UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

TCL Industries Holdings Co., Ltd.,
Petitioner

v.

Koninklijke Philips N.V.,
Patent Owner

PTAB Case Nos. IPR2021-00495, IPR2021-00496 and IPR2021-00497
Patent Nos. 9,590,977 B2 and 10,298,564 B2

DECLARATION OF SETH JAMES NIELSON, PH.D. IN SUPPORT OF PETITION FOR *INTER PARTES REVIEW* OF U.S. PATENT NOS. 9,590,977 B2 AND 10,298,564 B2

TABLE OF CONTENTS

Page

TABLE OF CONTENTS

TABLE OF CONTENTS

Page

TABLE OF CONTENTS

Page

TABLE OF CONTENTS

TABLE OF CONTENTS

Page

I. INTRODUCTION

1. My name is Seth James Nielson. I am the founder and chief scientist at Crimson Vista, Inc., a cybersecurity research and engineering firm. In addition to conducting research for government, legal, and industry clients, I hold an appointment as an Adjunct Professor at the University of Texas at Austin where I teach courses on cybersecurity and privacy.

2. I have been engaged by TCL Industries Holdings Co., Ltd. ("TCL" or "Petitioner") as a consultant in connection with TCL's Petitions for *Inter Partes* Review (IPR) of U.S. Patent Nos. 9,590,977 B2 (the "'977 Patent") and 10,298,564 B2 (the "'564 Patent").

3. I understand that the '977 and '564 Patents are assigned to Koninklijke Philips N.V. ("Philips"). Philips is also referred to as the "Patent Owner" in this declaration.

4. The '977 and '564 Patents have the same detailed description, figures and similar but not identical claim limitations. In this declaration I will provide my analysis for both the '977 and '564 Patents together, and will provide separate explanations as may be needed to account for the differences in the '977 and '564 Patent claims.

5. This declaration is based on the information currently available to me. To the extent that additional information becomes available, I reserve the right to

continue my investigation and study, which may include a review of documents and information that may be produced, as well as testimony from depositions that not yet been taken.

II. SUMMARY OF OPINIONS

6. After analyzing the evidence, it is my opinion that that with respect to the '564 patent:

- claims 1-2, 5-10, 14, 17-22, 25, and 28 are rendered obvious by Maillard, Davis, and Lundkvist.
- claims 1-4, 7, 15-16, 19, and 28 are rendered obvious by Maillard, Davis, Lundkvist, and Chaum.
- claims 11 and 23 are rendered obvious by Maillard, Davis, Lundkvist and Schneier

7. It is also my opinion that with respect to the '977 patent:

- claims 1-3, 8-10 are rendered obvious by Maillard, Davis, and Lundkvist.
- claims 1-3, 8-10 are also rendered obvious by Maillard, Davis, Lundkvist, and Schneier.
- claims 11-12, 14-17 are rendered obvious by Maillard, Davis, and Lundkvist.
- claims 19, 20 are rendered obvious by Maillard, Davis, Lundkvist and HAC.

- claim 18 is rendered obvious by Maillard, Davis, Lundkvist, and Schneier.

8. The following sections will first provide my qualifications and experience and then describe the details of my analysis and observations.

III. QUALIFICATIONS AND EXPERIENCE

A. Education and Experience

9. I am a subject matter expert in cybersecurity, including the sub-fields of applied cryptography and network security. I have twenty years of experience in both academic and industry circles. This experience includes providing research to US government agencies, assisting corporations ranging from start-up's to fortune 100, and teaching graduate and undergraduate courses in universities.

10. In the years 2000 and 2004 respectively, I earned a B.S. and M.S. in Computer Science from Brigham Young University (BYU) in Provo, UT. In 2009, I completed my Ph.D. in Computer Science from Rice University in Houston, TX.

11. During both my bachelor's degree and my master's degree, I published peer-reviewed research related to topics of computer networks and software engineering.

12. At Rice University, I studied topics of network security, cryptography, and other areas of cyber-security. I published papers related to vulnerabilities in search engines, including Google's search engine, and also in peer-to-peer

communications protocols (predecessors to blockchain). Rice University awarded me with the Brown Fellowship and the John and Eileen Tietze fellowship in recognition of my teaching and research contributions.

13. I have been teaching courses at the university level since 2014 when I was appointed as a Lecturer at Johns Hopkins University. I was subsequently promoted to Adjunct Associate Research Scientist in 2015 and appointed the Director of Advanced Research Projects for the Information Security Institute in 2016. I also held an appointment at the Johns Hopkins Applied Research Lab for collaborative research projects.

14. In 2019, I left Johns Hopkins University and received an appointment as an Adjunct Professor at the University of Texas at Austin. I currently teach the undergraduate Network Security and Privacy class as well as an Introduction to Cybersecurity Technology course in the law school. I am a Cybersecurity Fellow in the Robert Strauss Center for International Security and Law.

15. I develop the various curricula used in all of my courses. Course topics and lab work include standard elements such as cryptographic protocols such as Transport Layer Security ("TLS"), a protocol widely used for secure communications over the internet. Other assignments include the proper use and management of digital certificates, especially in browsers and web servers, as well as techniques used by malicious parties to circumvent these defenses. Students are

expected to become familiar with vulnerabilities in computer code, defensive systems such as firewalls, and the implications for privacy in data handling.

16. During my development of curriculum at Johns Hopkins, I created a novel approach to teaching some of these concepts. I subsequently published a paper about this curriculum entitled, "PLAYGROUND: Preparing Students for the Cyber Battleground" in the Journal of Computer Science Education.

17. During my time at Hopkins, I also mentored master's students in their capstone research and have published peer reviewed papers on the topics of Internet-of-Things ("IoT") forensics and vulnerabilities of chemical manufacturing systems.

18. In 2018, a research group and I produced a technical report from their capstone entitled, "Securing ADS-B Based Airborne Collision Avoidance Systems." This project added a cryptographic protocol on top of an otherwise unsecured communications system to protect the data transfer and prevent an attacker from faking the safety messages. Our work has been submitted to an industry partner that is promoting our design and submitting it for consideration with the FAA.

19. While at Johns Hopkins University, I co-founded a research project called "Crypto Done Right." This project was funded, in part, by Cisco. The goal of the project is to develop educational, training, and outreach resources for

improving the use of cryptography in industry. Although I have left Johns Hopkins, I continue to collaborate with the project (https://cryptodoneright.org). The project is currently transitioning into an independent non-profit.

20. In addition to my many qualifications from academia, I also have a deep background in industry. From 2000 to 2003, I worked as a software engineer at multiple companies, developing various applications and network components.

21. In 2005, and in connection with the research on vulnerabilities in Google's search engine, I worked as a summer intern at Google in their computer security group.

22. Overlapping with my Ph.D. program, I began work for Independent Security Evaluators in 2005. I developed cryptographic libraries that implemented various encryption algorithms, key transport algorithms, and related operations. I also helped prepare these libraries for tests and certifications.

23. After completing my Ph.D., I took on additional responsibilities at Independent Security Evaluators. I developed hardware accelerated cryptography systems, built an encrypted file-system, and analyzed cryptographic protocols for vulnerabilities. I also led the development of a secure communications product from prototyping and design to implementation and test. This product improved on the security of TLS by dividing trust between multiple Certificate Authorities. My work on the secure communications technology project led to the issuance of

multiple patents including U.S. 8,745,372 entitled "Systems and Methods for Securing Data in Motion."

24. In 2011, I began work as a Research Scientist at Harbor Labs and was eventually promoted to Principal. I served a wide range of clients providing them with specialized consulting in cyber security. These projects included an analysis of the privacy controls used by an anti-piracy vendor selected by the Motion Pictures Association of America ("MPAA") and the Radio Industry Association of America ("RIAA"). A redacted executive summary of our findings is publicly available. I also led our investigation into the vulnerabilities of devices produced by a major medical manufacturer, including their digital-certificate based authentication management.

25. In 2016, I founded Crimson Vista, a computer-security focused research and consulting company. I advise companies on the correct use of cryptography and proper cyber-security customs and practices. I actively develop new technologies and applications related to cryptography. I am a named inventor on three patents related to cryptography and have applications currently pending. In 2018, I was invited to speak at a workshop on cyber-deception and presented, "Can Software Know What's Real" and served on the Program Committee of the International Cryptographic Module Conference (ICMC). I subsequently served on the Program Committee of ICMC in 2019 and 2020.

26. In December 2018, Crimson Vista was awarded a research contract from the U.S. Army to develop new recovery technology for computers compromised by ransomware, a malicious technology used by hackers that locks up a victim's computer and is only released in exchange for money. I am the Primary Investigator and lead the research team. The technology we are developing includes analyzing cryptographic keys used by attackers in locking up victim's data and resources.

27. I am the co-author of the book, "Practical Cryptography in Python," which was published 2019. That same year, I was also invited to a panel at the Defense Strategies Institute's 7th Annual DoD Unmanned Systems Summit where I contributed to a discussion of "Hardening US Unmanned Systems Against Enemy Counter Measures."

28. In both 2020, and now in 2021, I have been assisting multiple clients with analyzing cyber incidents, including one for a Fortune 100 company. In these efforts, I am looked to bridge the gap between what went wrong technically (verifying and correcting previous analyses) with what it "means" in terms of impact. I have provided analyses about the potential impact to customers of the companies, including PII loss and so forth.

29. I have testified for legislative bodies on matters related to cyber-security and I have also testified as an expert witness in multiple district courts and

the Patent Trial and Appeal Board. Based on my extensive experience, and my recognition in the fields of computer security and cryptography, I am qualified to provide opinions in this matter.

B. Compensation

30. I am being compensated by TCL for my work in connection with this declaration. The compensation is not contingent upon my performance, the outcome of the IPR or any other proceeding, or any issues involved in or related to the IPR.

C. Documents and Other Materials Relied Upon

31. My opinions expressed in this declaration are based on documents and materials identified in this declaration, including the ’977 and ’564 Patents, the prior art references and background materials discussed in this declaration, and any other references specifically identified in this declaration. I have considered these materials in their entirety, even if only portions are discussed here.

32. I have considered a number of prior art references including the ones listed in the references listed in the Exhibit list of the TCL IPR Petitions.

33. I have also relied on my own experience and expertise in computer engineering and information security systems and protocols. In doing so, I have kept in mind that the priority date of the ’977 and ’564 Patents is June 27, 2003. All of the prior art references used to challenge the claims at issue pre-date this

priority date by more than one year.

34. All Exhibit numbers used in this declaration refer to Exhibits to TCL's IPR Petitions.

IV. STATEMENT OF LEGAL PRINCIPLES

A. Obviousness

35. TCL's counsel has also advised me that obviousness under pre-AIA 35 U.S.C. § 103 effective before March 16, 2013, is a basis for invalidity of a patent. Specifically, I understand that where a prior art reference discloses less than all of the limitations of a given patent claim, that patent claim is invalid if the differences between the claimed subject matter and the prior art reference are such that the claimed subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the relevant art. Obviousness can be based on a single prior art reference or a combination of references.

36. I understand that obviousness is not driven by a rigid formula but is instead a flexible inquiry that reflects the fact that a person of ordinary skill in the art exercising ordinary creativity may find a variety of reasons to combine the teachings of different references. I understand that a non-exclusive list of possible factors that may give a person of ordinary skill in the art a reason to combine references includes any one or more of the following:

- The scope and content of the prior art;
- The difference or differences between the subject matter of the claim and the prior art (whereby in assessing the possibility of obviousness one should consider the manner in which a patentee and/or a Court has construed the scope of a claim
- The level of ordinary skill in the art at the time of the alleged invention of the subject matter of the claim;
- Whether simple substitution of known elements obtains predictable results; and
- Some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to combine prior art reference teachings to arrive at the claimed invention.

V. OVERVIEW OF THE '977 AND '564 PATENTS

A. Summary of the '977 and '564 Patents

37. The shared specification of the '977 and '564 patents states that, "[i]t is an object of the invention to obtain a solution to the problem of performing a secure transfer of content within a limited distance." '977 at 2:39-41.

38. The '977 Patent discloses and claims an authentication procedure that includes three main operations:

1) first device authenticates the second device - this is shown in Figure 2 of the ‘977 Patent as step 205;

2) the first device shares a common secret securely with the second device - step 207 in Figure 2; and

3) the first device determines if the second device is within a certain distance by measuring round trip time of signals that are sent in a challenge-response procedure that uses the common secret - this is shown in step 209 in Figure 2.

39. Once the first three steps are completed, the last step 211 in FIG. 2 is to communicate the data between devices 201 and 203.

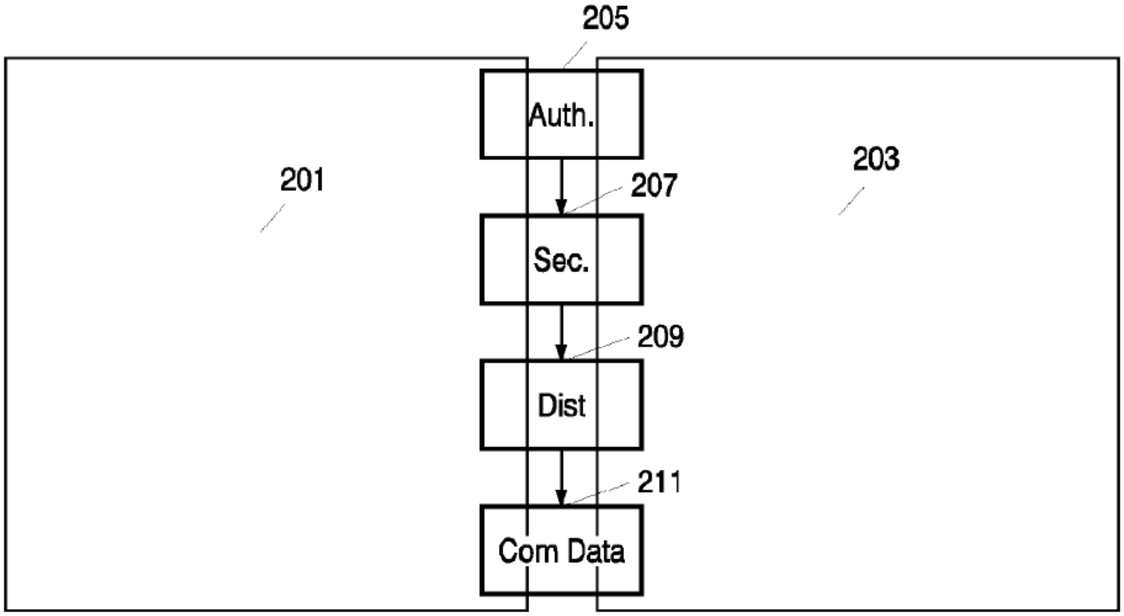


FIG. 2

40. The ‘564 Patent, which shares the same specification as the ‘977 Patent, is directed toward the same technologies with minor differences in claim language.

B. The Challenged Claims of the ’977 and ’564 Patents

41. The challenged claims of the ‘977 Patent are: claims 1-3, 8-12 and 14-20. The challenged claims in the ‘564 Patent are: claims 1-11, 14-23, 25 and 28.

C. State of the Art Prior to the ’977 and ’564 Patents

42. Considering the prior art I have reviewed and my own personal experiences, it is my opinion that the Philips’ patents do not claim anything inventive related to secure cryptographic communications. All of the technology described and claimed in the Philips’ patents was already well known at the time of the alleged invention. Below I have provided an overview of the technologies available at the time of the claimed invention of the Philips’ patents.

1. A Very Brief Overview of Cryptographic Communications

43. Cryptographic communications are the lifeblood of the contemporary always-on, always-connected Internet world. There is no true sense of “location” in cyber-space. I always tell my students that when you connect to the Internet, you are connected to *everyone* on the Internet. Everyone includes criminals, terrorists, spies, and other untrustworthy characters. The primary technology that

keeps our systems, data, and operations "separate" and safe is cryptography. Sadly, most of the general public has no idea what cryptography is.

44. For those individuals that have heard of cryptography, they almost immediately think of what computer security professionals call *confidentiality*. Confidentiality is typically achieved using *encryption* wherein readable data (called *plaintext*) is converted into something unreadable (called *ciphertext*). The goal for confidentiality is to make ciphertext data that it is *literally indistinguishable from completely random data*. Security Engineering at pp. 138-139.

45. At the same time, the unreadable data can be converted back into readable data for someone possessing the right *key*. A key is just data, but it is critical data that enables an algorithm to convert the seemingly random ciphertext back into plaintext.

46. But while confidentiality is crucial and necessary, so is the concept of *authentication*. Authentication is a broad topic, but the overarching idea is confirming the identity of a party who makes a claim as to their identity and confirming that data received claiming to be transmitted by a party is actually from that party. HAC (Ex. 1009) at p. 4.

47. For example, when someone connects to their bank's website using a web browser, how do they know they have actually connected to their bank and not

a criminal *claiming* to be their bank? It is critically important that the communications between the user and the bank be confidential (nobody in-between can read the data), but it is just as important that the user knows that it is really his or her bank! And it is important that every byte of data received by the user's browser be authenticated to ensure that all of it came from the bank and came from the bank unmodified.

48. Both confidentiality and authentication are necessary for secure communications. There are other requirements as well, but as confidentiality and authentication are the critical technologies of the asserted patents, this background section focuses on these two core concepts.

