptions, certain attacks against a protocol will not work. However, we have no general purpose method of proving that an arbitrary authentication protocol is secure.

The nearly impossible task of proving protocol security makes a mechanism that alerts users when someone else has presented and proved ownership of their certificate immensely valuable.

1.4 Thesis Statement

The digital receipts auditing system can detect the usage of compromised digital certificates, is secure against a wide variety of attacks, and has reasonable network overhead.

1.5 Thesis Outline

Chapter 2 gives an overview of the system. Chapter 3 explains the details of the different protocols used to set up and send side channel messages. Chapter 4 discusses the internals of the Receipt Resolution Server, the trusted third party necessary for the operation of the system. Chapter 5 analyzes the system with respect to known methods of attack. Chapter 6 explains the major design decisions of the implementation and Chapter 7 analyzes how the system might affect a real network. Finally, Chapter 8 contains conclusions and future work.

Chapter 2 — Overview

This chapter presents an overview of the digital receipts system, including what parties are involved, what messages are communicated, and how the system detects theft. The architecture of the digital receipts system is shown in Figure 2.1. The principal parties are the certificate owner, the verifier, and the Receipt Resolution Server (RRS).

To illustrate how the system works, consider the following hypothetical example. Suppose Bob works for MedSoft, a company specializing in software used by hospitals. MedSoft works closely with several partner companies, including LifeTech. MedSoft allows developers with certificates signed by a partner company to view source code. Alice, a LifeTech employee, wishes to access source code on the software repository administered by Bob. In message 1 in Figure 2.1, Alice presents her LifeTech certificate and a proof of ownership to Bob in order to gain access. In this example Alice is the certificate owner and Bob is the verifier.

In message 2, Bob sends a *receipt* to the RRS, indicating that Alice has proven ownership of her certificate to him. Receipts are simply records of transactions, analogous to paper receipts from a grocery store.

Message 3 is a *proof record* sent from Alice to the RRS that contains the specifics of the transaction. A proof record is analogous to a reverse monthly credit card statement in that it contains a record of all of the transactions involving that certificate during a particular time period. The difference is that instead of the credit card company sending a statement to the certificate owner, a proof record is sent from the certificate owner to the company. The RRS requires Alice to authenticate herself using a private key of a certificate registered with the RRS before accepting the

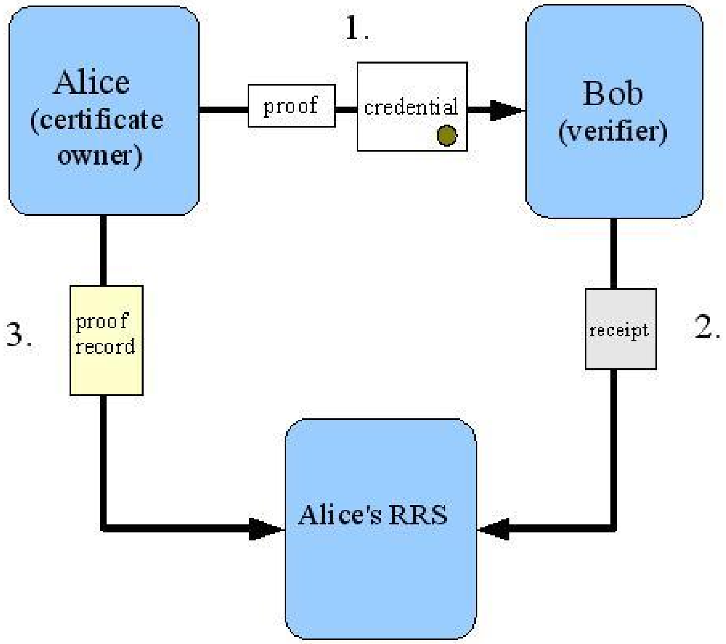


Figure 2.1: Digital receipts system architecture

proof record. Requiring Alice to authenticate helps mitigate forged proof records, as explained in section 5.1.3.

Receipts and proof records are sent to the RRS, which compares the transaction entries in the proof record with the receipts. These messages are encrypted to protect the privacy of the certificate owner. If there are receipts detailing transactions not on the proof record, the RRS alerts Alice that someone may have stolen her certificate.

We assume that an attacker may have stolen any secret off of Alice's machine, but the RRS is secure. Therefore, the RRS machine should be different than the one Alice uses her certificates on. This also frees Alice from the need for a machine that is always on and always connected to the network to collect the receipts, which could be sent at any time. In addition, it provides added security since an attacker has to steal secrets from both Alice and the RRS, rather than just Alice, in order to compromise the system.

2.1 Digital Certificate Theft Example

Suppose that someone steals Alice's private key and proves ownership of her certificate to Carol. When the RRS receives a receipt from Carol it will not find a corresponding entry for that transaction on Alice's proof record. The RRS then informs Alice there is a possibility that her certificate has been stolen.

2.2 Insecure Authentication Example

The digital receipts system also detects when a certificate ownership protocol has been broken. Suppose Alice proves ownership of her certificate to Mallory using the simple, yet insecure challenge/response protocol shown in Figure 2.2. In this protocol, Mallory sends a random number to Alice. Alice sends a message back to Mallory containing this number encrypted with the private key corresponding to the certificate she is presenting. Mallory verifies Alice's certificate by decrypting this message with the public key embedded in the certificate and making sure it matches the original random number.

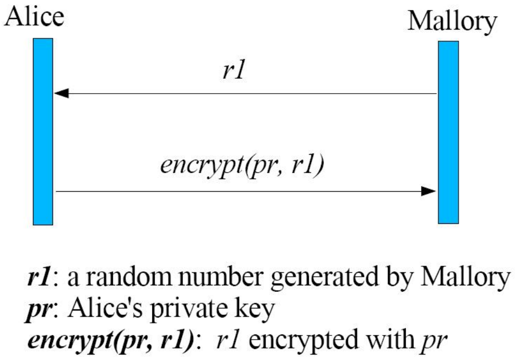

Alice
Mallory
r1
encrypt(pr, r1)
r1: a random number generated by Mallory
pr: Alice's private key
encrypt(pr, r1): r1 encrypted with pr

Figure 2.2: An insecure certificate ownership protocol

This protocol allows Mallory to masquerade as Alice. Mallory first entices Alice

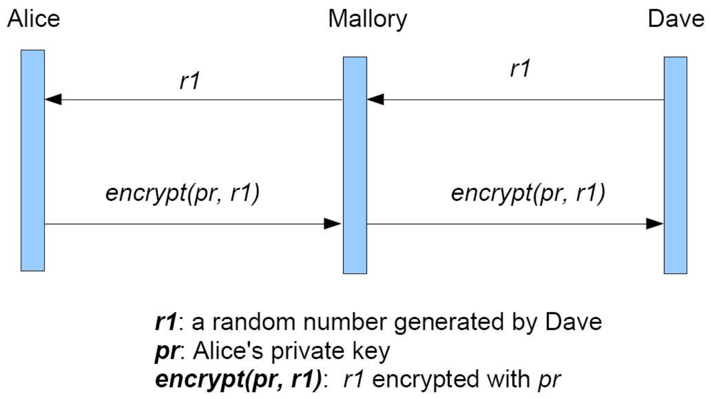

Alice
Mallory
Dave
r1
r1
encrypt(pr, r1)
encrypt(pr, r1)
r1: a random number generated by Dave
pr: Alice's private key
encrypt(pr, r1): *r1* encrypted with *pr*

Figure 2.3: Mallory exploiting a flawed protocol to impersonate Alice to Dave

to connect and send a certificate. Mallory then presents Alice's certificate to Dave as her own. Dave sends Mallory a random number, which Mallory forwards to Alice. Alice, seeking to prove ownership of her certificate to Mallory, then encrypts this number with her private key and sends this to Mallory. Mallory then forwards Alice's encryption to Dave, making Dave believe Mallory owns Alice's certificate.