2. The Basics: Symmetric and Asymmetric Encryption

49. Both confidentiality and authentication can be achieved using two basic building blocks that emerged or matured in the 1970's: symmetric and asymmetric encryption.

50. By way of explanation, symmetric encryption uses the same key to both encrypt and decrypt. The key, sometimes referred to as the conventional, secret key or shared key, is known and used by both parties. In general, if encrypted data is to be shared between multiple parties, the key must be shared

somehow before encryption and/or decryption can take place. For this reason, a symmetric key is sometimes referred to as "secret key."[1] HAC at pp. 15-16.

51. To use an analogy, this is similar to the safes with digital access codes that you find in hotel rooms around the country. You enter the same access code to lock the safe that you enter to unlock the safe. You could use this kind of safe to send physical data between two people so long as both individuals had the same combination access code (key). Without that access code, you cannot see what is inside the safe.

52. While symmetric ciphers existed before the 1970's, it was during this decade that the DES (Data Encryption Standard) algorithm was developed. DES was standardized in 1977 and formed the foundation of symmetric cryptography for decades. A variant of DES called 3DES (pronounced "triple DES") is still used in a few places. HAC at 250, 272, 277.

53. It was also during this same decade that Diffie, Hellman, and Merkle developed and published the concepts of asymmetric cryptography for the first time, including a seminal paper in 1976 and a patent in 1977. Asymmetric encryption is a complementary technology wherein there is a true private key not

[1] It is also sometimes called "private-key encryption" but this is extremely confusing given that private-key's in "public key" encryption can also encrypt. In my declaration, I will use "secret key" or "shared key" for symmetric encryption algorithms.

shared with anyone else ever. For a given private key, there is a corresponding public key that is intended to be shared, potentially with everyone. Unlike symmetric encryption where the same key is used to both encrypt and decrypt the data, what is encrypted[2] by the public key can only be decrypted with the private key, and what is encrypted with the private key can only be decrypted with the public key. For this reason, asymmetric encryption is sometimes referred to as "public/private key encryption" or "public key encryption" for short. HAC at p. 283.

54. Or, in other words, what a party can encrypt (with the private key), the world can decrypt (with the public key). And what the world can encrypt (with the public key), the party can decrypt (with the private key).

55. There are a couple of ways to think about asymmetric encryption. One analogy is the book drop at a library. You can put the books in, but you cannot get them back out. At the same time, you know that only the library can get those books.

56. Another analogy would be having a friend give you an open combination lock for which you do not know the combination. You can lock

[2] There are many asymmetric algorithms beyond encryption. However, for simplicity in this background, I discuss encryption only, as that is the aspect of asymmetric cryptography most relevant to the prior art and challenged claims.

anything with the lock but, once you do, you cannot reopen the lock. Only your friend can.

57. These analogies are not perfect, because they only illustrate the concept of encrypting with the public key and decrypting with the private key, but they do give some intuition for "one way" unlocking.

3. Authentication Protocols and Key Exchange

58. Many of the limitations of the claims of the '977 and '564 patents are related to authentication. One component that is heavily related to confidentiality is *key exchange*. Because symmetric encryption is many times faster than asymmetric encryption, and because asymmetric encryption has other limitations beyond the scope of this declaration, symmetric encryption is used more-or-less exclusively for bulk encryption of data for confidentiality.

59. At the same time, and also for reasons beyond the scope of this declaration, it is *highly* recommended in many applications to use a one-time key for the confidential communication session between two parties. Because this key is used just once and then destroyed, it is sometimes called an *ephemeral key*. HAC at pp. 493-494; Schneier (Ex. 1008) at pp. 15-17.

60. Succinctly: two parties exchange or agree on a short-term key for a confidential communication session.

61. For all the reasons that I explained previously, it was well understood in the cryptography community before 2003 that confidentiality is insufficient for most secure communications. Instead, it is almost always required to combine key exchange (or key agreement) with an authentication protocol. Accordingly, protocols based on symmetric and asymmetric encryption have been used to provide authentication and key exchange since at least the 1970's. See "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems" generally (https://people.csail.mit.edu/rivest/Rsapaper.pdf).

62. By way of emphasis, large portions of the claims of the '977 and '564 patents are just standard cryptography that was well understood before 2003.

63. To illustrate, in the following paragraphs I will walk through a few simplified examples of well-known cryptographic protocols, when these protocols were first used, and their relationship to the relevant claim language of the patents.

64. Also, while I will not delve into all of the formal notation used in cryptography, a few notations are helpful and space-saving. These are:

1. Party A will be called "Alice" and party B will be called "Bob".
2. Messages will be denoted with "M" (multiple messages are M0, M1, etc.).
3. Keys are denoted K.
4. A key shared by Alice and Bob is K_{AB}.

5. For a public key K, the corresponding private key is K^{-1}.
6. A message M encrypted by key K is written {M}K.
7. A hash function is denoted by H.
8. A hash value of some data or message is denoted by H(data)[3].

4. Challenge Response

65. In cryptography, the basic *challenge-response* protocol is commonly based on symmetric encryption. While symmetric encryption is used to provide confidentiality, it can also be used to authenticate parties that already have a shared key. For example, if Alice and Bob are sharing K_{AB}, Alice can prove she is talking to Bob by challenging him with a random number:

1. Hey, this is Alice. Am I talking to Bob? If so, encrypt this random number: 39281838001.
2. Yes, this is Bob. I'll prove it by encrypting {39281838001}K_{AB}.

[3] A hash value is the output numerical value of a hash function that is dependent on the input data and the corresponding hash function used to generate the hash. In cryptography hashes are often used for compression functions, contraction functions, message digests, fingerprinting, cryptographic checksums, message integrity checks, and manipulation detection codes. Schneier at 30.

66. This protocol, in various forms, has existed since the 1970's and is still used today. *See* Encryption-Based Protection Protocols for Interactive User-Computer Communication by Stephen Thomas Kent (1976). Alice, in this example, verifies Bob's answer by either decrypting the response or by computing her own encrypted version and comparing it. HAC at pp. 401-402; US Patent 6,075,860 to Ketcham at 9:57-10:9; Techniques for Privacy and Authentication in Personal Communication Systems at p. 8. Because Alice can pre-compute the encrypted version while awaiting Bob's answer, this approach is faster.

67. Accordingly, the elements of the claims dealing with the signals generated from a common secret are the well-known concept of challenge-response. To illustrate, here is representative theme from the claims of the '977 and '564 patents mapped to the challenge-response protocol:

1. (**challenge**) provide a first signal to the second device;
2. (**response**) receive a second signal from the second device;
3. (**pre-compute comparison value**) compute a local modified first signal according to the secret;
4. (**validate response**) determine whether the second signal is derived from a secret known by the first device, possibly by comparing to a local modified first signal.

5. Certificates

68. One of the most critical issues with public key cryptography is knowing that the public key is actually the public key associated with the party. For example, Alice and Bob could *both* claim to be Alice.

1. ALICE: Attention World, I'm Alice and here is my public key K0.
2. BOB: Attention World, I'm Alice and here is my public key K1.

69. Without additional information, simply receiving a public key is not enough to authenticate the sending party. Usually, the public key is combined with additional data.

70. One step toward authenticating a party and their public key is binding the public key to identification data about the party. This is commonly done using a *public key certificate*, also known as a *digital certificate*, an *identity certificate*, or often just a *certificate*[4]. Typically, a certificate is a labeled collection of data about the public key's owner (called the "subject") and the public key itself. It will include the subject's identity (e.g., "Alice" in our example or "www.amazon.com"

[4] There are other types of certificates, but these are so common that when industry professionals use the word "certificate" they are often referring to a public key certificate.

for Amazon's web server in real life), information about when the certificate is valid, and so forth. Schneier at p. 575.

71. To ensure that none of the data can be changed, the public key and the identification data are *signed* and the signature is included within the certificate. But what key is used to sign the certificate? Before answering that question, let's assume that Alice and Bob both have certificates.

1. ALICE: Attention World, I'm Alice and here is my certificate with my public key.
2. BOB: Attention World, I'm Alice and here is my certificate with my public key.

72. This is a step in the right direction because if Alice's certificate says the subject (owner) is "Alice" and Bob's certificate says the subject is "Bob" then the World can know that Bob's claim to be Alice is false.

73. The problem is, a certificate is just data. Anybody can create data. Bob can create a certificate with the subject "Alice" or any other subject name he prefers. For this reason, *receiving a certificate is not enough to authenticate the transmitting party*.

74. Two commonly used methods to authenticate another party's certificate are:

1. Receiving the party's certificate through some kind of inherently trusted channel such as a real-life, in-person meeting.
2. Or, trusting a certificate based on transmission or confirmation by another already-trusted party.

75. It is, of course, impractical for everybody in the world to meet every party with a certificate in person. Instead, our world's authentication mechanisms are built on *chains of trust*.

76. As I explained, certificates are *signed* to prevent tampering. But the signature is also used to chain the certificate to another certificate. It works like this:

1. ALICE: I'm Alice and here is my certificate.
2. WORLD: Ok, the certificate's subject is "Alice." But how do we know you didn't make up this certificate?
3. ALICE: Do you already trust Charlie?
4. WORLD: Yes.
5. ALICE: My certificate was signed by Charlie.

77. The World verifies the signature using Charlie's public key from Charlie's certificate, *which they already had and already trusted*. Because Alice's certificate was signed by Charlie, the two certificates are chained together. The

World can be assured that Alice's certificate is valid so long as Charlie can be trusted. Schneier at pp. 574-577.

78. But, despite all of this, a validated certificate *still* is not enough to authenticate a party. The reason is the certificate is public, meaning anyone has access to it. Anyone that gets a copy of the certificate can give it to anyone else.

1. ALICE: Hello, Bob, I'm Alice and here is my certificate.
2. BOB: Thanks Alice.
3. BOB: Hey Charlie, I'm Alice and here is my certificate.

79. Certificate validation has been used in TLS (originally known as SSL) since at least 1995. See "The SSL Protocol" by K. Hickman (https://tools.ietf.org/html/draft-hickman-netscape-ssl-00). In TLS, if a user navigates to Amazon's website and receives the certificate, what prevents them from turning around and using that certificate to claim that they are Amazon?

80. The final step in authentication is to use an authentication protocol, such as a challenge-response protocol or an authenticated key-agreement/key-transport protocol, to authenticate that the party that transmitted the certificate is also the owner.

81. Both steps are necessary. To summarize authentication with a certificate:

1. The party receiving the certificate must determine whether or not the certificate can be trusted.
2. The party receiving the certificate must, after validating the certificate, use an authentication protocol to confirm that the transmitting party is the certificate's owner.

82. Returning to the Amazon example, when a user's browser contacts Amazon's website, the website transmits a certificate. The browser first validates the certificate. Next, it either agrees on a session key using an authenticated key agreement protocol or it transports a session key back to Amazon's website using an authenticated key transport protocol. Once the browser is able to communicate with Amazon's website using this session key, it knows that it must be communicating with Amazon and that it has authenticated Amazon's identity.

83. In summary, challenge-response protocols are widely known in the art. Here is a very short list of some existing references.

84. <u>Handbook of Applied Cryptography (HAC - Ex. 1009):</u> This very well-known reference, published in 1997, sets for a wide range of symmetric and asymmetric challenge-response protocols.

85. <u>Applied Cryptography (excerpts reproduced in Ex. 1008):</u> Written by security expert Bruce Schneier, this book (second edition published in 1996) discusses challenge response protocols along with other major cryptographic

techniques, such as symmetric algorithms, asymmetric algorithms, certificates, and more.

86. Techniques for Privacy and Authentication in Personal Communication Systems: The GSM communication standard is a well-known standard for telecommunications (and is a predecessor network to modern systems). GSM authentication involves comparing an encrypted response value to a pre-calculated value. See also, e.g., US Patent 6,075,860 to Ketcham (1997).

87. SSL/TLS: The TLS protocol, formerly known as SSL, has been protecting Internet communications since 1995. It is possibly the most widely used security protocol in existence. It makes use of challenge-response protocols, asymmetric algorithms, symmetric algorithms, and certificates.

88. IPR References: ***Maillard*** (1999) discloses using certificates for identification, asymmetric algorithms, symmetric algorithms, and challenge-response protocols. ***Davis*** (1997) also discusses the use of digital certificates and challenge-response protocols.

6. Time Based Locality to Determine Proximity of Devices

89. A known problem with authentication protocols is that of a *replay attack*. In this type of attack, an eavesdropper records the traffic between authorized parties in order to re-use it later in an un-authorized manner. For

example, in the challenge-response protocol I described, the challenge must be a large enough random number in order to be reasonably secure.

90. Otherwise, imagine if the random number challenge was no more than four digits (i.e., between 0 and 9999). An eavesdropper recording the challenge-response protocol between two parties could record one or more challenges, and the associated responses, and then attempt to authenticate repeatedly until one of the previously used challenge numbers is accidentally recycled.

91. Typically, this is fixed by using a sufficiently large challenge number that is unlikely to repeat. But some protocols, such as the widely used Kerberos protocol, use *timestamps* to determine if a response is *fresh*. Kerberos, for example, has a default window of five minutes. Messages older than the predetermined allowed window of time are not considered valid.

92. The Kerberos five-minute window is not intended to bound the distance between devices. Under most circumstances, this timeframe is more than sufficient for communications between devices anywhere in the world. In other protocols, however, the time window may be significantly shorter and more closely tied to the nature of the communications.

93. At least as of early 1990's several publications started to address distance-bounding protocols that took proximity of the devices into account. In the context of the '977 and '564 patents, a "distance-bounding protocol" is a

mechanism for restricting the transfer of content from one device to only those other devices that are within a limited distance. In some distance-bounding technologies transmission delay is used to approximate distances between two devices. When a device transmits data to another device and subsequently receives a response, the time it takes to transmit the request and receive the response is known as the *round-trip time* or RTT. When data is transmitted electronically, under even the most ideal circumstances, the signal speed is bounded by the speed-of-light. Using RTT a device can calculate an upper bound on the distance between the devices. This concept was *well known* within the prior art and derived from standard cryptographic principles.

94. The ’977 and ’564 patents themselves admit that “distance-bounding protocols with public-key identification schemes” were known in the prior art for at least a decade before the priority date of the patent: “[i]n the article by Stefan Brands and David Chaum, ‘Distance-Bounding protocols’, Eurocrypt ’93 (1993), Pages 344-359, integration of distance-bounding protocols with public-key identification schemes is described. Here distance measurement is described based on time measurement using challenge and response bits and with the use of a commitment protocol.” ’977 patent at 2:29-36.

95. Brands and Chaum: This paper by Brands and Chaum set forth the need for their research in the abstract. “It is *often* the case in applications of

cryptographic protocols that one party would like to determine a practical upper-bound on the *physical distance to the other party*." Distance-Bounding Protocols at Abstract, emphasis added. The paper explains that a distance-bounding protocol can be integrated with an authentication ("identification") algorithm: "[o]ur distance-bounding protocols can be integrated with known public-key identification schemes…" Distance-Bounding Protocols at Introduction.

96. Ross Anderson: Mr. Anderson, a well-known security researcher, discussed this concept when describing the "Identification Friend or Foe" ("IFF") protocol between airplanes and anti-aircraft ground installations. In the 2001 edition of his book, "Security Engineering," he stated:

> This is a much-simplified account of IFF but it serves to illustrate the different trust assumptions that underlie an authentication protocol. If you send out a challenge N and receive, within 20 milliseconds, a response {N}K, then, because light can travel a bit under 3,730 miles in 20 ms, you know that there is someone with the key K within 2,000 miles.
>
> (Security Engineering at 21).

97. Anderson's discussion of the distance between the plane and ground installation was in the context of "reflection attacks" wherein the legitimate response from one party is replayed (or reflected) by an unauthorized party. The implication here was that if a response took too long, you knew that it was not a plane within the authorized bounded distance.

98. Kindberg: In 2002, Hewlett Packard researchers Kindberg, Zhang, and Shankar, presented their research paper "Context Authentication Using Constrained Channels" at the Fourth IEEE Workshop on Mobile Computing Systems and Applications held June 20-21, 2002. Their fundamental research idea is summarized as follows:

> This paper presents a paradigm shift from conventional authentication—of a principal's identity—to authentication of parameters that characterise a principal's context. *Location*, in particular, is a highly significant contextual parameter. It is one that features in what are known as mobile, ubiquitous, pervasive and nomadic computing systems. We present a model of context authentication based on the characteristics of communication channels. As an example, *we present protocols for location authentication that are based on physical channel characteristics*.
>
> Context Authentication Using Constrained Channels at Abstract, emphasis added.

99. Kindberg and his associates explained, as did Ross Anderson, that measurements based on the speed-of-light enables a device to calculate an approximate distance to another based on round-trip communication times. However, Kindberg goes beyond Anderson's disclosure by explicitly tying this measurement to authentication protocols.