The fraud is detected when the RRS receives Dave's receipt, and Dave's transaction does not appear on Alice's proof record.

2.3 Assumptions

The digital receipts system makes five assumptions about the resources and capabilities of an attacker. The intent of these assumptions is to not underestimate Mallory so that the design will defend against probable attackers.

1. Mallory has complete control of the network between the entities in the system. She is able to insert, delete, and modify network packets at will.

2. If Mallory has successfully stolen a certificate, she is assumed to have stolen any other secret stored on Alice's machine. Thus, digital receipts cannot rely

on any secret stored on Alice's machine.

3. Mallory does not have complete control over Alice's machine. If Mallory has complete control, she can influence everything that Alice sees on her computer screen and record all input, including passwords. Mallory would be able to modify or substitute any protection software that Alice might run. No system could provide security under such circumstances.

4. The RRS is not compromised. Clearly if the RRS software was compromised, or if a malicious insider worked for the company administering the RRS, the attacker could prevent the RRS from warning the certificate owner about potential certificate thefts. This also means that the RRS certificate has not been stolen

5. The proper usage of TLS provides confidentiality, message integrity, and end-point authentication. TLS authentication ensures that a party knows the private key of the certificate it presents during the initial TLS handshake. Both server and client/server TLS authentication are assumed to be secure. Although there have occasionally been vulnerabilities found in implementations of TLS, major implementations like OpenSSL are patched quickly. Furthermore, TLS is the security protocol used by large banks and other financial institutions and is a trusted protocol for protecting sensitive customer data.

Chapter 3 — Protocol Design

There are three communication protocols in the digital receipts system. The first permits Alice to send proof records to the RRS containing information recorded by Alice about each transaction during a given time window. The second protocol permits a verifier to send receipts, each of which contains information about a single transaction as recorded by the verifier. The Receipt Resolution Server (RRS) correlates the proof records with the receipts and identifies discrepancies that imply a compromised certificate. The third protocol allows Alice to register her machine to the RRS to send proof records at preestablished times.

3.1 Proof Record Protocol

Each machine Alice uses the certificate on sends regularly scheduled proof records to the RRS containing information about all certificate transactions during the reporting period. For example, Alice might send the RRS weekly reports detailing all certificate transactions originating from her laptop.

3.1.1 Requirements

A proof record contains the information necessary for the RRS to correlate each transaction with a receipt. It must be sent with confidentiality and integrity and the RRS must be able to detect forged proof records. The certificate owner must be guaranteed that the proof record reached the RRS intact.

3.1.2 Sending the Proof Record

Given that the method for sending a proof record requires both confidentiality and integrity, there are two alternatives for protecting the transmission: TLS and a custom protocol that utilizes the RRS public/private key.

3.1.2.1 TLS

The first method for transmitting the proof record is for Alice's machine to contact the RRS over TLS with the RRS acting as the server and with client authentication. This uses the time-tested security of TLS to authenticate the RRS, encrypt the proof record, and verify the integrity of the messages being sent. The RRS responds with a hash of the proof record so Alice can be sure the RRS actually received it. The client authentication obviously does not prevent an attacker who has stolen the private key from authenticating. The client authentication is to prevent the RRS from receiving a lot of false proof records from people who neither own nor have stolen certificates. Section 5.1.3 explains how false proof records from attackers who have successfully stolen the certificate private key are discovered.

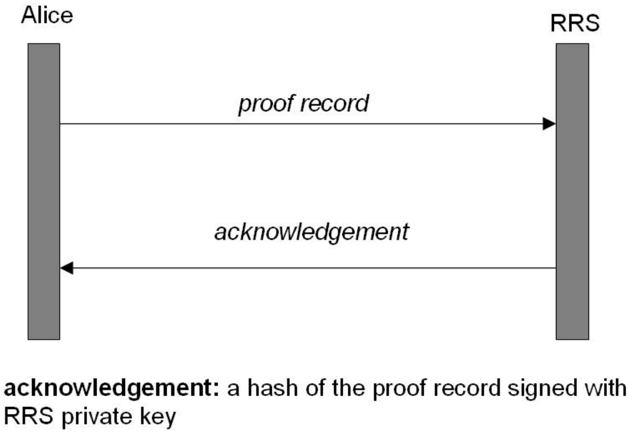

Alice
RRS
proof record
acknowledgement
acknowledgement: a hash of the proof record signed with RRS private key

Figure 3.1: Sending proof records over TLS

One method for authenticating Alice to the RRS would have been to have Alice enter a password. The advantage of this approach is that an attacker who has stolen Alice's certificate might not know Alice's password and thus not be able to forge a proof record. The primary disadvantage is that either Alice must be present each time a proof record is sent or the password must be stored on Alice's machine. Alice would be inconvenienced if she had to enter a password for every sent proof record and might disable the proof records system. If the password were stored on Alice's machine, there is a chance an attacker could steal it, especially if the attacker has already managed to steal one of Alice's private keys. Therefore, we choose not to consider password authentication as a option when sending proof records.

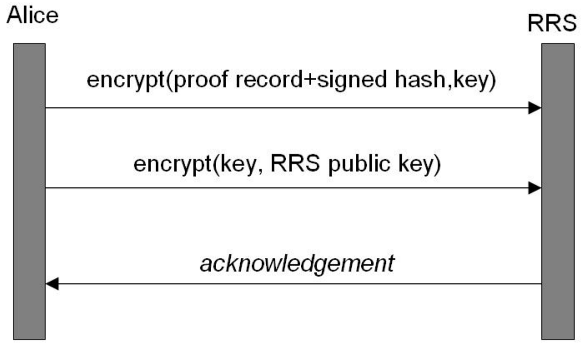
3.1.2.2 With Encryption by RRS Public Key

In this method, the certificate owner signs a hash of the proof record with his private key and encrypts the proof record and hash with a symmetric encryption key. This symmetric encryption key is then encrypted with the public key of the RRS. The symmetric key and the encrypted message are then sent to the RRS. The RRS responds with a hash of the proof record signed with its private key. This second method is more efficient in that it does not require the parties to go through the processor intensive TLS handshake protocol. For this reason it is used in the implementation described in Chapter 6. However, it lacks the tested security of TLS. This method is summarized in Figure 3.2.

3.1.3 Format of a Proof Record

Each proof record has a header that identifies the corresponding certificate. It contains the fields found in Table 3.1. If two proof records arrive from the same machine claiming to cover the same time period, the RRS alerts the owner that one of them may be forged.

A proof record contains an entry for each transaction in which the certificate


Alice
RRS
encrypt(proof record+signed hash,key)
encrypt(key, RRS public key)
acknowledgement

signed hash: a hash of the proof record signed with the private key of the certificate
key: a symmetric encryption key used to encrypt the proof record and signed hash
RRS public key: the public key of the RRS, used to encrypt the symmetric key
acknowledgement: a hash of the proof record signed with the RRS private key

Figure 3.2: Sending proof records with encryption by RRS public key

was used during the time interval. Each transaction entry contains the fields shown in Table 3.2.

Field	**Description**
Hash of Certificate	The hash of the certificate with which the transactions were conducted
Machine Identifier	A unique identifier assigned by the RRS during the registration of this machine (See Section 3.3)
Activity Period	The time interval for this proof record
Padding (optional)	Obfuscates the size of proof records to prevent attackers from inferring certificate activity

Table 3.1: Proof record header

Field	**Description**
Signature Material	The data Alice signed with her private key to prove ownership
Timestamp	The time Alice recorded for this transaction
IP Address of Verifier	Bob's IP Address

Table 3.2: Fields in a transaction entry

3.2 Receipt Protocol

Each time Bob receives a certificate and a certificate ownership proof, he sends a receipt containing information about the transaction to the certificate owner's RRS. One motivation to do this might be that customers refuse to patronize sites that do not send receipts. When receiving a certificate, Bob should check to see if the certificate has been revoked by the authority that issued it before continuing the transaction. Bob can be efficient by sending the receipt while making this check (see Section 4.5). In this case, the receipt would be sent before the transaction with Alice could continue.