> One way of establishing location information is to use network time-of-flight. In principle, with sufficiently accurate instrumentation and knowledge of real-time system parameters, we can use ***round-trip times to bound the location of a network node.*** To gauge a bound on the distance to node *M*, node *N* sends a 1-bit message to it, which *M* is to return to N immediately. If the speed of signal propagation is *c* then, if *M* can return the message to node *N* in time $t < (l + 2d/c)$, it is

> within a distance *d* of *N*, where *l* is the total communication latency imposed by software and hardware.
> (Context at 3, emphasis added).
>
> The basic idea of our approach to authenticating a principal's location is to employ a challenge-response protocol. The authenticator can choose a nonce and either ask the principal in question to send it back to the authenticator over a send-constrained channel; or send the nonce to the principal in question over a receive-constrained channel and check if the principal has received it.
>
> (Context at 5).

100. Caswell: Kindberg also cited to and expanded upon another research paper from the year 2000 entitled, "Creating Web Representations for Places" by Deborah Caswell & Philippe Debaty. This reference, while not using the RTT technique, explained the motivations behind *why* authentication should be combined with location. "It is valuable for a place to be able to *authenticate a person/device's presence in the place*. For purposes of *user convenience, revenue protection, and place resource protection*, there are times when *services should only be offered to those who are physically present in a place*, and withhold those services from those viewing the place's web representation remotely." Creating Web Representations at Place-Specific Requirements, 115-16, emphasis added.

101. Caswell goes on to give the following example use-case: "[a]nother example is when supervision of *place resources* is needed. If a guest comes into Carol's office and she wants to share the contents of her file on the guest's PDA, *she might not want the guest to be able to save or print the file*. She wants to

maintain ownership. She also wants access to the file to go away once the guest leaves her office, where she can no longer supervise their use." Creating Web Representations at Protect Revenues, 122-23, emphasis added.

102. Noble and Corner: Other researchers, working from this same motivation, proposed systems that used authentication along with RTT to enforce distance bounding. For example, researchers Noble and Corner from the University of Michigan, used RTT measurements to ensure that two devices were physically close. Their research was directed toward physical security fobs that unlock laptops or other portable devices. U.S. Patent No. 7,302,571 to Noble et al. (provisional application filed in 2001), disclosed using RTT measurements to ensure that the fob is physically close to the laptop.

103. More specifically, the '571 patent disclosed that the laptop sends the fob authentication challenges in an on-going fashion to ensure that the fob is still present.

> There are two reasons why a laptop might not receive a response from the token. The user could truly be away, or the link may have dropped a packet. The invention recovers from the latter to avoid imposing a performance penalty on a still-present user. To accomplish this, the invention *makes use of the expected round-trip time between the laptop and the token*. Unlike wide-area round-trips, this time is known and relatively stable. The invention retries key requests if responses are not received within twice this round-trip number…
>
> If there is still no response, the user is declared absent, and the file system is secured.

('571 patent at 9:60-10:2; 10:7-8, emphasis added).

104. The '571 patent taught using RTT with challenge-response authentication protocols in order to ensure that two devices are sufficiently close in an authenticated manner.

105. <u>Saito</u>: As taught by Saito et al. in U.S. Patent Publication 2003/0145214A (and corresponding Japanese patent applications filed in January 2002), distance-bounding authentication mechanisms are valuable in protecting audio and video transmissions. Saito explained that the DTCP protocol could be modified with distance-bounding technology to prevent media access at an unauthorized distance: "In other words, if the case of carrying out the AKE [Authentication and Key Exchange of DTCP] procedure by using the TCP/IP packets is considered, it would become possible to exchange the AKE packets, between neighboring homes, over a long distance, or across the national border (because the TCP/IP packets can be exchanged in such a manner), and *there can be cases where the transfer (including copy) of the AV data becomes possible over a range that exceeds a range of the private use according to the Japanese copyright law article 30, for example*." '214 Patent Publication at [0060], emphasis added.

106. <u>IPR References</u>: These concepts as well as the motivation to combine authentication protocols with RTT-based distance-bounding protocols were already well known, and the specific limitations in the claims of the '977 and '564 Patents

described by ***Davis***, ***Lundkvist*** and ***Chaum*** prior art references that I further explain in my declaration.

D. The Level of Ordinary Skill in the Art

107. I understand from TCL's counsel that the claims and specification must be read and construed though the eyes of a person of ordinary skill in the art ("POSITA") at the time of the priority date of the claims. I have also been advised that to determine the appropriate level of ordinary skill in the art, the following factors may be considered: (a) the types of problems encountered by those working in the field and prior art solutions thereto; (b) the sophistication of the technology in question and the rapidity with which innovations occur in the field; (c) the educational level of active workers in the field; and (d) the educational level of the inventor.

108. The '977 and '564 Patents relate to protocols for secure authentication of devices prior to exchange of protected content among those devices. Such systems and techniques would be familiar to persons having studied computer science and/or electrical engineering or having some level of experience designing communication security protocols.

109. Based on the above considerations and factors, it is my opinion that a person of ordinary skill in the art at the time of the alleged invention would have had at least a bachelor's degree in Computer Science, Computer Engineering, or

Electrical Engineering and 1-2 years of industry or research experience in the area of secure cryptographic communications. This description is approximate and additional education experience could make up for less work experience and vice versa.

110. I am a person of at least ordinary skill in the art. As shown by my qualifications and my curriculum vitae attached as Appendix A, I am aware of the knowledge and skill possessed by a person of ordinary skill in the art at the time of the alleged invention claimed by the ’977 and ’564 Patents. Additionally, in the course of writing books about cryptography, co-founding a project dedicated to correct cryptography, and a wide range of consulting engagements related to cryptography (including cryptography related to media, DRM, and content access protocols) I have also learned about the level of skill in the art at the time of the alleged invention. In performing my analysis, I have applied the standard set forth above.

VI. SUMMARY OF OPINIONS, IDENTIFICATION OF THE PRIOR ART, AND A POSITA’S REASONS TO COMBINE THE PRIOR ART

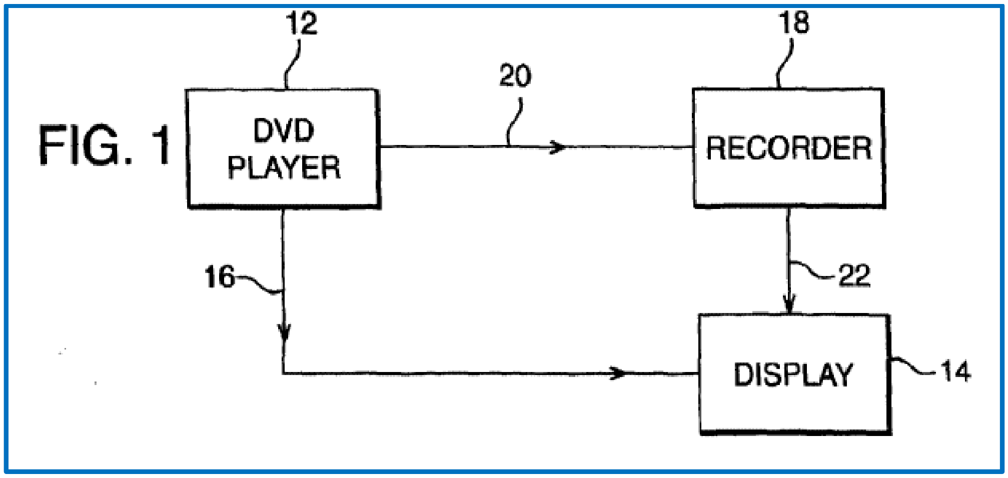
111. It is my opinion that the following prior art references discussed in my declaration disclose all technical features in the challenged claims of the ‘977 and ’564 Patents and render them obvious: European Patent Application Publication EP 1045585 A1 to Maillard et al. (“Maillard,” Ex. 1004), PCT Patent Application Publication No. WO 97/39553 A1 to Davis et al. (“Davis,” Ex. 1005), PCT Patent

Application Publication No. WO 02/35036 to Lundkvist ("Lundkvist," Ex. 1006), U.S. Patent No. 4,926,480 to Chaum ("Chaum," Ex. 1010) and other references that I have identified in my declaration.

112. In addition to providing summaries of each of the prior art references below, I will also explain the reasons a POSITA would have had for combining the prior art references. I understand, however, that a POSITA would need to have had a reason to combine these pieces of prior art with a reasonable expectation of success. It is my opinion that there are a multitude of reasons why a POSITA would have been motivated to combine the above prior art references at the time of the priority date of the '977 and '564 Patents.

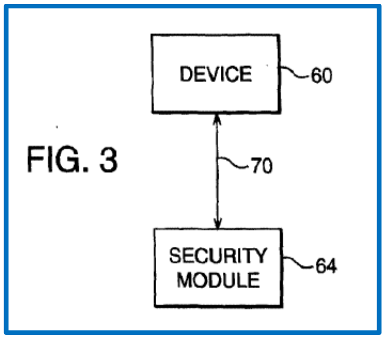
A. Summary of Maillard

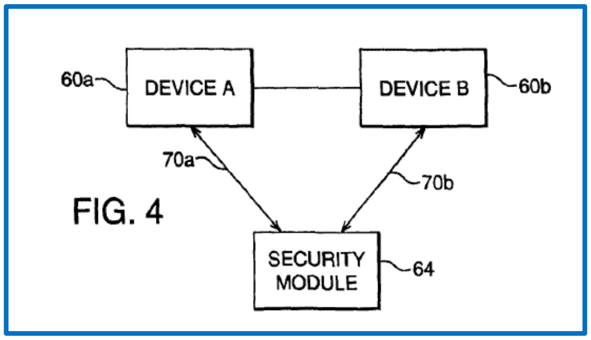
113. Maillard describes methods for providing a secure communication of digital data between devices to prevent illegal copying and distribution of a digitally recorded data. Maillard at Abstract, [0001]. As an example of a content transfer system, Maillard describes a DVD player in its Figure 1 (see below) that provides content to other devices (e.g., a TV and/or a DVD recorder). Maillard at [0074]-[0076]. "Whilst the elements of player 12, display 14 and recorder 18 have been indicated separately, it is conceivable that some or all of these elements may be merged, for example, to provide a combined player/television set." Maillard at [0077].



114. In order to transmit content, Maillard establishes secure communications. As part of this establishment, Maillard requires one or more of the devices to be validated (Maillard at [0078]), where "[a] preferred device validation system is based on the *transfer of certificates* between a device and a security module." Maillard at [0079], emphasis added. see also Maillard at [0088].

115. Maillard's security module (see Figures 3 and 4 below) can be a smartcard but can also be part of one of the devices, e.g., inserted into a socket of the device, e.g., through a PCMCIA connection. Maillard at [0008], [0011].

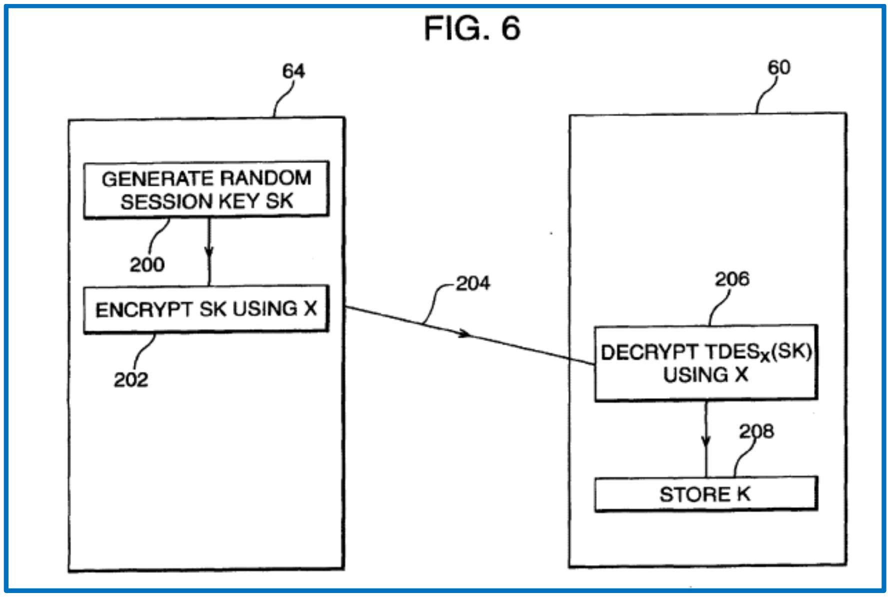




116. Figure 5 further details Maillard's validation process which eventually identifies the receiving device's public key as permitted and/or not excluded; upon

a successful verification, the security module transmits a "session key" to the receiving device as shown in FIG. 6. Maillard at [0104]-[0108].

FIG. 6

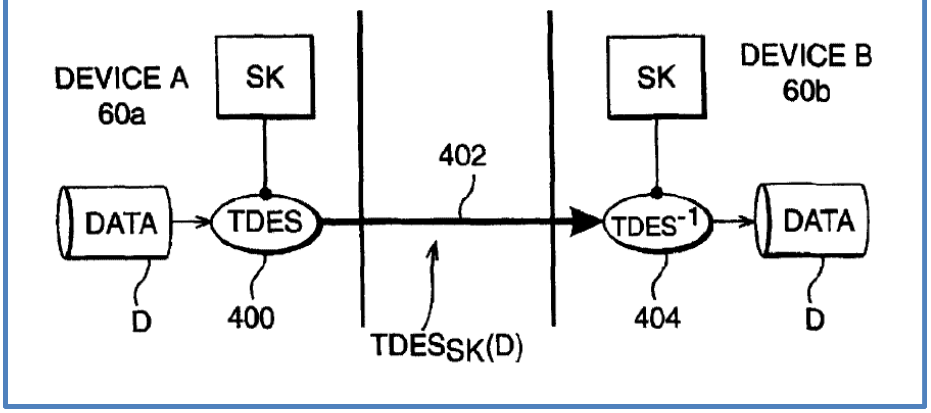


117. In more detail, the security module generates a random session key SK, and encrypts it using a random number X that was previously shared in a secure manner between the devices. Maillard at [0094]-[0096], [00105]. The encrypted session key is to the device 60, which decrypts the received message using X, and stores the session key SK in memory. *Id.* Maillard also describes the session key can be updated at any time including when the device is tuned on, upon establishment of connection with the security module, etc. Maillard at [0052], [0107], claim 37.

118. Maillard's session key SK is subsequently used to protect content by encrypting data that is transferred to the device 60. Maillard at [0105]-[0107]. This process is illustrated in FIG. 9 of Maillard (below) wherein the data is

encrypted using the session key, SK, via Triple DES encryption algorithm by a transmitting device and set to a receiving device, which uses the same session key, SK, to decrypt and obtain the data. Maillard at [0123]-[0124], FIG. 9.

FIG. 9



119. In summary, Maillard describes operations for authenticating a device and sharing a common secret as a prior step before transferring protected content from one device to another device.

B. Summary of Davis

120. Davis relates to "a wireless authentication system which mitigates the likelihood of unauthorized use of an electronic device through periodic challenge/response messages." Davis at 1:17-21.

121. Davis observes that smart cards are used for authentication purposes but can be vulnerable when "the user accesses his or her personal computer and leaves the personal computer unattended for some duration without removing the card or disabling the personal computer during his or her absence." Davis at 2:28-

3:19. Accordingly, Davis teaches a two-part authentication that (1) ensures proximity of the user to the node by requiring the response to be received within a prescribed period of time that corresponds to a distal range from the node and (2) is based on (the well-known concept of) a challenge-response protocol. Davis at Abstract, 4:1-19, 11:8-13:16, FIGS. 5 and 6.

122. Davis describes both components of this solution and how these elements work together. Davis at 11:8-12:9, FIG. 5. By way of illustration, Davis states, "… a wireless authentication system to control an operating state of a first node (e.g., computer) based on the proximity of an authorized user to the first node. The wireless authentication system comprises a security device implemented within the first node and a user authentication token in possession of the authorized user (e.g., worn, carried, etc.). The security device ***generates a Challenge message and transmits the same to the token. In response, the token generates and transmits a Response message to the security device if the token is within a predetermined distance from the security device***." Davis at 4:2-11, emphasis added.

123. In the annotated Figure 5 (below) the red and blue illustrate Davis' time/distance measurement and challenge-response operations, respectively.

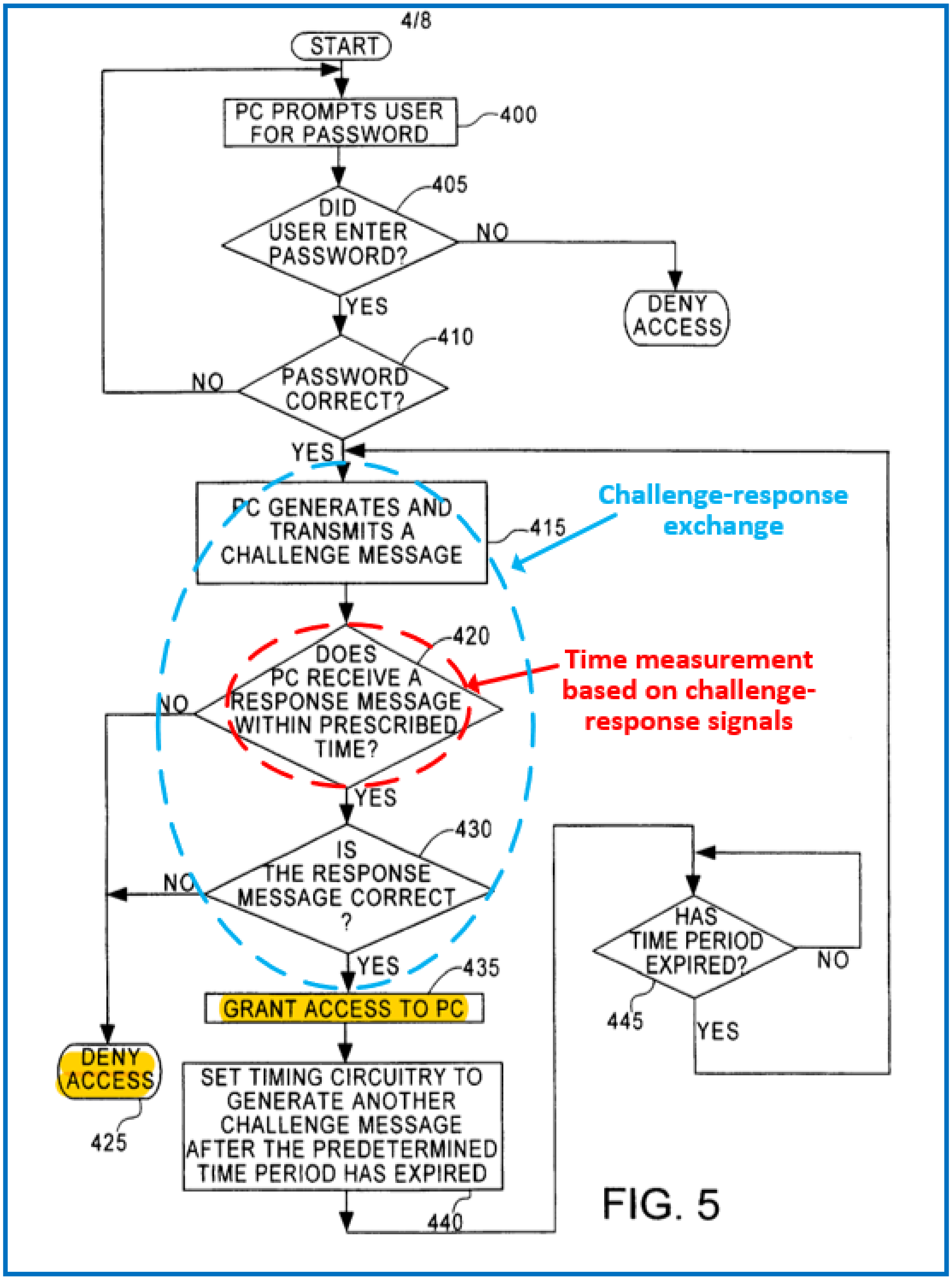

4/8
START
PC PROMPTS USER FOR PASSWORD
400
DID USER ENTER PASSWORD?
405
NO
DENY ACCESS
YES
PASSWORD CORRECT?
410
NO
YES
PC GENERATES AND TRANSMITS A CHALLENGE MESSAGE
415
Challenge-response exchange
DOES PC RECEIVE A RESPONSE MESSAGE WITHIN PRESCRIBED TIME?
420
Time measurement based on challenge-response signals
NO
YES
IS THE RESPONSE MESSAGE CORRECT ?
430
NO
YES
435
GRANT ACCESS TO PC
DENY ACCESS
425
SET TIMING CIRCUITRY TO GENERATE ANOTHER CHALLENGE MESSAGE AFTER THE PREDETERMINED TIME PERIOD HAS EXPIRED
440
HAS TIME PERIOD EXPIRED?
NO
445
YES

FIG. 5

124. As highlighted in yellow in annotated Figure 5 (above), after determining that the correct response is received within a designated time, the user

"is granted access to the contents (i.e., data, applications and other information stored thereon) of the personal computer 110 as well as its networked resources." Davis at 7:22-26.

125. Davis also describes a number of challenge-response protocols that are also used to de before allowing access to the contents of a computer. Davis at 12:10-13:16, FIGS. 6A-6C.

126. Furthermore, Davis discloses various hardware embodiments for the invention. In Figure 1, Davis illustrates an example system for implementing its secure communications between a personal computer and a user authentication token worn by an authorized user. Davis at 5:5-9, 6:28-7:19; FIG. 1; see also 5:10-15; 8:311:7; FIGS. 2-4.

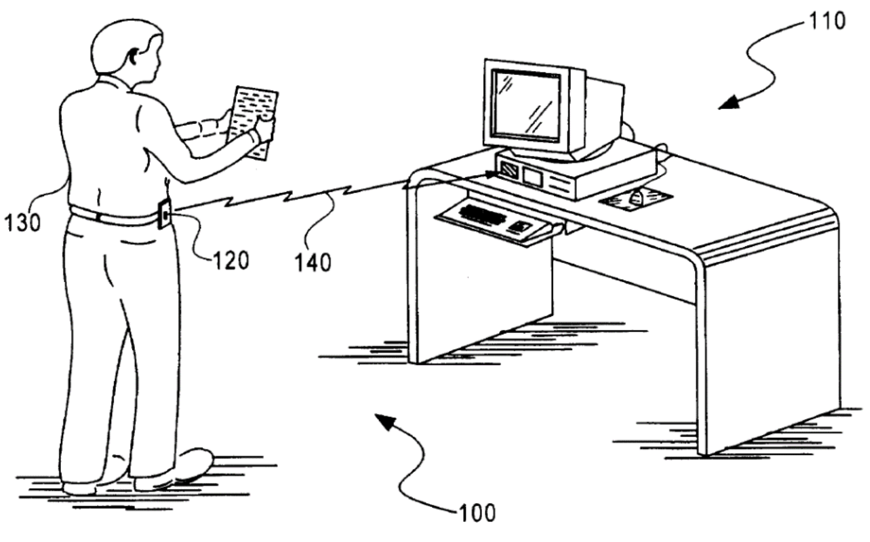
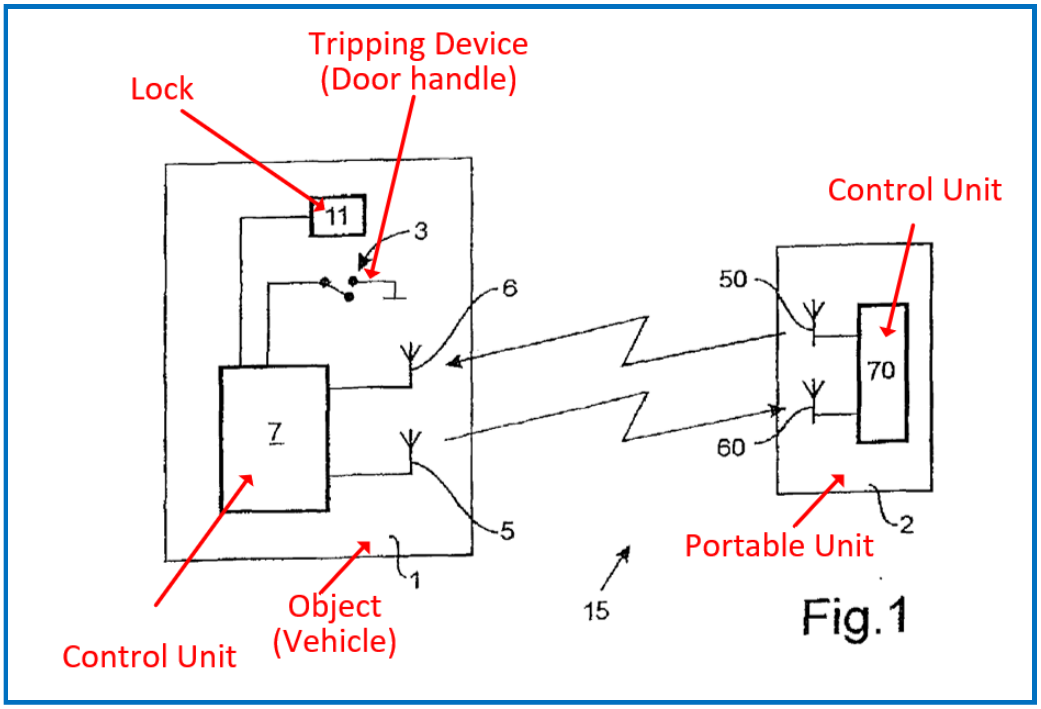


FIG. 1

127. Davis discloses that the token can be part of an electronic device, such as a pager or a cellular phone (Davis at 7:1-19), and envisions additional applications for its authentication system including in mass storage devices and electric door locks (*id*.), including locking mechanisms for car doors. Davis at 11:8-12.

C. Summary of Lundkvist

128. Lundkvist describes a method for controlling authorization for access to an object that is based on a challenge-response protocol using signals that are communicated between the object (i.e., the control unit of the object) and a wireless portable unit. Lundkvist at Abstract. Lundkvist describes the preferred embodiment of its invention "for authorization control for a vehicle, such as a car or truck," but emphasizes that this example is "in no way limiting [the] application of the invention." Lundkvist at 1:23-25.



129. Figure 1 of Lundkvist (annotated above) schematically shows the communications between an object (such as a vehicle) and a portable unit (e.g., something like a smart card). Lundkvist at 6:20-31. When the vehicle door handle (tripping device) is actuated, the control unit of the vehicle begins the challenge-response procedure with the control unit of the portable unit. Lundkvist at 8:10-17.

130. Figure 2 of Lundkvist illustrates one embodiment in which a challenge-response protocol is used to unlock a vehicle's door. Lundkvist at 8:4-9. These operations provide authorized access if the challenge-response operations are successfully verified within a predetermined round-trip time (RTT). Lundkvist at 8:24-28.

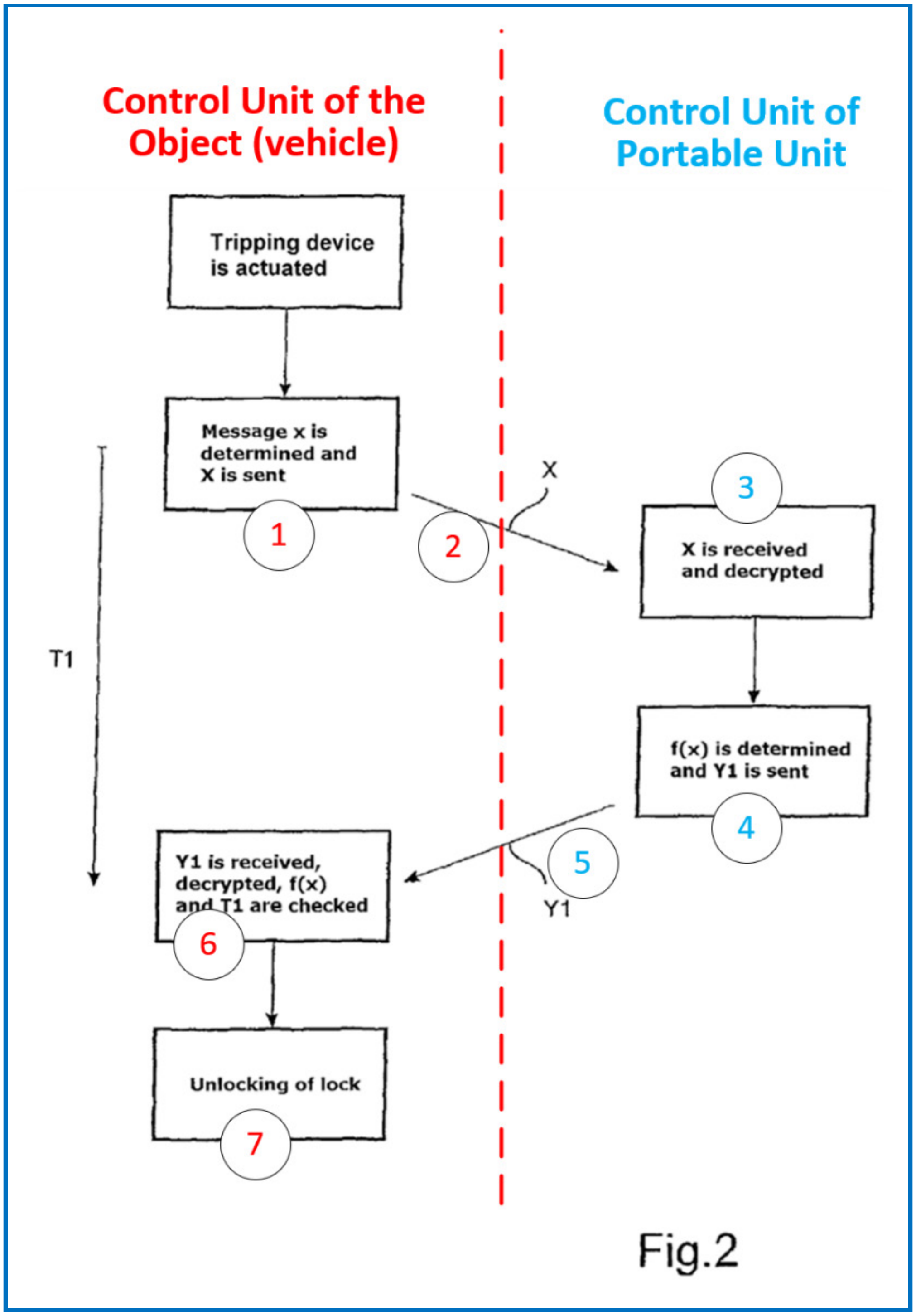

Control Unit of the Object (vehicle)
Control Unit of Portable Unit
Tripping device is actuated
Message x is determined and X is sent
1
2
X
3
X is received and decrypted
T1
f(x) is determined and Y1 is sent
4
5
Y1
Y1 is received, decrypted, f(x) and T1 are checked
6
Unlocking of lock
7

Fig.2

131. In Lundkvist, after the tripping device (e.g., door handle) is activated, a series of signals are produced and exchanged between the object (e.g., car) and the portable unit to determine whether the door should be unlocked. These signals

and exchanges are described below, and illustrated with the aid of annotated Figure 2, where the numbers in circles correspond to the operations described below:

1. The control unit of the object (e.g., car) creates a message that comprises first information **x**. Lundkvist at 8:11-16.
    i. The first information **x** consists of identity information, **O_ID**, that is unique to the object and a random number, **O_RND**, generated by the control unit 7 of the object. This message is encrypted to produce **X** (upper case). *Id*.
2. The signal **X** is transmitted to the portable unit. Lundkvist at 8:16-17.
3. The portable unit receives and decrypts the message to obtain the first information **x**. Lundkvist at 8:19-20.
4. The portable unit processes the first information **x** (using function f(x)), and encrypts it to produce a second signal **Y1**. Lundkvist at 8:19-23.
    i. Specifically, in this step, the "signal **Y1** comprises the first information **x** in processed form, more specifically a function **f(x)** of the first information **x**. In particular, **f(x)** comprises the message part **E_SVAR = f(O_RND)**." Lundkvist at 8:21-23.

5. The portable unit transmits **Y1** to the object. Lundkvist at 8:19-24.
6. The control unit of the object receives and decrypts the second signal **Y1**. Lundkvist at 8:23-24). It then performs two actions:
    i. Measures the round trip time **T1**, which is the time from transmission of the first signal **X** to reception of the second signal **Y1**. Lundkvist at 8:24-26.
    ii. Determines if the received **E_SVAR** is equal to **f(O_RND)**. Lundkvist at 8:24-28.
7. If **T1** is less than a predetermined value, and if the received **E_SVAR** is equal to **f**(**O_RND**), the door is unlocked. *Id.*

132. Therefore, Lundkvist describes a challenge-response protocol used for conducing distance measurement based on timing measurements of the transmitted and received signals.

D. Summary of Chaum[5]

133. The Chaum patent "relates to secure transaction systems, and more specifically to configurations and cryptographic techniques for transactions between two subsystems moderated by a third subsystem." Chaum at 1:17-20.

[5] Chaum is used as part of invalidity grounds for only the challenged claims of the '564 Patent, but my analysis with reference to Chaum is also applicable to the

134. Chaum, similar to Davis and Lundkvist, recognizes that challenge-response protocols are common cryptographic tools that were known in the prior art. In the relevant section, Chaum describes:

> Challenge response techniques form a basic part of many cryptographic protocols known in the art …Such a protocol might be initiated by the first party sending a random challenge to the second party, who is then to return an encryption of the challenge using the secret key. ***If conventional cryptography is used, then both parties would typically share this key, and the first party could use it to encrypt the challenge and verify that the result is identical to what was supplied by the second party. If public key digital signatures are used, then the second party would sign the challenge using its secret signing key and the first party would verify the signature using the appropriate public key***.
> Chaum at 11:11-26 (emphasis added).

135. Therefore, similar to Davis and Lundkvist, Chaum recognizes that a challenge-response procedure can be carried out using a symmetric encryption algorithm, or alternatively, using an asymmetric encryption algorithm for encrypting the response. Chaum's challenge-response alternative that uses asymmetric encryption is identical to the procedure described in Figure 6A of Davis (reproduced below and annotated). Davis at 12:10-22.

challenged claims of the '977 Patent in showing that a POSITA would have found these claims obvious.