In the real world, most verifiers do not check certification revocation lists and the RRS may be unavailable during some transactions. Therefore, it is not a requirement that a transaction be blocked until the receipt is sent. The verifier can send the receipt asynchronously, or after the certificate transaction has ended. This causes the certificate transaction to complete faster. However, it is slightly less efficient for the RRS to match proof records and receipts. This is because asynchronous receipts are not guaranteed to be in the RRS database when the proof record arrives. Therefore, the RRS has to perform matches both when the proof record arrives and when each late receipt arrives.

3.2.1 Requirements

A receipt must contain the information necessary to allow the RRS to correlate each transaction with a proof record. It must be sent in such a way as to provide confidentiality and integrity and the RRS must be able to mitigate false alarms caused by fraudulent receipts. The verifier must also be guaranteed that the receipt reached the RRS.

Field	**Description**
When to send receipts	Under what circumstances should a receipt be sent
URL and port of RRS	Where to send the receipt
TLS required? (T or F)	Whether the receipts should be sent over TLS or encrypted with the RRS public key
Hash of RRS public key	Allows the verifier to authenticate the RRS

Table 3.3: Extensions to a certificate necessary to enable the sending of receipts

3.2.2 Certificate Extensions

Bob must be given the information on where to send the receipt in a secure way. Otherwise an attacker could cause a verifier to send receipts that never arrive at the RRS. For example, say Mallory has stolen Alice's certificate. Mallory presents the certificate and a proof of ownership to Bob. Bob needs to know where to send the receipt. Mallory provides him the address of a bogus RRS, and Bob's receipt never reaches Alice's RRS. Alice is never warned that her certificate is stolen

To avoid this problem, the RRS information is embedded in the certificate. This certificate extension contains the fields found in Table 3.3.

3.2.3 Sending the Receipt

A receipt contains the fields shown in Table 3.4. Future research may reveal that additional fields, such as the business name or service being authenticated to, could help the RRS better determine patterns that could lead to the capture of an attacker. For example, if the RRS noticed that the certificates were only presented

Field	**Description**
Hash of Certificate	Which certificate this receipt corresponds to.
Signature Material	What the certificate owner signed to prove ownership of the certificate.
Signature	The private key signature of the signature material.
Timestamp	The time that the certificate owner recorded for this transaction.
Certificate Owner Machine ID	The RRS assigned unique ID of the certificate owner's machine sending the proof record.
Certificate Owner IP Address	The IP address of the certificate owner.

Table 3.4: Fields found in a receipt

to gain access to open source companies that specialize in medical software, the RRS might be able to use that information to determine clues about the attacker's identity.

The verifier can send the receipt in one of two ways, over TLS or using the RRS public key to encrypt the receipt.

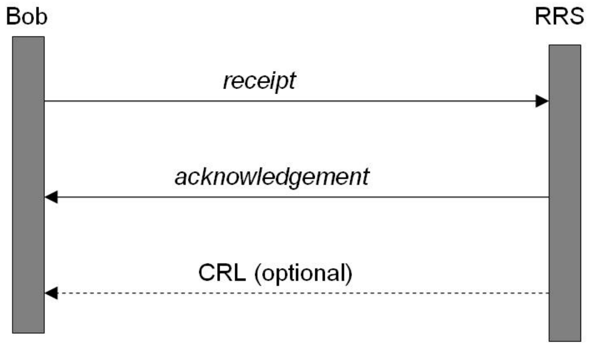
3.2.3.1 TLS

Bob, the verifier, can open a TLS connection with client authentication to the RRS URL and port listed in the certificate with the RRS acting as server and using client authentication. Bob then

- sends the receipt

- waits for an acknowledgement from the RRS, which is a hash of the receipt signed by the RRS private key
- waits for the RRS to send a CRL (optional, see Section 4.5 for more details)

Because it relies on the strong security of TLS, this is the recommended method. This method is summarized in Figure 3.3.



acknowledgement: a hash of the receipt signed with the RRS private key
CRL: a certificate revocation list from the certificate's issuing authority

Figure 3.3: Sending receipts over TLS

3.2.3.2 Encryption by RRS Public Key

The overhead in setting up a TLS session is considerable. Therefore, a faster, yet less proven method involves using the RRS public key to encrypt the receipt. For efficiency, this is the method used in the implementation described in Chapter 6. In this method, Bob opens a connection the the RRS URL and port listed in the certificate. Bob then

- sends his own certificate
- sends the receipt concatenated with a hash of the receipt signed with the public key of Bob's certificate. This data is encrypted with a symmetric key.
- sends the symmetric key, encrypted by the RRS public key
- waits for the RRS to send an acknowledgement, which is a hash of the receipt concatenated with the timestamp. This acknowledgement is signed by the RRS private key.
- waits for the RRS to send a CRL (optional, see Section 4.5 for more details)

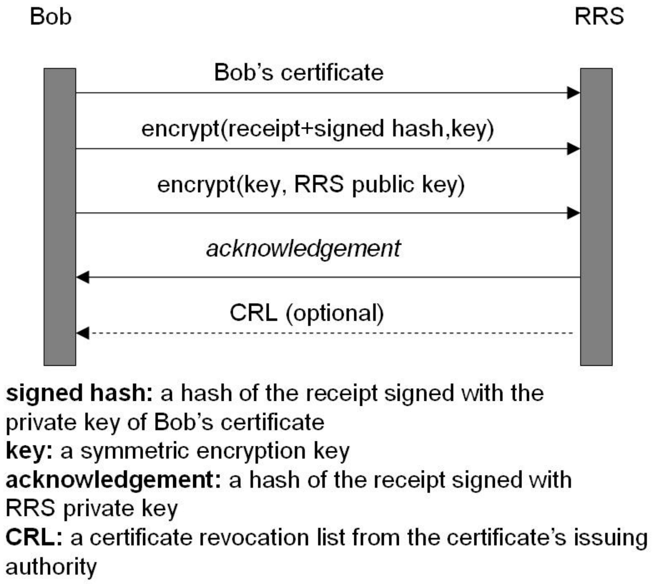

Bob
RRS
Bob's certificate
encrypt(receipt+signed hash,key)
encrypt(key, RRS public key)
acknowledgement
CRL (optional)
signed hash: a hash of the receipt signed with the private key of Bob's certificate
key: a symmetric encryption key
acknowledgement: a hash of the receipt signed with RRS private key
CRL: a certificate revocation list from the certificate's issuing authority

Figure 3.4: Sending receipts with encryption by RRS public key

There are different possibilities for allowing Bob to obtain the RRS public key. One method is to include the domain of the RRS on the certificate and trust that only the RRS has a certificate for that domain, as is done in setting up TLS connections with banks and credit card companies. However, if an attacker managed to obtain a valid certificate for the RRS domain, the attacker could masquerade as the RRS. A more secure method is to embed a hash of the public key of the RRS into the certificate whose ownership is being proven, as shown in Table 3.3. The implementation described in Chapter 6 uses certificates with an embedded RRS public key hash.

3.3 Registration

The RRS must ensure that each proof record received actually came from the certificate owner. A naive approach for verifying the authenticity of the proof records would be to rely on a secret known only to the RRS and Alice. If an attacker like Mallory is able to gain one secret from the user, like a certificate private key, it is highly probable that she can obtain others. Therefore it is unwise for the RRS to rely solely on a shared secret with the user to authenticate the proof records.

To mitigate this problem, each certificate owner's machine sends the RRS a schedule of when to expect proof records. If Mallory forges a proof record, it will conflict with the authentic proof record when it arrives. Due to the server authentication property of TLS, the certificate owner can verify that the entity he is sending the proof record to is, in fact, the RRS. Therefore, the certificate owner can be sure no forged proof records have been accepted for a particular time period after the genuine proof records for that period are acknowledged for each machine.

3.3.1 Requirements

The RRS must be informed when and how often each certificate owner's machine will send proof records and what time intervals these records will cover. The certificate owner must be assured that the RRS has received that knowledge.

3.3.2 Establishing a Business Relationship

Before registering certificates, Alice must establish a business relationship with the RRS. To do this, she submits her name, contact information, and method of payment. If the RRS is to act as Alice's *certificate authority* (CA), the RRS will generate certificates for her. Alice may then register these certificates. If the RRS is not a CA, Alice sends her certificates to the RRS. Alice then selects a passphrase for online access to her account to examine registrations, warnings, and certificate activity. However, because an attacker may be able to read secrets on Alice's machine and thus obtain her passphrase, one cannot cancel registrations, suppress warnings, or make permanent changes from the online account. Alice should not store her passphrase on any of her machines in order to reduce the chance of theft. Even if the passphrase is stolen, the attacker will not be able to make any changes that might affect the system's ability to detect the usage of stolen certificates.

3.3.3 Registering a Machine

Each machine that uses the certificate is registered with the RRS in advance and scheduled to make regular interval proof record reports. The RRS notifies Alice of each machine registration. The machine to be registered opens a TLS connection with the RRS with client authentication and sends a message containing the fields shown in Table 3.5. The RRS responds with an acknowledgement message that contains a hash of the registration message signed with the RRS private key and a unique machine identifier. These messages are summarized in Figure 3.5. The RRS then sends a message to Alice via a different channel (like a phone call) informing her of the registration. In addition, the list of registered machines is available to Alice at any time. If Alice sees a registration acknowledgement message for a machine that she never registered, she should be wary that someone has stolen her certificate.

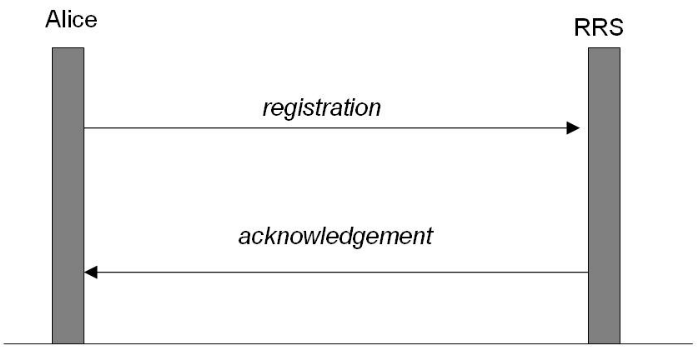
The RRS waits a certain, owner defined time period after the end of the reporting period for a particular proof record. If a certificate owner's machine fails to send a proof record by that time, the RRS notifies the certificate owner.

Field	**Description**
Hash of Certificate	Used to identify the certificate
Start Date	When to expect the first proof record
Interval	How often to expect proof records

Table 3.5: Machine registration data

3.3.4 Canceling a Registration

To cancel a machine registration, Alice sends a request to the RRS. The RRS then contacts Alice out-of-band for confirmation. Cancellations are also displayed in her online account. If a machine with a cancelled registration continues to send proof records, the RRS notifies Alice, who should then be concerned that the her communication channel with the RRS has been compromised.



acknowledgement: a hash of the registration signed with
The RRS private key

Figure 3.5: The registration protocol over TLS

Chapter 4 — The Receipt Resolution Server

This chapter discusses how the Receipt Resolution Server (RRS) analyzes the proof records and receipts for unusual activity. At a high level, the RRS stores the receipts from Bob until the associated proof record arrives from Alice. Then, for each transaction in the proof record, the RRS finds the matching receipt. The verifier public key listed in the proof record helps the RRS to determine that the receipt came from the verifier that Alice interacted with at the specified time. If a receipt does not correspond to any proof record transaction, the RRS alerts Alice of possible theft.

4.1 Requirements

The RRS must acknowledge receipt of all proof records, receipts, and registrations and ensure the confidentiality and integrity of these messages. It must also be able to match proof records and receipts.

4.2 Matching Proof Records and Receipts

The RRS attempts to match each transaction in a proof record with a receipt. It assumes that the same machine will not present multiple copies of a certificate with the same signature material to the same verifier at the same time. Therefore, the RRS searches for a receipt that has the same certificate hash, machine id, timestamp, verifier public key, and signature material as the proof record transaction.

The inclusion of timestamps in a protocol is discouraged by Abadi and Needham [2] due to the difficulty of synchronizing clocks on different machine. For this reason, Bob, the verifier, obtains from Alice the timestamp Alice recorded for the transaction and includes this timestamp in the receipt. If Mallory successfully steals Alice's certificate, she might set the timestamp to be months in the future. The RRS would not realize the receipt was unmatched until the proof record for that time period came in, causing the theft to be undetected for months. Therefore, the RRS will do a sanity check on the timestamps of incoming receipts. Any receipt whose timestamp is more than several days off will be flagged as suspicious and the certificate owner will be notified.

4.3 Alarm Side Channels

Any unmatched receipts or duplicate, conflicting proof records cause a warning to be sent to Alice over a side channel. The most secure side channel is a courier sent to the physical address of Alice. Unfortunately, sending a courier is very expensive. The RRS may also opt to telephone Alice or send a letter as a potentially less secure but more affordable channel. The problem with these methods is that an attacker might be able to intercept the warning message by hijacking the phone line or removing the letter. For this reason, all warnings are available to the certificate owner upon request by periodically connecting to the RRS via HTTPS to view warnings, machine registration data, and other information.

4.4 Investigation

If an alarm is raised, the RRS can investigate to see if more information can be gained about the attack. It can first examine the IP address of the unmatched receipt and attempt to gain further information about where the stolen certificate was used. It can also search other transactions on the proof record to see whether the unmatched signature material was used in any legitimate transactions in a nearby time window. If so, there is a possibility that an ownership protocol was breached and that the verifier in the proof record transaction should not be trusted. Future research is needed to identify additional data and methods of forensic analysis.

4.5 Certificate Revocation Lists

When a verifier receives a certificate, he or she should consult the certificate authority (CA) that issued the certificate to see if it has been revoked. This is possible by checking a *certificate revocation list* (CRL), a list of serial numbers of revoked certificates. According to RFC 3280 [11]:

> . . . various circumstances may cause a certificate to become invalid prior to the expiration of the validity period. Such circumstances include change of name, change of association between subject and CA (e.g., an employee terminates employment with an organization), and compromise or suspected compromise of the corresponding private key. Under such circumstances, the CA needs to revoke the certificate.

Therefore, a verifier should check with the CA to see if any certificate presented to him has been revoked before accepting it. Instead of requiring the verifier to make two connections each time a certificate is received, the RRS can couple sending the CRL with the receipt acknowledgment. This is possible whether or not the RRS is the issuing certificate authority so long as the CRL is signed by the CA. If the RRS is not the CA, the RRS must keep updated, signed CRLs on hand from the CAs issuing the certificates that the RRS handles or fetch them on demand for the verifier.

4.6 Database Design

The RRS stores all necessary data in a database consisting of six tables. The owner information table is shown in Table 4.1. It keeps track of the certificate owners' contact information, including their names, phone numbers, addresses, etc. Table 4.2 describes the certificates table, which stores the certificate and links it to

Field	**Description**
Owner ID	primary key
Name	
Phone	
Email	
Address	

Table 4.1: The owner information table

an owner. Each certificate may have only one owner, although an owner may have more than one certificate.

Each entry in the proof records table corresponds to a proof record. As described in Table 4.3, each proof record entry contains a reference to the certificate, the time window, and a reference to the machine the proof record came from. Each transaction in the transaction records table, described in Table 4.4, contains information about an instance where a certificate ownership proof was sent. Each transaction record references exactly one proof record, although proof records typically reference many transactions. Transaction records contain information recorded by the certificate owner such as the timestamp, the signature material used to prove ownership of the certificate, and the public key and IP address of the verifier.