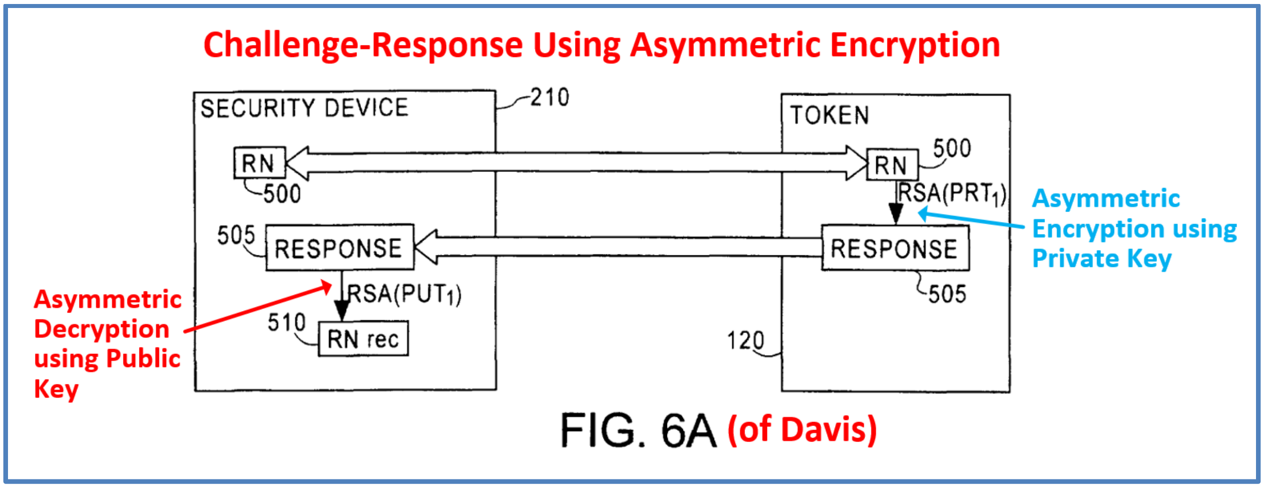
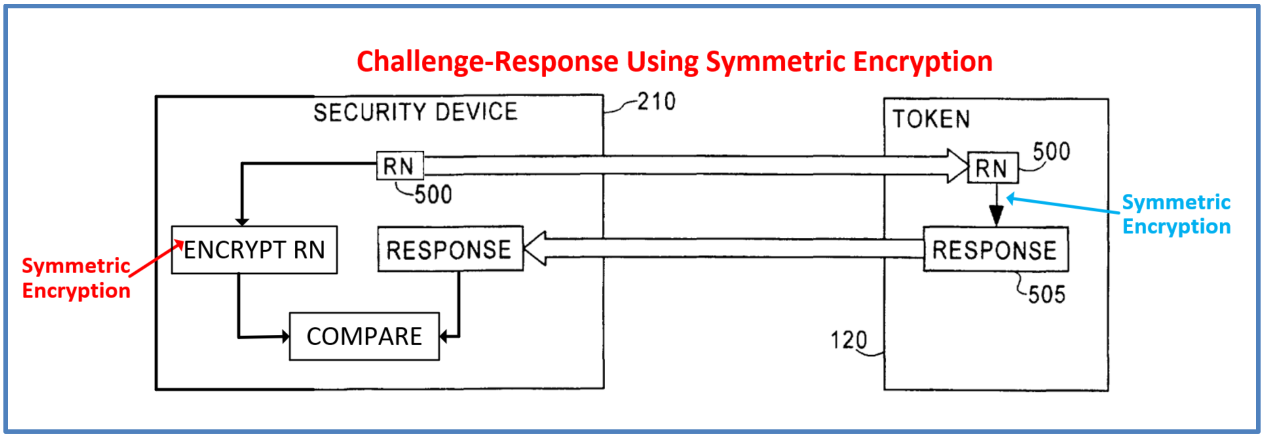

Challenge-Response Using Asymmetric Encryption
SECURITY DEVICE
210
TOKEN
RN
500
RN
500
RSA(PRT1)
Asymmetric Encryption using Private Key
505
RESPONSE
RESPONSE
505
Asymmetric Decryption using Public Key
RSA(PUT1)
510
RN rec
120

FIG. 6A (of Davis)

136. In the following ***modified*** reproduction 6A of Davis, the symmetric encryption alternative of Chaum is visualized. In this figure, a symmetric key is used to encrypt the response. As the encrypted response is predictable to the challenger (with the same symmetric key), it can pre-compute the response by performing its own encryption of RN.



E. A POSITA's Reasons To Combine Maillard, Davis and Lundkvist or Maillard, Davis, Lundkvist, and Chaum

137. In my opinion, a POSITA would have been motivated to combine Maillard, Davis and Lundkvist. As explained in greater detail below, Maillard

disclosed mechanisms for content protection through improved device identification, including in certain location-based networks. A POSITA would have found it obvious to combine Maillard, which does not say how to ensure that devices are in the expected locations, with Davis, which teaches how to ensure that devices are within an approximate distance. The POSITA would have also found it obvious to combine Davis, which identifies but does not describe certain cryptographic operations, with Chaum and Lundkvist's teachings about the same.

138. Maillard is directed toward, "providing secure communication of digital data between devices…" Maillard at Abstract. More specifically, Maillard describes how content, such as copyrighted audio and video content, can be protected and access controlled between devices. A key component in Maillard's disclosure is the "independent security module," which is generally just referred to as the "security module." Maillard explains that, an "independent security module is used to validate a device in, for example, a digital audiovisual system." Maillard at 0008.

139. The "security module" described by Maillard *can* be a physically attached component, such as a smart card or PCMIA card. Maillard at 0011. But the security module can also be connected via "communication link" including a link to multiple devices. Maillard at 0088. While Maillard does not limit the

scope of its disclosed techniques, it describes many aspects of its invention using a "home network" as a preferred embodiment. Maillard at 0053, 0058, Claim 38.

140. For example, Maillard, in discussing the benefits of "unauthorized lists" or "authorized lists" (blacklists vs whitelists), says, "the use of an authorization list can also prevent device identifiers intentionally published on the Internet from working since these identifiers will not be valid anywhere except in, for example, a home network." Maillard at 0015.

141. A POSITA, however, would recognize that while Maillard's white-list would provide the necessary step of identifying authorized devices, it does not provide a mechanism for enforcing that those devices are on the "home network," i.e., physically within the bounds of the home network. It was known to a POSITA that data signals of one communication type, such as IEEE 1394 as mentioned in Maillard, can be vulnerable to attacks, where signals could be adapted and retransmitted over more traditionally long distance mechanisms, such as Ethernet. US patent 6,496,862, with a priority date of 1998, describes a "gateway" that interconnects ethernet and IEEE 1394 specifically for media and television. A POSITA would readily comprehend that Maillard's security module for device authorization and authentication in a home network could be exfiltrated to a remote location. Thus, instead of a DVD player transmitting data to a local TV on a home network, Maillard's security module would enable the DVD player to transmit to

another unauthorized TV anywhere in the world so long as the security module is connected in this way. Maillard 0055.

142. Thus, the POSITA would have been motivated to find mechanisms for enforcing that the security module and associated devices were on the "home network." Maillard in view of Davis provides an obvious solution.

143. Davis addresses a similar content-access problem. In Davis, the content of a personal computer, a printer, or the data on a hard drive, should only be accessible when an authorized user is physically nearby. Davis at Abstract, 11:19=21, 2:28-3:30, 6:2-11, 6:28-7:26. Davis describes how using a challenge-response protocol, well known in the art, can also provide a measurement of distance. Davis at 42-19, 6:28-7:26. Thus, an authorized party must remain physically close to the content, or access to the content is lost. A POSITA would easily comprehend how this physical distance requirement could solve Maillard's "home network" enforcement problem. For example, Maillard's security module can be a "smart card" (Maillard at 0008 to 0011) and Davis explicitly discusses the strengths and weaknesses of this technology. Davis points out that the card can be inserted and authenticated but, afterward, the authorized user of the card can leave without the authorization for content being revoked. Davis 3:1-19. This is key issue in Maillard wherein the security module may be communicating over a network. Maillard at [0053].

144. Davis's disclosure relies on the use of the distance measurement to authorize access to protected content. While Davis provides concrete examples of how this can be done, Davis notes repeatedly that there are various alternatives that are not described in detail. For example, Davis makes it clear that the encryption algorithms used in the challenge-response protocol may be either public key-based, such as with RSA, or private-key (or shared-key) based, such as with DES. Davis at 6. Nevertheless, Davis only describes the use of public key approaches in detail (while emphasizing that there are many other means of authentication). Davis at 12, Figs. 6A-6C.

145. Although there are, as Davis says, "other means of authentication," within cryptography, and specifically in the realm of challenge-response, a POSITA would have understood that there were largely only two classes of encryption that can be used on a challenge and/or response, namely, symmetric encryption and asymmetric encryption. See, e.g., Section V.C. As stated above, even Davis, although not describing a symmetric encryption approach, explicitly recognized that encryption with a key is generally public/private (asymmetric) or shared (symmetric). Thus, a POSITA, in possession of Davis would have found it obvious that challenge-response based protocols symmetric encryption would provide an alternate way of encrypting and decrypting the signals involved in a challenge-response protocols. As I explained in Paragraph 58 above, symmetric

encryption would have been a faster operation. So a POSITA would have been motivated to seek the alternate approach of using symmetric operation.

146. Lundkvist and Chaum are two references that describe such approaches. Both references also relate to distance bounding and, therefore, would be looked to by the POSITA for additional guidance. Lundkvist describes a challenge-response protocol with a time measurement and also describes asymmetric cryptography as an option. Lundkvist at 3:14-25, 4:1-5. Lundkvist, also says that symmetric encryption can be used in the same protocol. Lundkvist at 7:25-8:2. In Lundkvist, the symmetrically encrypted data is decrypted. Lundkvist at 8:19.

147. Chaum describes two alternate challenge-response procedures: one that is identical to one of Davis's procedures in Figure 6A that uses asymmetric encryption, and an alternate embodiment wherein symmetrically encrypted data is compared against the expected encrypted data without encryption. Chaum at 11:19-23. A POSITA in possession of Maillard and Davis would recognize these disclosures of symmetric cryptographic designs as a limited number of alternative implementations that can be used to successfully implement the challenge. Besides, as I explained earlier, symmetric encryption would have been faster than asymmetric encryption that is exemplified in Figure 6A of Davis. So a POSITA would have had a reason to make this substitution.

148. Chaum, does identify that in some circumstances, encryption-based challenge-response ***may*** be too variable for accurate distance measurement. Specifically, Chaum notes, "Variability in the time required to compute a cryptographic function applied to a challenge may be large compared to the accuracy of distance measure required, which is one reason such techniques may not be preferred for the present problem." Chaum at 11:26-31. However, this statement by Chaum should not be misunderstood. First, Chaum did not say that algorithms might be ***too slow***,[6] but rather said that they might have too much variability. Chaum's concern about variability is also made with reference to both asymmetric and symmetric encryption, which, as previously explained, generally have very different timing characteristics. This suggests that Chaum was concerned about variability as a general problem rather than a commentary on any particular asymmetric or symmetric algorithm.

149. Most importantly, Chaum did not say that variability ***was*** a problem, only that it ***may*** be a problem. Only in circumstances ***where it is in fact discovered*** to be a problem might Chaum's additional teachings be helpful. It would be obvious to a POSITA in possession of both Davis and Lundkvist that these

[6] Even if being slow was a concern at the time of Chaum's patent in 1988, this would not have been the case in 2003 (time of alleged invention of the '564 Patent) with vastly improved processing and computational capabilities.

references disclose using symmetric algorithms for timed distance measurement and that it would be equally obvious to use Chaum's direct comparison of an encrypted value within that context.

150. A POSITA also would have found that Davis and Lundkvist could be naturally combined because Davis explained that one application of its access authorization technique was to allow access to a car, bus, etc. (Davis at 9:26-10:2, 11:8-12), which was similar to Lundkvist's working example of allowing access to a car. Lundkvist at 8:11-28.

VII. UNPATENTABILITY OF THE CHALLENGED CLAIMS OF THE '977 PATENT

A. Claim 1

151. In my opinion, claim 1 is unpatentable as obvious over Maillard, Davis and Lundkvist.

1. Element 1[Preamble]: "A receiving device[7] comprising:"

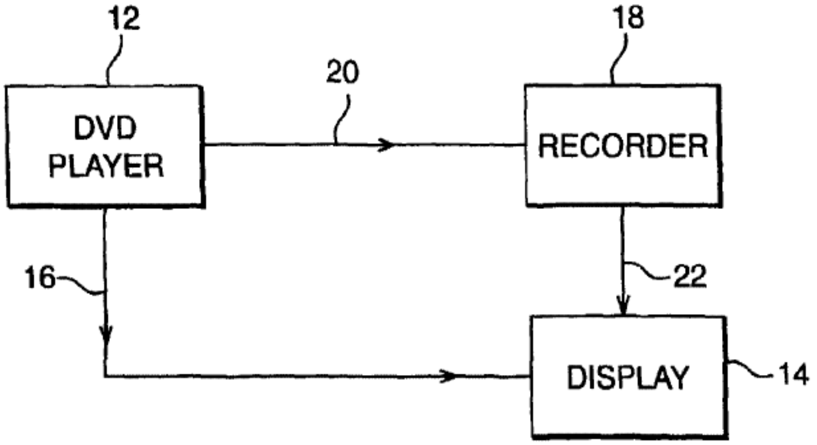
152. I understand that the preamble of a patent claim is sometimes not limiting. Nevertheless, to the extent that the preamble of Claim 1 of the '977

[7] In this declaration, I use the terms "receiving device" and "second device" interchangeably in my analysis of the claims and annotation of figures. Claims 1-10 of the '977 patent use the term "receiving device," while the remaining challenged claims of the '977 and '564 Patents refer to the same entity as the "second device."

patent is limiting, Maillard describes a "device" that, as explained more fully below, is a "receiving device" as required by Claim 1.

153. Figure 1, for example, shows an example DVD-Recorder-TV system, where the DVD provides protected audio-visual content to the recorder or the TV, with Figure 3 showing a receiving device 60 that would be the equivalent of the digital recorder or the TV that would receive the content. Maillard at 0074-0077, 0088.

FIG. 1



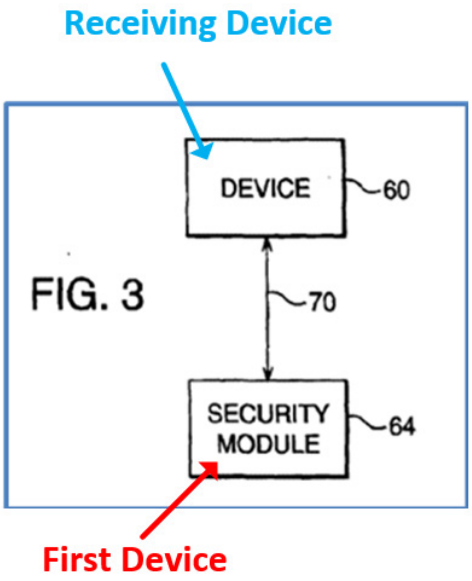


FIG. 3

Receiving Device

First Device

154. Maillard also states, “The session key SK is thereafter used to encrypt data transferred between the device 60 and the security module 64.” Maillard at [0106]. The “device 60” receives content such as media content from the security module or another device with keys from the security module. Maillard at [0114], [0117]-[0124]; see also Maillard at Fig. 9 (see also, e.g., Fig. 7).

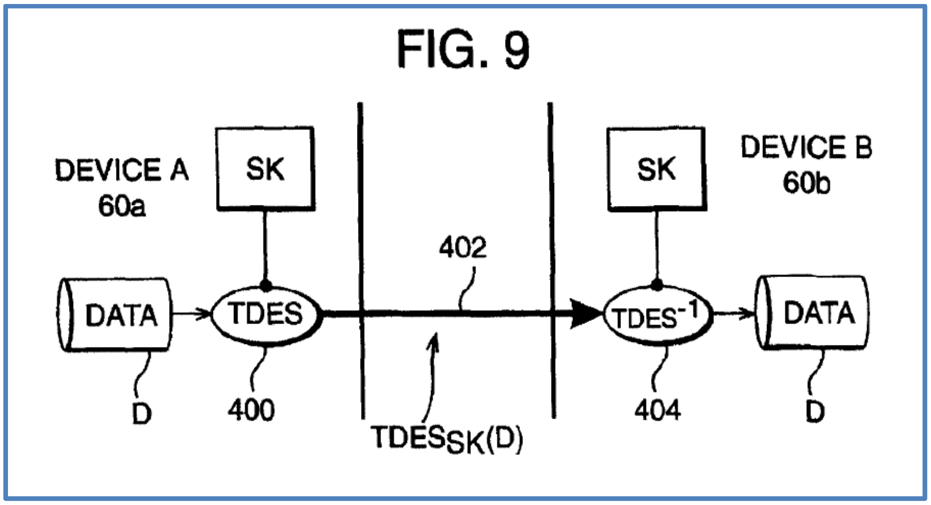


FIG. 9

2. Element 1[a]: "means for providing a certificate identifying said receiving device"

155. Maillard discloses providing a certificate identifying said receiving device. Specifically, Maillard discloses that the "device 60" provides a certificate to the security module for purposes of identifying itself. In Maillard's disclosures, the unique public key contained within the certificate is the identifier of the device.