Table 4.5 describes the receipts table, which contains information recorded by a verifier about a certificate ownership proof. Each receipt should reference at most one transaction record. Receipts that do not reference a transaction records may indicate a stolen certificate, as the certificate owner was not aware that the certificate was presented. The machines table, described in Table 4.6, maps machines to owners.

Field	**Description**
Certificate Hash	primary key
Owner ID	foreign key references Owner Information
Certificate	

Table 4.2: The certificates table

Field	**Description**
Proof Record ID	primary key
Certificate Hash	foreign key references Certificates
Start Time	
End Time	
Machine ID	foreign key references Machines

Table 4.3: The proof records table

Field	**Description**
Transaction Record ID	primary key
Proof Record ID	foreign key references Proof Records
Signature Material	
Verifier Public Key	
Timestamp	
Verifier IP	

Table 4.4: The transaction records table

Field	**Description**
Certificate Hash	foreign key references Certificates
Transaction Record ID	foreign key references Transaction Records
Signature Material	
Timestamp	
Owner Machine ID	foreign key references Machines
Owner IP Address	
Verifier IP	

Table 4.5: The receipts table

Field	**Description**
Machine ID	primary key
Owner ID	foreign key references Owner Information

Table 4.6: The machines table

Chapter 5 — Threat Analysis

Section 1.3 explains how it is impossible to create an airtight proof that a protocol is secure, as can be done in other areas of computer science and mathematics. Schneier discusses how security in real world systems is often vulnerable to the vagaries of hardware, buggy software, economics, and uneducated users [15]. Rather than a proof, security researchers frequently perform a threat analysis of the system to be scrutinized. Schneier calls it "a way to start making sense of the vulnerability landscape" [15]. A threat analysis lists relevant categories of attack and explains how the digital receipts system guards against those attacks. This chapter lists those attacks which are most likely to be executed and discusses how the system thwarts those attacks.

5.1 Attacks

This section explains the protection that the digital receipts system provides against various types of attack. For each attack, the motivation for the attack is explored and the protection the digital receipts system provides is discussed.

5.1.1 Directing the Verifier to a False RRS

Attack: Mallory presents a stolen certificate and attempts to trick the verifier into sending the receipt to the wrong RRS.

Motivation: If the receipt is sent to the wrong RRS, the real RRS will never receive the receipt and thus never detect that the certificate was used without the owner's consent.

Protection: The RRS IP address and a hash of the RRS public key are embedded inside the certificate. The certificate is guaranteed to have integrity since it is signed by a trusted certification authority. The verifier can check the integrity of a certificate to avoid being fooled into sending the receipt to the wrong entity.

5.1.2 Forging Machine Registration

Attack: Mallory registers a machine that does not belong to Alice.

Motivation: If Mallory can falsely register a machine, she can send fabricated proof records from that machine to cover her transactions with a stolen certificate. The RRS will assume Alice is aware of these transactions and will not send Alice any warnings.

Protection: Because machine registration is performed over a TLS connection using client authentication, Mallory cannot register a machine to send proof records for a certificate that she does not own. In order for Mallory to successfully register a machine to send proof records for some certificate, she must either own that certificate or have successfully stolen the private key. Alice becomes aware of the bogus registration when she receives notification of the registration via a side channel or when she connects to the RRS to view the registration history.

5.1.3 Forging Proof Records

Attack: Mallory uses a stolen certificate and then submits a fake proof record to the RRS.

Motivation: An attacker able to forge a proof record can make it appear to the RRS that the certificate owner was aware of transactions performed with a stolen certificate. The RRS never flags the receipts from Mallory's transactions as suspicious because they correspond to transactions on a proof record. Therefore Alice may not realize her certificate has been stolen.

Protection: Each proof record is initially authenticated with the private key of the certificate it represents. This prevents those who have not successfully stolen the certificate and the machine identifier from submitting fake proof records. As explained in Section 3.3, it is unwise to rely on any secret that Alice possesses to authenticate the proof records. If Mallory is able to lift a private key off Alice's machine, there is a chance Mallory has stolen other secrets as well. This is why authentication of proof records does not rely solely on secrets shared between Alice and the RRS. Each owner's machine also registers with the RRS in advance and specifies a schedule for sending proof records. A forged proof record from a successful thief only fools the RRS until the legitimate one arrives. Mallory cannot prevent Alice from attempting to send a proof record (see the assumptions in Section 2.3). Because Alice receives a signed hash of the proof record as acknowledgement from the RRS, she can be sure her legitimate proof record arrived. When it does, the RRS will realize that it has received conflicting proof records for the same time interval and warn Alice.

5.1.4 Altering Proof Records

Attack: Mallory alters a proof record in transit.

Motivation: An attacker able to alter a proof record can make it appear to the RRS that the certificate owner is aware of transactions performed with a stolen certificate. The RRS never flags these transactions as suspicious, and Alice never realizes her certificate has been stolen.

Protection: If Alice sends the proof record over TLS as described in Section 3.1.2.1, the RRS will discover the proof record has been altered due to the properties of TLS. If she chooses to send it by encrypting it with the RRS public key, as described in Section 3.1.2.2, she will receive an acknowledgement from the RRS which includes a hash of the proof record signed with the RRS private key. Alice then knows whether the proof record received by the RRS has been altered.

5.1.5 Hijacking the Proof Records

Attack: Mallory prevents the proof records from arriving at the RRS.

Motivation: If Mallory can submit a fake proof record and then prevent the genuine one from arriving, the RRS will not discover the fake.

Protection: Alice's machine, which sends the proof records automatically according the schedule established in the registration process, receives an acknowledgement from the RRS which includes a hash of the proof record signed with the RRS private key. If this acknowledgement never arrives Alice's machine knows the proof record was never received by the RRS. If after several tries Alice's machine is still does not receive an acknowledgement, it warns Alice.

5.1.6 Forging a Receipt

Attack: Mallory forges a receipt.

Motivation: Forged receipts will not match a transaction on any proof record. They set off false alarms, causing genuine alarms to go unnoticed.

Protection: Unfortunately, the certificate verifiers have no preexisting relationship with the RRS and thus no way to authenticate. Verifiers must prove previous interaction with Alice. The only real evidence of this transaction is the signature on the signature material Alice used to prove ownership. For this reason, the signature material and the signature are included on receipts. This allows the RRS to do an internal consistency check on all incoming receipts and immediately discard those where the signature does not match the signature material. An attacker that possesses a signature material/signature pair can fabricate many receipts based on that one pair. For that reason the RRS has an option which can be set to reject receipts with duplicate signature material.

5.1.7 Altering a Receipt

Attack: Mallory alters a receipt before it reaches the RRS.

Motivation: An attacker capable of altering a receipt could make the RRS think the receipt belonged to another certificate, causing the RRS to have an unmatched receipt for a certificate other than the stolen one. The RRS subsequently sends a warning that the wrong certificate has been stolen.

Protection: Both methods described in Section 3.2 for sending a receipt preserve integrity. If the receipt is sent over TLS it can be assumed to have integrity due to the properties of TLS. If the receipt is sent using the encryption by RRS public key method, a hash of the receipt is signed by the verifier before encryption takes place. This hash lets the RRS verify integrity. In both methods the verifier receives a hash of the receipt signed by the RRS private key as an acknowledgement. If the acknowledgement does not match the receipt the verifier knows the receipt was altered in transit.

5.1.8 Hijacking a Receipt

Attack: Mallory removes the receipt from the network before it reaches the RRS.

Motivation: If Mallory prevents a receipt from reaching the RRS, the RRS will never discover the unmatched receipt. Alice will not be warned if someone else starts using her certificate.

Protection: After sending the receipt, the verifier receives a hash of the receipt signed by the RRS private key as an acknowledgement. If the acknowledgement does not come, the verifier should assume that the RRS never received the receipt. The verifier may retry up to a designated number of times. If the verifier is still unable to send the receipt it should warn Alice via a side channel (See Section 4.3).

If Alice is warned of an aborted receipt for a transaction she did not perform, she should be suspicious someone has stolen her credential.

5.1.9 Forging the Timestamp on a Receipt

Attack: Mallory presents a stolen certificate and sends the verifier a fabricated timestamp indicating that the transaction happened in the distant future.

Motivation: If Mallory fools the RRS into thinking that the receipt is for a transaction that is months in the future, the RRS will not discover it is unmatched until the proof record for that month arrives, causing a successful theft to go undetected for months.

Protection: The RRS does a sanity check on the timestamps of incoming receipts. Any receipt whose timestamp is more than several days off will be flagged as suspicious and Alice will be notified.

5.1.10 Denial of Service Attacks

Attack: Mallory floods the RRS with fake receipts.

Motivation: A legitimate receipt that exposes a certificate theft is dropped due to insufficient RRS resources.

Protection: The RRS makes a receipt validation check by making sure that the signature over the signature material is correct. This causes invalid receipts to be discarded rapidly. The only way for Mallory to generate valid receipts is to have a signature material/signature pair. She can get this in one of two ways. First, she can replay signature material/signature pairs received from a previous transaction with Alice. This is why the RRS checks to see if the signature material/signature pair has been used before. Second, Mallory could create pairs if she had stolen the certificate private key. However, Mallory would have little motivation to forge a receipt for a certificate she had stolen, because the receipt will be unmatched with any transactions and Alice will be alerted, something Mallory wants to avoid.

An attacker with an enormous amount of processing power like a *botnet* might still be able to send enough invalid receipts to overload the RRS with validation checks. A botnet is a large collection of compromised machines running under the command of an attacker. Botnets can include thousands of nodes and harness an enormous amount of processing power. Preventing these type of attacks is outside the scope of this thesis. However, further research into anti-spam measures can be done to explore ways of mitigating such a problem. For example, Back [5] introduces a measure called *hashcash*. Based on requiring clients to compute partial hash collisions, Back claims that hashcash defends server resources from premature depletion. Hashcash provides "graceful degradation of service with fair allocation across users in the face of a DoS attack where one user attempts to deny service to the other users by consuming as many server resources as he can."

Attack	**Defense**
Redirecting the verifier	RRS URL and hash of RRS public key included in certificate
Forging machine registration	Certificate owner notified of all registrations, registrations available upon request
Forging proof records	Proof record protocol requires certificate ownership proof, regularly scheduled proof records
Altering proof records	Proof record protocol provides message integrity
Hijacking proof records	Certificate owner receives signed hash of proof record from RRS
Forging receipts	Receipt cryptographic consistency check
Altering receipts	Receipt protocol provides message integrity
Hijacking receipts	Verifier receives signed hash of receipt from RRS
Forging timestamps	Verifier performs timestamp sanity check
Denial of service	Receipt cryptographic consistency check

Table 5.1: Summary of attacks

5.2 Privacy Concerns

The digital receipts system preserves the privacy of certificate owners from eavesdroppers who control the network and wish to glean information about who a certificate owner is transacting with. All of the receipts and proof records are encrypted either using TLS or using a symmetric key which is in turn encrypted with the public key of the RRS. Both of these methods provide confidentiality, preventing Mallory from snooping through these messages to see who Alice has transacted with. Because the proof records are sent in regular time intervals with the option of padding to obfuscate the length, an attacker would have difficulty even knowing how many transactions a certificate owner made in a given time period. Since the RRS services many different users and certificates, and since receipts are also encrypted, it would be difficult for an attacker to use the receipt exchanges to infer certificate activity for any single user or certificate.

5.3 Hardware and Network Failures

If Alice's machine, due to hardware failure or some other reason, does not send its proof record, the RRS waits a certain amount of time and then warns Alice via a side channel (see Section 4.3) that no proof record has been received.

If a proof record is sent, but does not arrive at the RRS due to network failure, Alice's machine that is sending the proof records knows because it does not receive an acknowledgement hash from the RRS. Alice's machine retries for a certain time period and then warns Alice that the RRS appears to be down. If the condition persists, Alice should contact the RRS administrators.

If a receipt does not get delivered, the verifier knows because no acknowledgment has been received from the RRS. The verifier may retry or inform Alice that the RRS or the network appears to be down. If a verifier chooses not to send a receipt, the RRS will have an unmatched proof record transaction. Though this does not necessarily cause an alarm, it could cause a stolen certificate to go undetected. If Mallory only uses a stolen certificate at places that do not send receipts, obviously there will be no unmatched receipts and the RRS will not detect the compromise. For this reason Alice may refuse to conduct business with a verifier that consistently refuses to send receipts. Boycotts by certificate owners might give such businesses an incentive to send receipts.

If the RRS computers go down, no receipts, proof records, or registrations can be delivered. Having the RRS computers go down would prevent stolen certificates from being discovered for at least the duration of the outage.

Chapter 6 — Implementation

In order to demonstrate the feasibility of a digital receipts system, we created a software prototype that integrates the system into TrustBuilder, a middleware trust agent that uses digital certificates and policies to facilitate the establishment of trust between strangers. The remainder of this chapter includes the major design issues addressed during prototype creation.

6.1 TrustBuilder

TrustBuilder is under development at the Internet Security Research Lab at Brigham Young University. Written in Java, TrustBuilder manages keys, credentials, and policies for a negotiation party and determines which credentials and policies should be released at any point during a negotiation. TrustBuilder is built around IBM's Trust Establishment system, which maps attributes contained in X.509v3 credentials to roles. Trust Establishment can be obtained at www.alphaworks.ibm.com. TrustBuilder also handles the process of proving certificate ownership. Signature material is exchanged in hello messages at the start of each negotiation.

During negotiation, TrustBuilder checks a user's policies see which certificates can be released to fulfill the other negotiating party's policies. It uses the private keys corresponding to these certificates to sign the signature material associated with the trust negotiation session. We extended TrustBuilder with methods to facilitate the sending of proof records and receipts.

When TrustBuilder receives a certificate sent by the other negotiating party, it first verifies that the signature material is signed correctly. If so, it will place it in a cache of certificates used to determine if the other negotiating party fulfills the current policy. The modified version of TrustBuilder sends a receipt to the Receipt

Resolution Server (RRS) at this point. TLS connections are made using the Java Secure Sockets Extension library and make use of Java KeyStores.

Another modification to TrustBuilder ensures that the timestamp on the proof record matches the timestamp in the receipt. The negotiating parties agree beforehand on a single timestamp that proof record transactons and receipts associated with the trust negotiation will bear. The client selects the timestamp and sends it to the server in the hello message.

6.2 X.509v3 Certificates

In order to work with receipts, the certificates must contain some additional information about where to send the receipt and how to authenticate the RRS, as described in Section 3.2.2. X.509v3 certificates are capable of having such extensions through the addition of custom fields. These fields, called OID fields, include a key/value pair where the key is a sequence of numbers separated by dots, like 1.2.3.4.1. Therefore, the attributes in Table 3.3 were mapped to OID keys and each certificate used in the modified TrustBuilder is required to contain appropriate values mapped to those keys.

6.3 Proof Record Sender (PRS)

We could have implemented the sending of the proof records into TrustBuilder itself. However, there are two major problems with this. First, TrustBuilder would always have to be running so that it could send proof records at the appropriate intervals. Second, TrustBuilder would have to be made aware of certificate transactions using other applications. Therefore, we created a separate module to handle the sending of the proof records. This module also handles machine registration with the RRS as described in Section 3.3.

This proof record sender module runs constantly and collects certificate transaction information from any application using a certificate. It stores this information in a MySQL 5.0 database. Communications to this database are made using the Java Database Connectivity *JDBC* libraries. To create a proof record, the proof record sender uses SQL queries to collect all the transaction information from the database for the time window of the proof record and sends it to the RRS. It also keeps track of which proof records were not successfully received by the RRS in order to attempt resending them at a later time.