> … each CE manufacturer 52 assigns to each of its CE devices 60 a respective encrypted certificate $\text{Cert}_{\text{CEman}}$(Device_Kpub) shown at 62. ***This certificate contains, inter alia, a unique device public key Device_Kpub,*** together with an indication of the device 25 capability (recorder, player, etc.). The certificate is encrypted by an equivalent key to the public key CEman_Kpub. To enable the contents of the certificate to be decrypted, the CE manufacturer 52 stores in the CE device the CA public key CA_Kpub and the encrypted certificate Cert_{CA}(CEman_Kpub) of the CE manufacturer 52. ***Thus, the public key Device_Kpub of the CE device 60 can serve as an identifier of the device.***"
>
> Maillard at [0084] (emphasis added).

156. I have further provided a detailed technical analysis of Maillard's certificate exchange protocol, which I have reproduced in Appendix B.

157. Maillard provides additional disclosures that indicate that the public key of the device is its identifier. For example, Maillard discloses that revocation lists identifying devices that should no longer be trusted, and authorization lists identifying pre-registered devices, identify devices by their public key:

> The security module 64 supports two types of list [sic]. A "revocation list" contains device public keys associated with invalid devices and is

> used to blacklist non-compliant devices. An "authorization list" contains device public keys associated with valid devices and is used to restrict transfer of data to between pre-registered devices only.
>
> Maillard at [0100].

158. The device's public key, contained inside the certificate CertCEman, is provided to the security module during an identification and validation operation. Maillard at [0094]-[0100].

3. Element 1[b]: means for receiving a first signal from a first device after the first device determines, based on information obtained from the certificate, that the receiving device is compliant with a set of compliance rules.

159. Maillard discloses (1) a first device determining, based on information obtained from the certificate, that the device is compliant with a set of compliance rules. After determining that the device is compliant, a POSITA would have found it obvious to combine Maillard with Davis and Lundkvist to disclose (2) receiving a first signal from the first device. NOTE: as indicated in this paragraph, I am, for clarity, analyzing this claim limitation in chronological, rather than lexical, order.

160. Specifically, Maillard discloses a security module as a first device. The security module receives the encrypted certificate from device 60 and subsequently determines that the certificate identifies the device and is compliant with a set of compliance rules.

161. In one example, Maillard discloses that the public key of the device's certificate (which is the device's identifier) is compared with a list of public keys

stored in the security module. The list is either an authorization list for pre-registered devices or a revocation list of unauthorized devices.

> Validation of the device 60 is carried out by the security module 64 using the public key Device_Kpub of the device 60 at step 128. The security module compares the received device public key Device_Kpub with a list of device public keys previously stored in the security module…
>
> The security module 64 supports two types of list [sic]. A "revocation list" contains device public keys associated with invalid devices and is used to blacklist non-compliant devices. An "authorization list" contains device public keys associated with valid devices and is used to restrict transfer of data to between pre-registered devices only.
>
> Maillard at [0099]-[0100].

162. In another example, Maillard discloses that the certificate is signed when created and the signature must validate when the certificate is received. "A signature may be included in any of the above certificates to enable the contents of the certificate to be verified following decryption of the certificate." Maillard at [0087]. See also, Maillard at [0024].

163. Both of these examples constitute determining whether or not the device's certificate is compliant with a list of rules. For clarity, the three rules identified are:

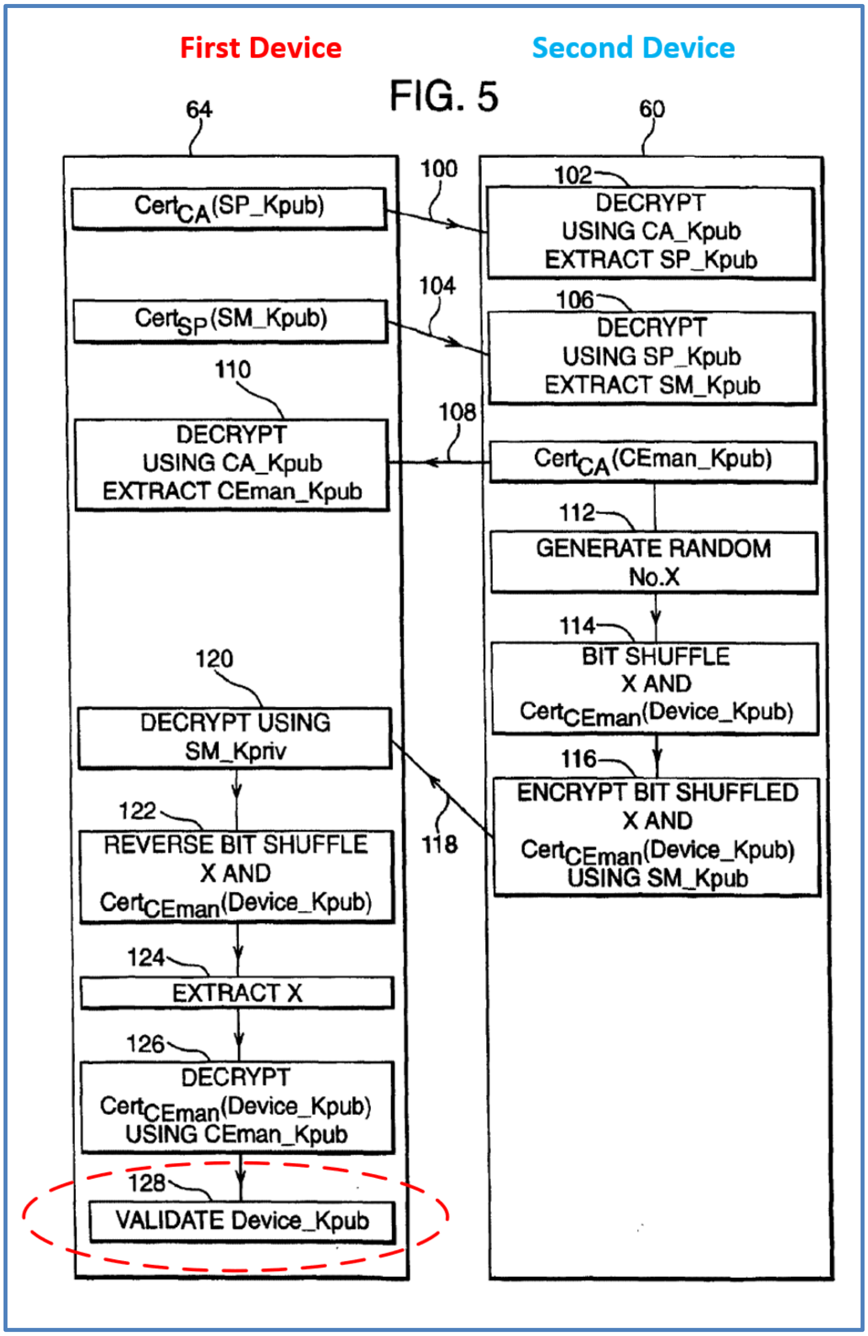
1. Certificate is not on a blacklist
2. Certificate is on a pre-authorized list
3. Certificate is signed by the proper authority

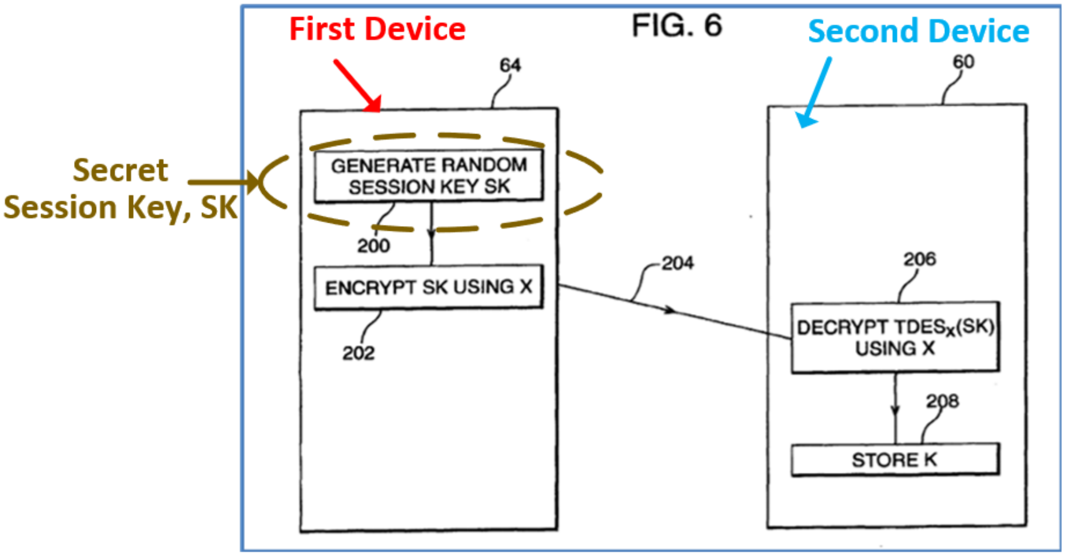
164. Once the security module ("first device") has determined that the certificate of the device 60 ("receiving device") is compliant, it (securely) transmits a session key "SK".

> If the device 60 is determined to be an invalid device, the security module 64 terminates communication with the device 60…
>
> In step 200 the security module 64 generates a random session key SK. The random session key SK is TDES encrypted at step 202 by the security module 64 using the random number X transmitted to the security module 64 by the device 60. The encrypted session key TDESX(SK) is transmitted to the device 60 at step 204.
>
> Maillard at [0103]-[0105].

165. The random number "X" is also a key, which was previously transmitted by the device 60 to the security module during the device validation. See, Maillard at [0097]. SK, encrypted by X, is only known to the security module and the device 60. Thus, it is a shared secret between them.

166. The entire process is depicted in Maillard Figure 5 (which shows the validation and exchange of X) and Figure 6 (which shows the subsequent transfer of SK encrypted by X). An annotated Figure 5 and Figure 6 are reproduced below:


First Device
Second Device
FIG. 5
64
60
100
Cert_CA(SP_Kpub)
102
DECRYPT USING CA_Kpub EXTRACT SP_Kpub
104
Cert_SP(SM_Kpub)
106
DECRYPT USING SP_Kpub EXTRACT SM_Kpub
110
108
DECRYPT USING CA_Kpub EXTRACT CEman_Kpub
Cert_CA(CEman_Kpub)
112
GENERATE RANDOM No.X
114
BIT SHUFFLE X AND Cert_CEman(Device_Kpub)
120
DECRYPT USING SM_Kpriv
116
ENCRYPT BIT SHUFFLED X AND Cert_CEman(Device_Kpub) USING SM_Kpub
122
REVERSE BIT SHUFFLE X AND Cert_CEman(Device_Kpub)
118
124
EXTRACT X
126
DECRYPT Cert_CEman(Device_Kpub) USING CEman_Kpub
128
VALIDATE Device_Kpub



167. As discussed in Section VI.E., a POSITA would have recognized that Maillard directs at least some of its teachings towards devices on a home network, which is a network where all devices are expected to be within a close physical proximity. However, the examination of certificates and other mechanisms described in Maillard do not provide enforcement for such limitations. As previously explained, a POSITA would have found it obvious to combine Maillard with teachings of Davis in order to provide this kind of validation.

168. Like Maillard, Davis also limits access to content. Davis acknowledges standard authentication approaches, such as challenge-response, which is also used by Maillard. Davis at 3:6-13, 4:13-19, 6:12-27, 10:17-11:7, 11:21-32; Maillard at [0097]. However, Davis also emphasizes that data should only be available when an authorized piece of hardware, similar to Maillard's

security module, is close to the content viewing system. Davis at 2:17-27, 4:2-12, Maillard at [0015], [0053], [0101], claim 38. Davis teaches that challenge-response protocols, already well known in the art (and used in Maillard), can be *timed* in order to estimate proximity. Davis at 4:2-19, 6:2-11, 7:11-19, 14:26-15:19.

169. In a timed challenge-response protocol, a value, usually a random value, known as a *challenge* is transmitted to a receiver. The receiver processes the challenge in a way that only the receiver should be able to do, typically using a secret key. The output of the processing is a *response* that is sent back to the sender. The sender validates that the response is correct (i.e., processed in a way that reveals knowledge of the secret key) and then calculates an approximate distance based on the response time. This calculation uses, for example, speed-of-light calculations, adjustments for expected processing time, and so forth. Davis at 2:11-13, 14:26-15:19; see also, Section V.C.

170. Davis teaches a number of methods for a timed challenge response. These methods all involve using public key cryptography. Davis at 12:10-13:22, FIG. 6A-C. Nevertheless, Davis also indicates that any appropriate authentication mechanism can be used. Davis at 12:11-13. It would have been obvious to a POSITA that challenge-response protocols typically use asymmetric or symmetric

encryption. See Section V.C. Annotated figured 6A, 6B, and 6C of Davis are reproduced below:

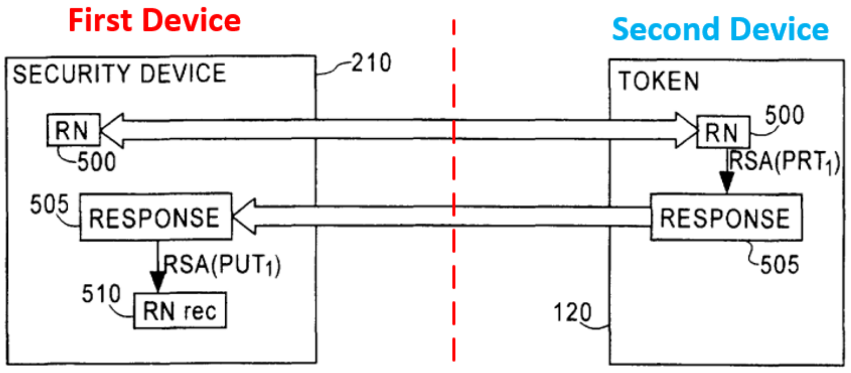


FIG. 6A

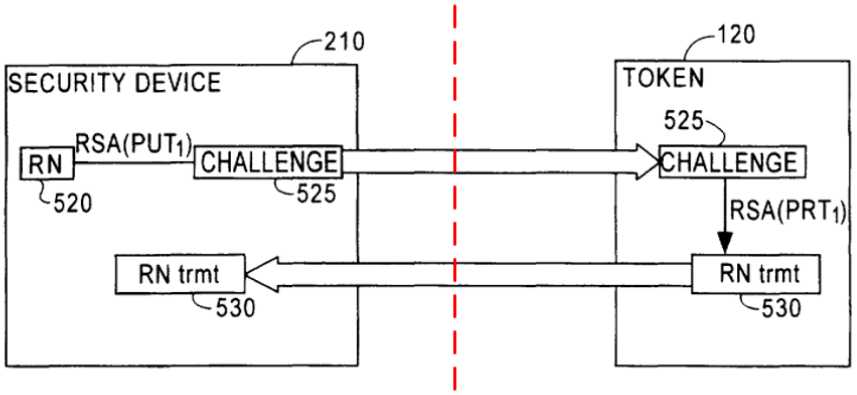


FIG. 6B

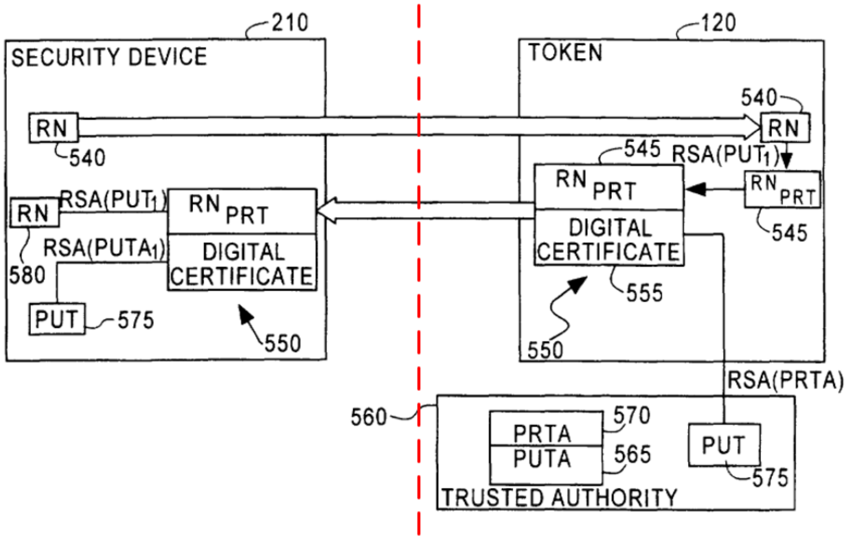


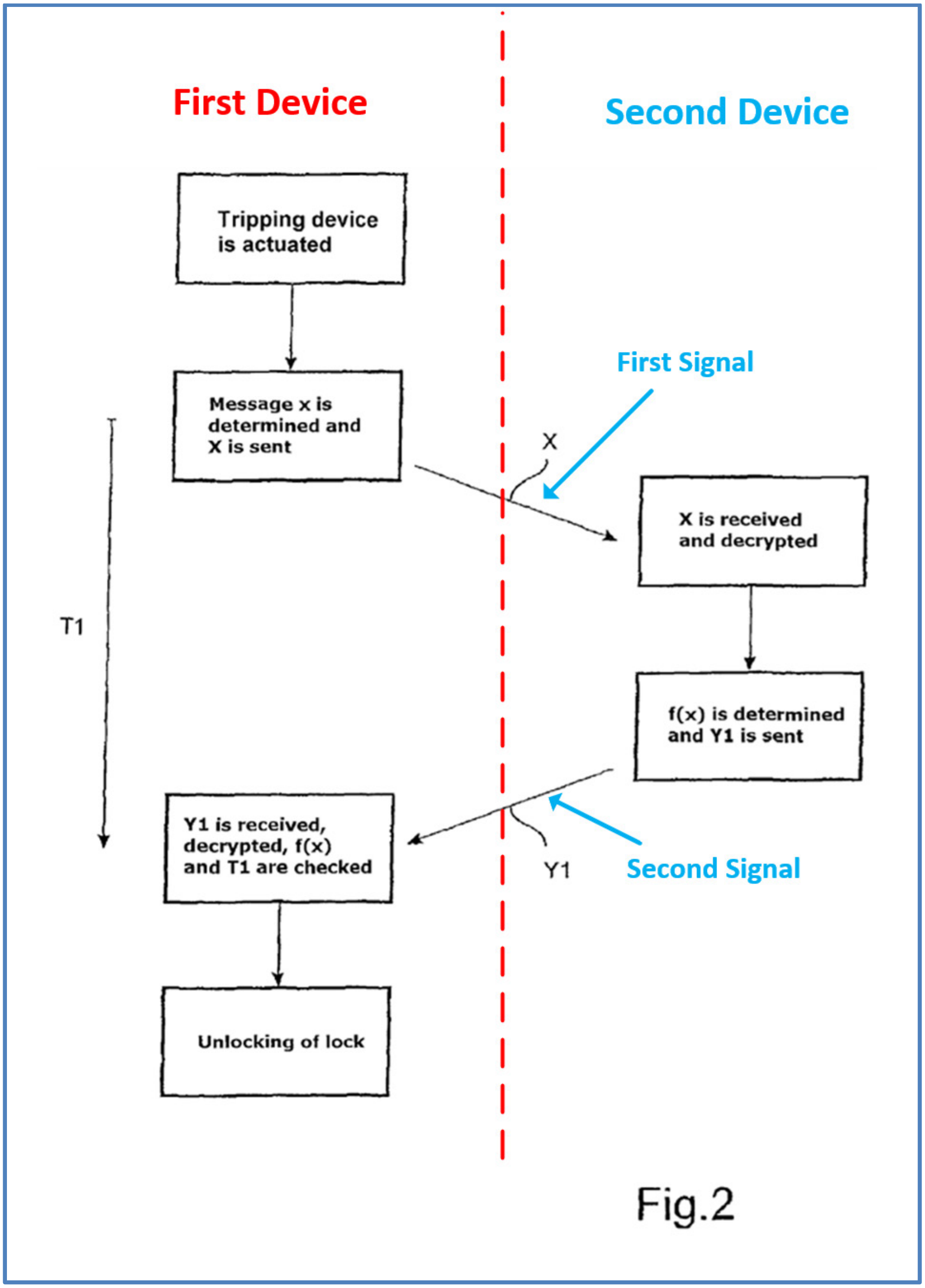
FIG. 6C

171. As I explained earlier, it would have been obvious to a POSITA to look to a reference such as Lundkvist for more examples of challenge-response protocols, as Lundkvist also uses timed challenge-response protocols to protect access. Lundkvist, like Davis, discloses an asymmetric-encryption based challenge-response protocol, but identifies that this protocol can be performed using either asymmetric or symmetric encryption. With limited options to try, a POSITA would have discovered though very limited, straight-forward experimentation, that symmetric encryption is much faster than asymmetric encryption and, for this reason, the symmetric-based encryption variant of Lundkvist will calculate distances much more accurately than Davis.

172. In this Maillard-Davis-Lundkvist combination, the first signal is the challenge that comprises a challenge value called "O_RND" (and some associated meta data) that is encrypted under a symmetric key. Lundkvist at 8:11-17. A POSITA would have found it obvious to use the session key, SK, pre-shared during device validation, for encrypting Lundkvist's challenge. I note that SK was intended to be used as an encryption key (Maillard at [0106], [0123], [0124]), and was previously shared between the devices, encrypted by Maillard's key X, ensuring that no other parties have access to it.

173. In more specificity, Lundkvist describes a control unit of the object, which is a "first device". This control unit, in the combination, is the security

module identified in Maillard. Lundkvist discloses that the control unit (security module, or "first device"), creates a message that comprises the first information x (lower case, not to be confused with the X key in Maillard) consisting of identity information, O_ID, that is unique to the object and a random number, O_RND, generated by the control unit. This message is encrypted (i.e., by the session key SK) to produce X (upper case, but still not to be confused with the X key in Maillard). Lundkvist at 8:11-16. The signal X is the claimed "first signal" that is transmitted to, and received by, the portable unit (receiving device). Lundkvist at 8:16-17, FIG. 2 (annotated version reproduced below).


First Device
Second Device
Tripping device is actuated
Message x is determined and X is sent
T1
X
First Signal
X is received and decrypted
f(x) is determined and Y1 is sent
Y1 is received, decrypted, f(x) and T1 are checked
Y1
Second Signal
Unlocking of lock
Fig.2

174. Because the challenge is encrypted by SK and SK is only transmitted after verification that device 60's certificate conforms to compliance rules, the first

signal is sent after the verification that the receiving device's certificate conforms to compliance rules, as required by this limitation.

4. Element 1[c] means for generating a second signal after receiving the first signal, wherein said second signal is derived using a secret known by the first device

175. I have previously explained why a POSITA would have found it obvious, and have been motivated, to combine Maillard, Davis, and Lundkvist in Section VI.E. and with respect to Element 1[b]. I incorporate those discussions into my analysis of Element 1[c] by reference.

176. Within the Maillard-Davis-Lundkvist combination, Lundkvist describes the second signal Y1 that is generated by the receiving device. Lundkvist at 8:11-28; Fig. 2. An annotated Lundkvist Figure 2 is reproduced below:

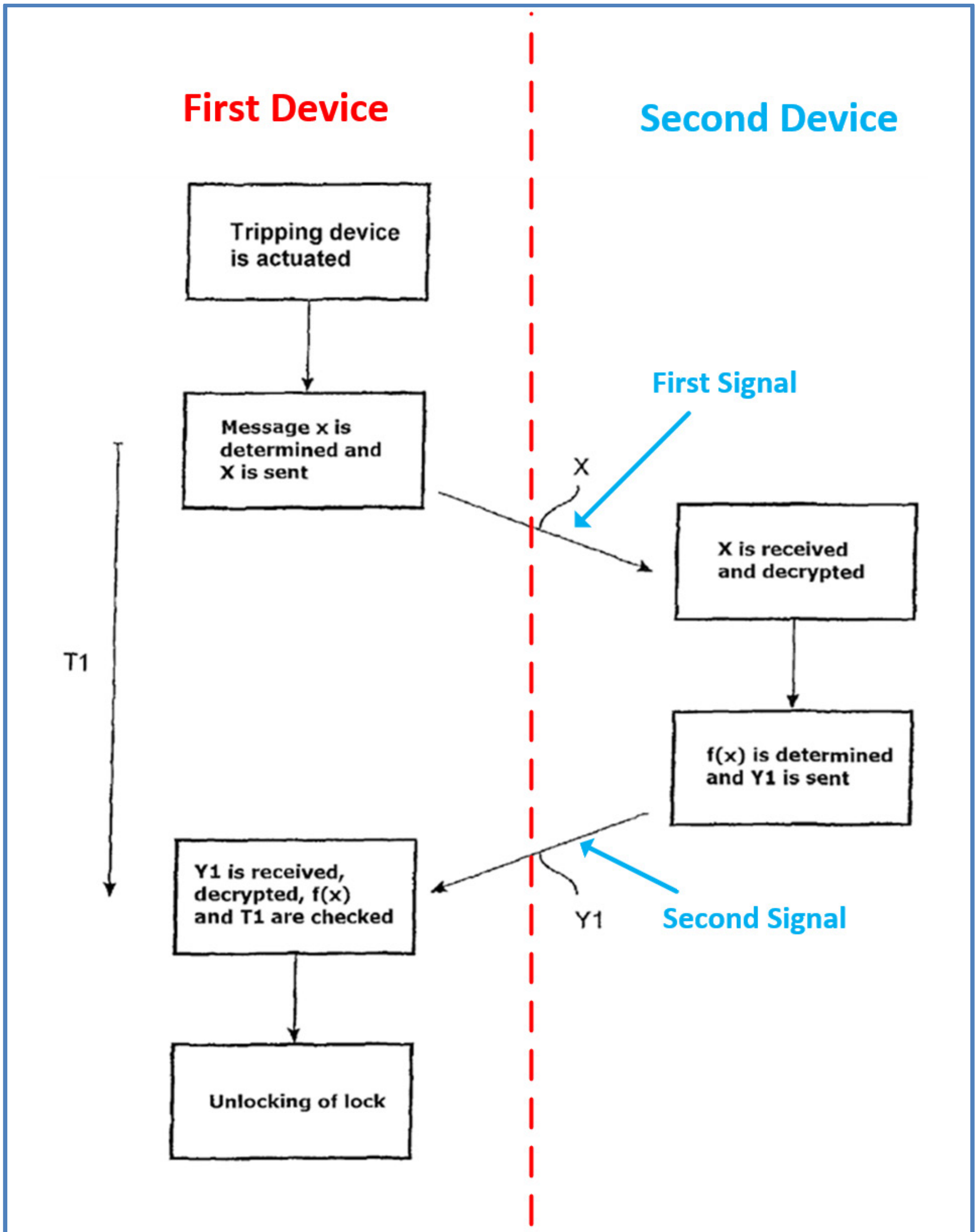

First Device
Second Device
Tripping device is actuated
Message x is determined and X is sent
First Signal
X
X is received and decrypted
T1
f(x) is determined and Y1 is sent
Y1 is received, decrypted, f(x) and T1 are checked
Y1
Second Signal
Unlocking of lock

Fig.2

177. Lundkvist discloses that the portable unit of the car, which is the receiving device (and is Maillard's device 60 in the combination) decrypts the first

signal X. As I explained with reference to limitation 1[b], X is encrypted by the session key SK. Lundkvist decrypts X to obtain the first information x, which contains the random number O_RND. Lundkvist at 8:19-20. The portable unit (device 60, or receiving device) then processes the first information x (using function f(x)), and encrypts it to produce second signal Y1. Lundkvist at 8:19-23.

178. As explained with respect to limitation 1[b], Lundkvist discloses both an asymmetric-encryption and symmetric-based encryption variant of the challenge-response protocol. As also explained, a POSITA, with limited options to try, would have easily discovered that the symmetric encryption is faster and, therefore, more accurate for distance measurements. Thus, a POSITA would have also found it an obvious choice to use symmetric encryption. And for the same reasons enumerated with respect to limitation 1[b], a POSITA would have found it obvious to use the shared secret session key (SK) as the encryption key for the symmetric encryption.

179. The process of symmetric encryption typically works in one of a couple of ways. In the 1990's, well-known cryptographer Bruce Schnier noted that some "encryption" schemes were simply XOR'ing the data and the key. He correctly pointed out that this is a very weak, very breakable encryption. Nevertheless, he also pointed out that it was, perhaps unfortunately, very common,

and could be done very quickly with a few computational resources. Schneier at p. 14.

180. Although XOR'ing with a key is generally not secure, as Schneier stated, there is a very specialized version of this approach which is very, very strong. If the key is as long as the data to be encrypted, truly random, and never repeated, XOR'ing the data with this kind of key is known as "One Time Pad" (OTP) and very secure. In the context of Maillard-Davis-Lundkvist combination, Maillard explains that the session key could be changed at any time. Maillard at 0107. Therefore, based on this suggestion by Maillard, it would have been certainly within the knowledge of a POSITA that the session key could be changed frequently to permit secure One Time Pads or to improve the security of the protocol even if a simple XOR encryption were used.

181. Otherwise, symmetric encryption typically takes the form of plugging the key into an encryption algorithm, such as DES (now deprecated) or AES, and some kind of "mode of operation," such as cipher-block-chaining mode or counter mode, to produce the encrypted data.

182. The most basic mode of operation is "Electronic Code Book" or ECB. In this mode, data is directly encrypted, one chunk at a time, using either the DES or AES algorithm and the key. ECB is strongly discouraged (I tell people to NEVER use it), but it does remain in use even today despite the warnings.

183. Counter mode is a popular mode wherein either the DES or AES algorithm, along with the key, generate a key stream of arbitrary length. The key stream is unpredictable and, from a certain point of view, approximates the One Time Pad. The data is XOR'ed with the key stream to produce the encrypted content. Schneier at p. 197. Figure 9.6 from Applied Cryptography is reproduced below.

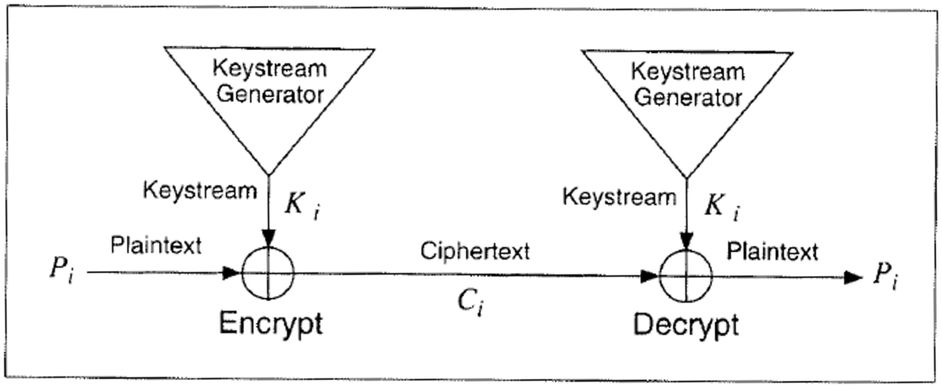


Figure 9.6 Stream cipher.

184. Importantly, in all of these methods, the encrypted output is a calculation based on the input data and the key. Or, in other words, the encrypted output is derived from the input value and the key.

185. Accordingly, the second signal Y1, which is information f(x) encrypted using the session key SK, is a signal that is derived from the secret SK known by the first device.

186. Thus, Y1 (the "second signal") can be conceptually expressed as

$$Y1 = E_{sk}(f(D_{sk}(X)))$$

Where "E_{SK}" represents encryption using SK and "D_{SK}" represents decryption using SK and f is Lundkvist's transformation function. X, as previously discussed, is the first signal, but also the encrypted information O_RND.

187. Simplified, $Y1 = G_{SK}(X)$, where X is the first signal and G_{SK} is the composition of the functions (E_{sk} * f * D_{sk}). This illustrates that Y1 ("second signal") is derived from X ("first signal") using session key SK ("secret known to the second device").

188. The above operations for generating the second signal in the Maillard-Davis-Lundkvist combination are similar to the '977 Patent description as to how the second signal is generated: "The second device receives the signal via a receiver 311 and 313 modifies the signal by using the locally stored secret. The signal is modified according to rules known by the first device 301 and transmitted back to the first device 301 via a transmitter 315." '977 Patent at 6:34-39.