6.4 Receipt Resolution Server (RRS)

The RRS software always has threads running to handle the reception of registrations, proof records, and receipts. Like the proof record sender, the RRS stores the receipt and proof record information in a MySQL 5.0 database which it connects to over JDBC. The proof record sender uses SQL queries to match the transaction information in the proof record with the receipts. In the prototype, unmatched receipt alerts are printed to the screen. In a real world system such alerts would cause the RRS employees to make efforts to contact the certificate owner. These alerts could also be automated.

Chapter 7 — Performance Analysis

One potential drawback of the digital receipts system is that it increases network traffic. To see how the system would impact network congestion, a queuing theory analysis was performed using data from Brigham Young University's network and from the implementation of the digital receipts system into trust negotiation as explained in Chapter 6. This performance analysis first discusses traffic loads on the BYU network. Then, a queuing theory analysis is performed on the most saturated link in the network under atypically busy conditions. This analysis shows the effects on queuing time, queuing size, and utilization as the percentage of trust negotiation traffic to total traffic ranges from 0 to 100%.

7.1 BYU's Network

Due to security concerns, we do not include detailed information about the BYU network topology. By far the busiest link a packet would traverse en route from the Computer Science building to the Internet is a 500 Mbps duplex link. The peak inbound traffic across this link in a 60 day period during a busy semester was 373 Mbps and outbound traffic peaked at 333 Mbps. Peaks generally occur between 10 a.m. and 6 p.m. Network traffic decreases by roughly 50% from this peak during 2 a.m. to 7 a.m. An employee at BYU's Office of Information Technology estimated that the percentage of authentication traffic to total traffic is less than 1%.

7.2 Queuing Theory Analysis

In order to adequately predict the impact of receipts on a network, the analysis of the network is performed under a peak load. Therefore, we assume the link usage is 373 Mbps. Then the maximum utilization, signified by p, over the 60 day period is calculated as follows

$$\begin{aligned} p &= (373 * 10^6\, bits/second)/(500 * 10^6\, bits/second) \\ &= .746 \end{aligned}$$

Trust negotiation packets have an average size of 276 bytes. Assuming packet size follows an exponential distribution, the arrival rate λ of packets to the link can be calculated as

$$\begin{aligned} \lambda &= 373 * 10^6\, bits/second * 1\, byte/8\, bits * 1\, packet/276\, bytes \\ &= 168,931\, packets/second \end{aligned}$$

Then, given λ and p, the average service time T_s, average queue size w, and average waiting time T_w of packets at the link can be calculated.

$$\begin{aligned} T_s &= p/\lambda \\ &= .746/(168,931\, packets/second) \\ &= 4.416 * 10^{-6}\, seconds \end{aligned}$$

$$\begin{aligned} w &= p^2/(1-p) \\ &= .746^2/(1-.746) \\ &= 2.19 \end{aligned}$$

$$\begin{aligned} T_w &= p * T_s/(1-p) \\ &= .746 * 4.416 * 10^{-6}\, seconds/(1-.746) \\ &= 1.297 * 10^{-5}\, seconds \end{aligned}$$

A typical trust negotiation that involves the exchange of two certificates and two policies results in 8,108 bytes being sent over the network. This total does not include the actual transfer of the resource. Each certificate proof requires that a receipt be sent, adding an additional 1,480 bytes per certificate. Thus, the total number of bytes exchanged for such a receipts enabled trust negotiation is 11,068. We can calculate the ratio of bytes sent during a receipts enabled trust negotiation to a trust negotiation not using the receipt system as 11,068/8,108, which equals 1.365.

Let us consider the ratio of trust negotiation traffic to total traffic as a variable h, which will range from 0 to 1. The ratio of non-trust negotiation traffic to total traffic is then (1-h). If all of the trust negotiation traffic began using the digital receipts system it would increase the packet-arrival rates to the network links. Let λ be the packet-arrival rate with no traffic implementing the digital receipts system. Let us call the packet-arrival rate with h fraction of traffic being trust negotiations implementing the digital receipts system λ'. Then

$$\begin{aligned}\lambda' &= (1.365\lambda h + \lambda(1-h))\\ &= (1.365 * (168,931\,packets/second)h + (168,931\,packets/second)(1-h))\\ &= 61,659.8h + 168,931\end{aligned}$$

This would increase our utilization according to the function

$$\begin{aligned}p' &= \lambda' * T_s\\ &= (61,659.8h + 168,931) * 4.416 * 10^{-6}\,sec\\ &= .272h + .746\end{aligned}$$

We can then calculate what ratio of trust negotiation traffic using the digital receipts system to total traffic would cause p to exceed 1, causing the queues to permanently overflow. This is calculated by solving the equation for h.

$$
\begin{aligned}
1 &= .272h + .746 \\
h &= .934
\end{aligned}
$$

This means that if 93.4% of the traffic across the link was trust negotiation traffic and all trust negotiation traffic began using the digital receipts system, the queues would permanently overflow. We can also calculate the effect on queue sizes and waiting time.

$$
\begin{aligned}
w' &= p'^2/(1-p') \\
&= (.272h + .746)^2/(1-(.272h + .746)) \\
&= -.272(h+2.74)^2/(h-.934)
\end{aligned}
$$

The graph in Figure 7.1 shows that as the percentage of trust negotiation traffic increases, the average number of packets queued when receipts is used ranges from 2.18 to infinity.

One can also calculate the average queuing time as a function of h.

$$
\begin{aligned}
T'_w &= p' * T_s/(1-p') \\
&= (.272h + .746) * 4.416 * 10^{-6}\, seconds/(1-(.272h + .746)) \\
&= -4.416 * 10^{-6} * (h + 2.743)\,/(x - .934)
\end{aligned}
$$

T'_w produces the graph shown in Figure 7.2, time which shows that as the percentage of trust negotiation traffic increases, the average queuing time varies from $1.297 * 10^{-5}$ seconds to infinity.

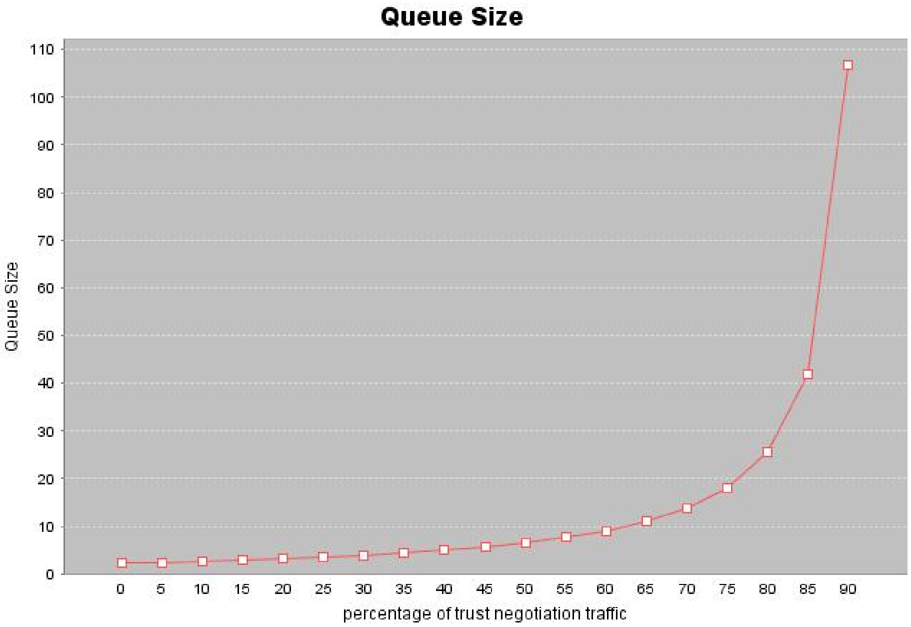


Figure 7.1: Average queue size as a function of h

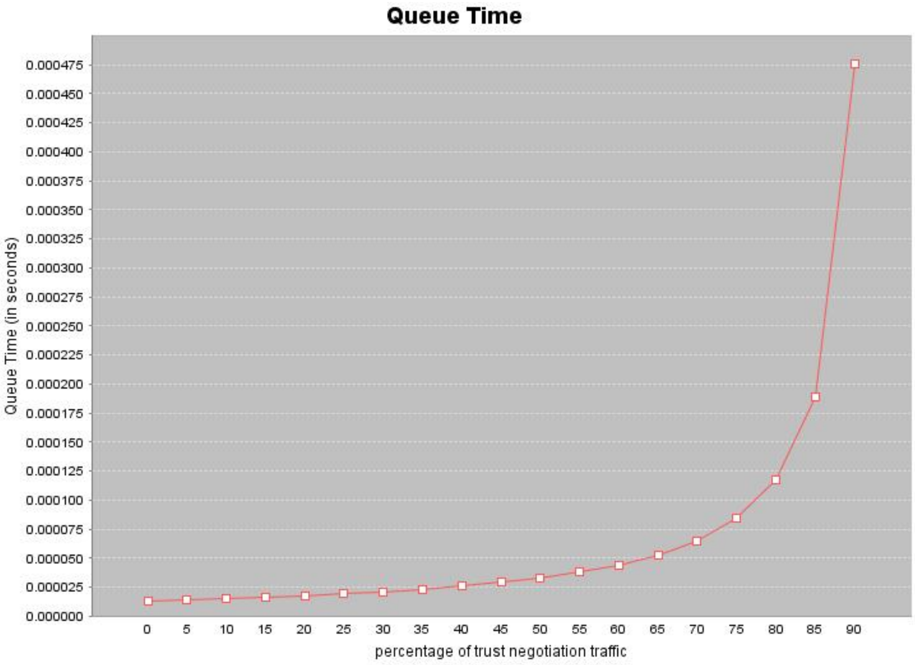


Figure 7.2: Average queue time as a function of h

7.2.1 Conclusions

This section has shown some of the effects of the digital receipts system on traffic based on a real network under peak loads. Queuing theory has been used to show the effect on utilization, average queue sizes, and average queue times for the busiest link a packet would traverse between a Computer Science department workstation and the Internet in the BYU network. Figures 7.1 and 7.2 show that queue sizes and waiting times remain manageable so long as trust negotiation traffic is less than half of all traffic. As the percentage of trust negotiation traffic to total traffic approaches 93.4%, the queue sizes approach infinity. Since an employee at BYU's Office of Information Technology estimates that authentication traffic is less than 1% of total traffic, implementing the digital receipts system should not seriously affect queue times and sizes. Furthermore, these calculations were made under worst case scenarios using the busiest link under a peak load for a 60 day period. This link had a utilization of .746, while most other links in the network are very underutilized, having a utilization ranging from .012 to .024 during peak loads. Therefore, the impact of the receipts system on BYU's network is likely to be minimal.

Chapter 8 — Conclusions and Future Work

The three pillars of security are prevention, detection, and response. The vast majority of the research in cryptography and protocols focuses on prevention. CRLs, as described in section 4.5, deal with response. This system addresses detection, an area that has been largely ignored in the realm of private key compromise. It does this by introducing a system that alerts owners when evidence suggests that their private keys may have been stolen.

The digital receipts system can detect the usage of stolen digital certificates, is secure against a wide variety of attacks, and has minimal network overhead given the topology and typical loads of a real network. It detects both stolen private keys and breaches in certificate ownership protocols, mitigating the fact that proving protocol security is currently infeasible. The system cannot completely secure transactions that use digital certificates, but does ameliorate some of the common problems associated with using these certificates.

This thesis has identified several important areas for future work.

1. Currently, protection against denial of service attacks is limited to the internal consistency check on the receipt. It would be interesting to explore how to use a system like hashcash to thwart denial of service attacks.

2. As mentioned in Section 3.2, research could reveal additions to the system that would allow the Receipt Resolution Server (RRS) to more accurately pinpoint the source of a stolen certificate.

3. More studies on the performance impact of digital receipts could be done. Research projects could include the following:

   - How would the system affect the load on a busy web server?
   - What system load would make it more efficient to restructure the RRS as a back-end database with several front-end web servers?
   - How much effort would it take to overwhelm an RRS?
   - How does this system contribute to self-similar network traffic patterns?
   - Would granting the packets from this system a high priority cause more dropped packets from other traffic flows?

4. The concepts introduced in this thesis can be extended to other systems that use knowledge of a secret to authenticate. For example, suppose a certain system requires a username and password to login. One can detect password theft by comparing system login records to user login records. If extra logins appear in the system record, someone may have stolen the password.

Ultimately, the decision whether or not to use receipts is an extension of the classic tradeoff between security and performance. Network administrators should consider whether the additional network traffic is worth early detection of a compromised private key. Further research could create a cost model to compare the economic cost of the loss of a private key with the cost of increased network overhead to demonstrate whether or not receipts would be justified from a financial standpoint.

References

[1] National and state trends in fraud and identity theft. Technical report, Federal Trade Commission, February 2005.

[2] M. Abadi and R. Needham. Prudent engineering practice for cryptographic protocols. In *IEEE Transactions on Software Engineering*, January 1996.

[3] R. Anderson and R. Needham. Programming satan's computer. In *Computer Science Today: Recent Trends and Developments*. Springer LNCS 1000, 1995.

[4] R. Anderson and R. Needham. Robustness principles for public key protocols. In *15th Annual International Cryptology Conference*, Santa Barbara, California, August 1995.

[5] A. Back. Hashcash - a denial of service counter-measure. http://www.hashcash.org/papers/hashcash.pdf, August 2002.

[6] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. In *ACM Transactions on Computer Systems*, volume 8 number 1, 1990.

[7] T. Dierks and C. Allen. The tls protocol version 1.0. http://www.ietf.org/rfc/rfc2246.txt, January 1999.

[8] W. Diffie, P. C. van Oorschot, and M. J. Wienter. Authentication and authenticated key exchanges. In *Designs, Codes, and Cryptography*. Springer, 1992.

[9] P. Gutmann. Lessons learned in implementing and deploying crypto software. In *11th Usenix Security Symposium*, San Fransisco, CA, August 2002.

[10] J. Holt, R. Bradshaw, K. E. Seamons, and H. Orman. Hidden credentials. In *2nd ACM Workshop on Privacy in the Electronic Society*, Washington, DC, October 2003.

[11] R. Housley, W. Polk, W. Ford, and D. Solo. Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. http://www.ietf.org/rfc/rfc3280.txt, April 2002.

[12] A. Levi. Performance evaluation of public-key cryptosystem operations in wtls protocol. In *Proceedings of the 8th IEEE Symposium on Computers and Communications*, Antalya, Turkey, June 2003.

[13] A. D. Rubin and P. Honeyman. Formal methods for the analysis of authentication protocols. Technical Report 93-7, Center for Information Technology Integration, 1993.

[14] B. Schneier. *Applied Cryptography.* John Wiley and Sons, 1996.

[15] B. Schneier. *Secrets and Lies: Digital Security in a Networked World.* John Wiley and Sons, 2000.

[16] W. Stallings. Queuing analysis. In *www.williamstallings.com/StudentSupport.html*, 2000.

[17] W. Stallings. *Network Security Essentials.* Pearson Education, 2003.

[18] K. Seamons W. Winsborough and V. Jones. Automated trust negotiation. In *DARPA Information Survivability Conference and Exposition*, Hilton Head, SC, January 2000.

[19] A. Whitten and J. D. Tygar. Why johnny can't encrypt: A usability evaluation of pgp 5.0. In *8th USENIX Security Symposium*, Washington DC, August 1999.

[20] A. Yasinsac. *Evaluating Cryptographic Protocols.* PhD thesis, University of Virginia, 1996.

[21] V. Zorkadis. Security versus performance requirements in data communication systems. In *Third European Symposium on Research in Computer Security*, Brighton, United Kingdom, November 1994.