5. Element 1[d]: means for transmitting said second signal;

189. Lundkvist transmits the second signal from the portable unit (device 60, "receiving device") to the transmitter (security module, "first device"). Lundkvist at 8:11-28; Fig. 2. An annotated Lundkvist Figure 2 is reproduced below:

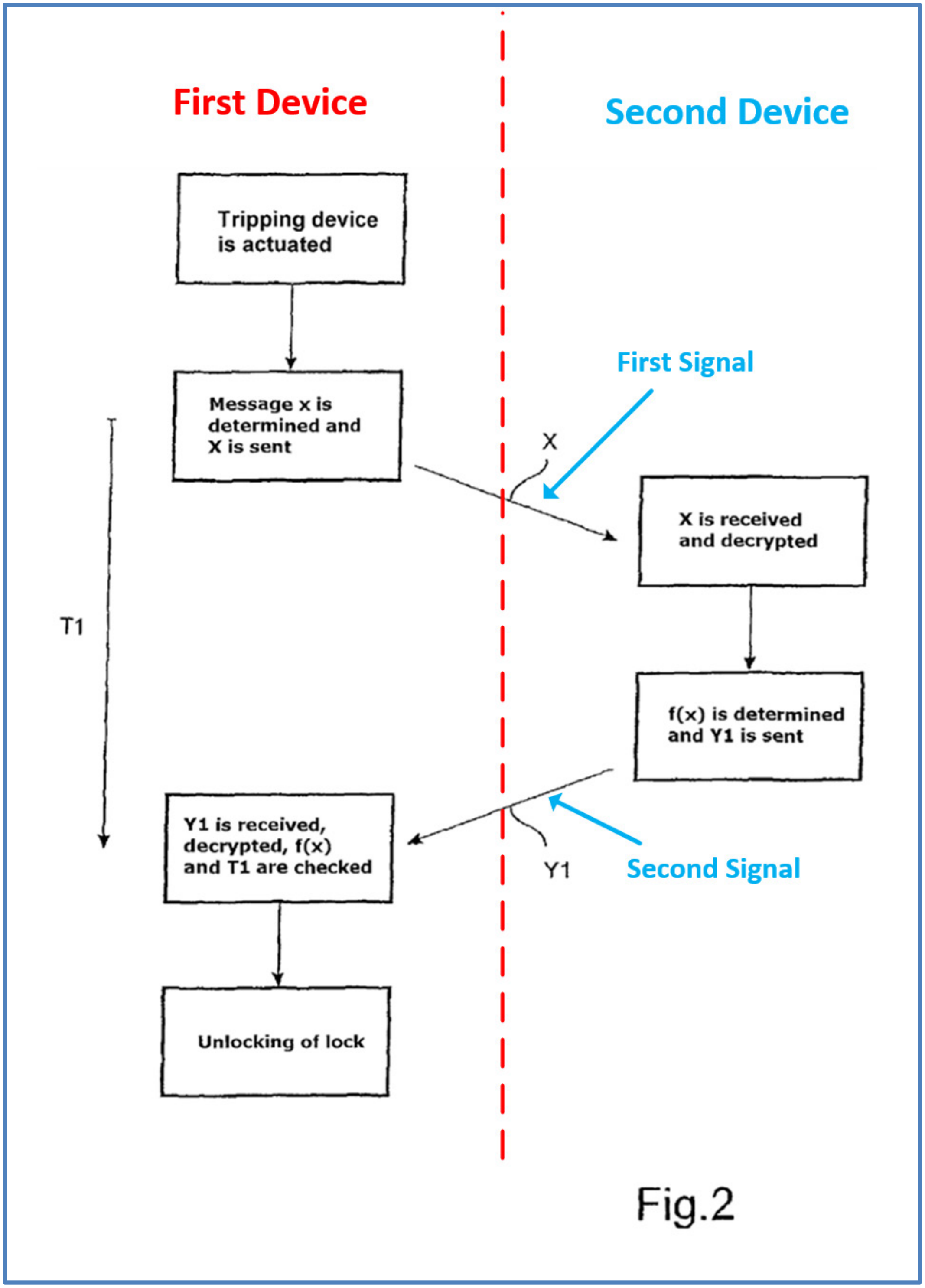

First Device
Second Device
Tripping device is actuated
Message x is determined and X is sent
First Signal
X
X is received and decrypted
T1
f(x) is determined and Y1 is sent
Y1 is received, decrypted, f(x) and T1 are checked
Y1
Second Signal
Unlocking of lock

Fig.2

6. Element 1[e]: means for generating a secure authenticated channel using the secret.

190. The '977 patent describes what it means by a secure authenticated channel (SAC): "[t]echnology to perform device authentication and encrypted content transfer is available and is called a secure authenticated channel (SAC)." '977 Patent at 2:11-13.

191. Maillard describes creating secure authenticated channels (SAC) using that terminology and in a manner that is compatible with the '977's definition. Specifically, Maillard's SACs incorporate device authentication and encrypted content transfer.

192. Maillard describes creating a SAC between two device 60's (e.g., a DVD player and a display), based on authorization from a mutual security module. It can also, be generated between the device 60 and the security module. In both cases, the SAC is generated using the session key SK. Maillard at [0114], [0117]-[0124]; see also Maillard at Fig. 9, reproduced below (see also, e.g., Fig. 7).

193. As described previously with respect to limitation 1[a] and limitation 1[b], in Maillard, the session key SK is only transmitted from the security module to one or more device 60's after the device or devices are authenticated by validation of their respective certificates. Moreover, the key SK is transmitted encrypted by a key X. Key X is generated by the device 60, but transmitted in encrypted form under the security module's public key. This links SK to device

authentication. That is, only the security module could have decrypted X, so SK is understood to be shared between the identified security module and the identified device. Thus, data protected by SK is both encrypted and device authenticated.

7. Element 1[f]: means for receiving over the secure authenticated channel a protected content after the first device determines that the second signal is derived using the secret and a time between a transmission of the first signal and receipt of the second signal by the first device is less than a predetermined time.

194. Maillard describes device 60 receiving protected content over the secure authenticated channel identified in my analysis of element 1[e]. The combination of Maillard-Davis-Lundkvist transmits the protected content (and, thus, it is received over the SAC), after the security module (control unit, or "first device") determines (1) that the encrypted response (Y1, "second signal") is encrypted by the session key SK ("derived using the secret"). And also determines (2) that the time between the transmission of the encrypted challenge (X, "first signal") and the receipt of the encrypted response (Y1, "second signal") by the security module (control unit, or "first device") is less than a predetermined time.

195. I explained in my analysis of limitation 1[e] that Maillard describes the creation of secure authenticated channels (SACs). The SAC's described by Maillard are used for transmitting content to device 60.

196. Maillard describes a SAC from one device 60 to another (e.g., DVD player to display). The SAC is generated using the SK from the security module, but is created between the two devices. Data (e.g., audio and/or visual data - Maillard at [0109]) is encrypted using the session key, SK, using Triple DES encryption algorithm at a first device and transmitted to a second device; the second device uses the same session key, SK, to decrypt and obtain the data. Maillard at ¶¶ [0123]-[0124]; FIG. 9. The security module may be a "smart card" that is plugged into one of the device 60s, permitting an SAC to be created between a device 60 plus the smart card and another device 60. Maillard at [0085]. It is also possible for an SAC to be created directly between a security module and a device 60. Maillard at [0113]-[0114]; FIG. 7.

197. I have previously explained why a POSITA would have found it obvious, and have been motivated, to combine Maillard, Davis, and Lundkvist and with respect to Element 1[b]. I incorporate those discussions into my analysis of Element 1[f] by reference.

198. In the Maillard-Davis-Lundquist combination, Davis and Lundkvist each describe the additional security protocols that authorize access upon satisfaction of two conditions: (1) round trip time is within a predtermined value, and (2) the challenge-response protocol is correctly computed. Davis, pp. 11-12; FIG. 5; Lundkvist, 8:11-28; FIG. 2.

199. As previously explained with respect to other limitations of Claim 1, Maillard describes embodiments that are designed for a home network. In at least these embodiments, A POSITA would have found it obvious to not transmit content over the SAC to be received by device 60 unless the device was on the home network. Thus, it would have been obvious to a POSITA to not transmit the protected content (and thus, be received by the receiving device) until after the validating the challenge response and verifying that the receipt of the response was within a predetermined time, as described by Davis and Lundkvist.

200. Lundkvist provides details for this challenge-response process. First, Lundkvist describes that the control unit of the object (security module, or “first device”) receives the second signal Y1. The second signal Y1 is the encrypted response value f(O_RND), where O_RND was transmitted in the first signal and f is some function. Lundkvist at 8:23-24. A POSITA would have found it obvious to encrypt Y1 using the shared symmetric key SK, as the value was securely shared as part of the authentication process.

201. Once the second signal is received, the control unit measures the round trip time T1, which is the time from transmission of the first signal X (the encrypted value of O_RND) to reception of the second signal Y1 (the encrypted value of f(O_RND)). Lundkvist at 8:24-26.

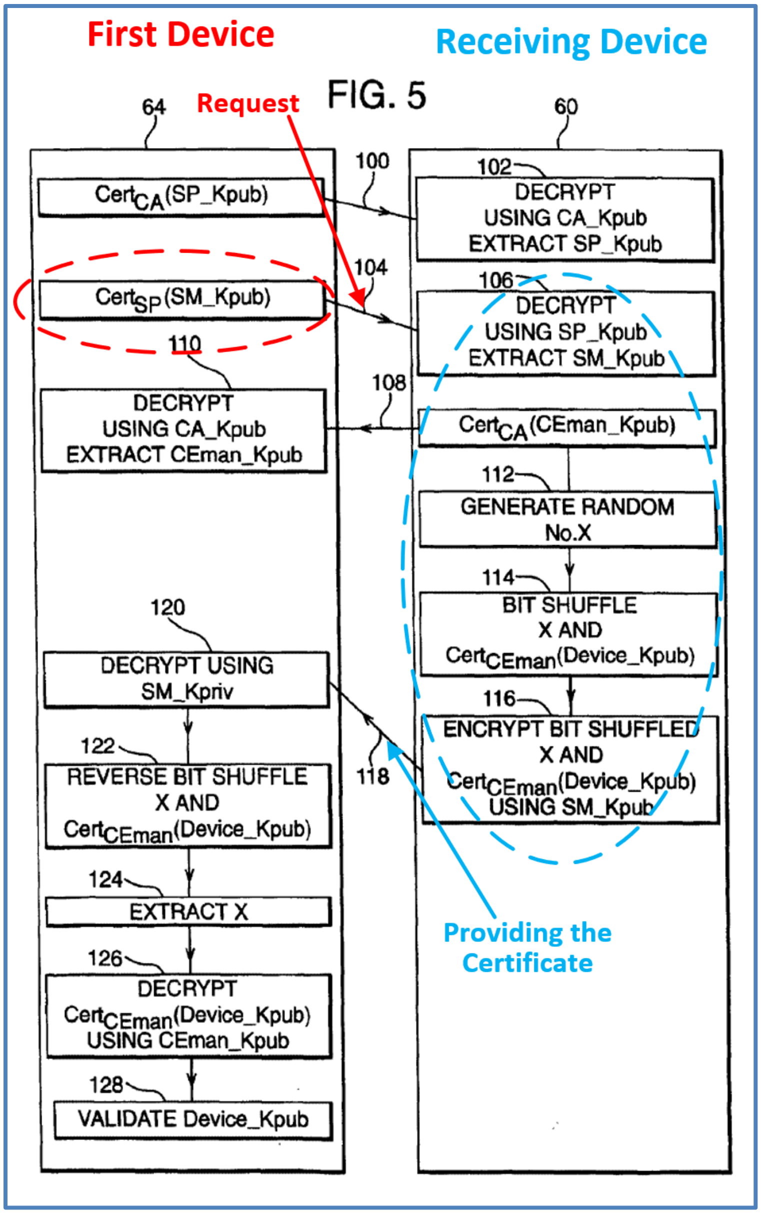
202. The control unit decrypts Y1 to get f(O_RND), also known as E_SVAR and determines if E_SVAR is equal to f(O_RND). Lundkvist at 8:23-28.

203. As explained in reference to limitation 1[c], all forms of symmetric encryption involve a derivation that uses the key to compute the output. Thus, Y1, the second signal, is derived using the key and the successful decryption of Y1, which is required to verify the challenge, determines that Y1 was derived using the key.

204. Thus, I conclude that the combination of Maillard-Davis-Lundkvist renders obvious all the limitations of claim 1.

B. Claim 2: The receiving device of claim 1, wherein said providing said certificate is responsive to a request.

205. Maillard discloses that device 60 ("receiving device") provides its certificate in response to a request from the security module. In particular, Maillard discloses that "[t]he validation procedure is initiated by the security module…" Maillard at [0091]. The security module transmits its encrypted certificate to the device 60, which is both a request for the device's certificate and provides the capability to respond. Specifically, the device 60 requires the security module's certificate in order to correctly encrypt its own certificate with the security module's public key. Maillard at [0091]-[0095]. Annotated Figure 5 of Maillard below illustrates this point.


First Device
Receiving Device
FIG. 5
64
Request
60
100
Cert$_{CA}$(SP_Kpub)
102
DECRYPT USING CA_Kpub EXTRACT SP_Kpub
104
Cert$_{SP}$(SM_Kpub)
106
DECRYPT USING SP_Kpub EXTRACT SM_Kpub
110
108
DECRYPT USING CA_Kpub EXTRACT CEman_Kpub
Cert$_{CA}$(CEman_Kpub)
112
GENERATE RANDOM No.X
114
BIT SHUFFLE X AND Cert$_{CEman}$(Device_Kpub)
120
DECRYPT USING SM_Kpriv
116
ENCRYPT BIT SHUFFLED X AND Cert$_{CEman}$(Device_Kpub) USING SM_Kpub
122
REVERSE BIT SHUFFLE X AND Cert$_{CEman}$(Device_Kpub)
118
124
EXTRACT X
Providing the Certificate
126
DECRYPT Cert$_{CEman}$(Device_Kpub) USING CEman_Kpub
128
VALIDATE Device_Kpub

206. In addition, Maillard clearly describes sending the certificate to the first device in step 118. A POSITA would have found sending a certificate in response to a request an obvious endeavor as opposed to automatically sending the certificate. There is certainly nothing novel in sending a certificate responsive to a request. Thus, I conclude that the combination of Maillard-Davis-Lundkvist renders obvious claim 2.

C. Claim 3: The receiving device of claim 1, further comprising means for receiving said secret.

207. I have already explained in my analysis of limitation 1[b] that the receiving device receives the secret. I incorporate this analysis by reference.

208. Thus, I conclude that the combination of Maillard-Davis-Lundkvist renders obvious claim 3.

D. Claim 8: The receiving device of claim 1, further comprising means for displaying said received protected content.

209. Maillard discloses devices that display content. In particular, Maillard describes that the second device can be "a digital display 14 for the display of the data played by the DVD player 12. The display 14 is preferably provided in the form of a digital television." Maillard at [0075]; see also, Maillard at [0123]-[0124].

210. Thus, I conclude that the combination of Maillard-Davis-Lundkvist renders obvious claim 8.

E. Claim 9: The receiving device of claim 1, wherein the secret comprises a random number.

211. Maillard discloses that the session key SK ("the secret") is a random number. Maillard explains that "the security module 64 generates a random session key SK…. The session key SK is thereafter used to encrypt data transferred between the device 60 and the security module 64." Maillard at [0105]. A POSITA would know that a "random" symmetric key was a random number.

212. Thus, I conclude that the combination of Maillard-Davis-Lundkvist renders obvious claim 9.

F. Claim 10: The receiving device of claim 1, wherein the predetermined time is based on a communication system associated with the first device.

213. I have previously discussed in reference to limitation 1[f] the challenge-response protocol of Lundkvist in the Maillard-Davis-Lundkvist combination. I incorporate my analysis of this limitation by reference.

214. In Lundkvist, the control unit (analogous to the security module, or "first device") transmits the challenge X to the portable unit (analogous to the device 60, or "second device"), which subsequently sends the response Y1 back to the security module. The control unit, upon receipt of Y1, determines the roundtrip time T1. The control unit then compares the measured time to a predetermined value to determine whether or not the two devices are at a distance greater than "a maximal permitted distance." Lundkvist at 1:12-17, 12:8-10.

215. Lundkvist's "predetermined time" is based on round-trip time estimates appropriate for the communication medium. For example, Lundkvist describes both systems where sound waves are used for signal transmissions as well as systems that use radio waves or microwaves. Lundkvist at 3:1-6; 13:11-14. Determining distance using sound waves is based in part on the speed of sound while determining distance using radio waves or microwaves is based in part on

the speed of light. Thus, Lundkvist's predetermined time is based on the communications system associated with the first device.

216. Similarly, Davis explains that the timing measurement can correspond to different forms of wireless communication transmission systems (e.g., infrared, radio frequency (RF) or other transmission protocols), and the predetermined time is used to determine whether the two devices are within a particular proximity (e.g., within 20 feet) of each other. Although both of these systems are based on the idealized speed of light, each will have different speeds in practice. Davis, 7:6-13, 8:9-12, 10:20-22; see also Maillard, ¶ [0075].

217. Therefore, the combination of Maillard-Davis-Lundkvist renders obvious claim 10.

G. Claim 11

218. Claim 11 and its dependents (and the claims of the '564 Patent) use the term "second device" to identify what was called the "receiving device" in claims 1-10 of the '977 Patent. To avoid repetition, I will refer back to my prior analyses of claim 1 in the remaining sections of my declaration, with the understanding that the second device and the receiving device are equivalent terms.

Element 11[Preamble]: A second device for receiving protected content, the second device comprising:

a memory, the memory storing a public key and a private key, wherein the public key and private key are a pair;

a microprocessor circuit connected to the memory via a communication bus, the microprocessor circuit arranged to

219. Maillard discloses device 60 ("second device") which is an electronics device such as a DVD player or display. A POSITA would have known that that these types of devices were configured with a processor and memory.

220. I have previously explained in section VI.E and claim 1 that it would have been obvious to a POSITA to combine Maillard, Davis, and Lundkvist and I incorporate those discussions by reference into my analysis of claim 11.

221. Within the Maillard-Davis-Lundkvist combination, Davis describes a token, such as a smart card, that includes its own memory and processor. Davis at 10:17-11:7, FIG. 4. In this combination, Davis's token ("second device") responds to challenges from the security device attached to the personal computer ("first device"). It would have been obvious to a POSITA that Davis's token ("second device") could be attached to Maillard's device 60 ("second device") as Maillard describes such devices as being able to plug in smart cards. Maillard at [0085].

222. Therefore, at the very least, it would have been obvious to a POSITA that the Maillard-Davis-Lundkvist combination renders obvious a memory and a processor in the second device.

223. Davis further explains that memory stores cryptographic keys, which a POSITA would understand include the RSA public/private key pairs disclosed within the Davis reference. Davis discloses that, "[t]he memory element 320… is preferably configured to contain its unique public and private key pair and perhaps a digital certificate to allow the token 120 to securely transmit its public key 'PUT' to the security device…"." Davis at 10:17-11:7, FIG. 4.

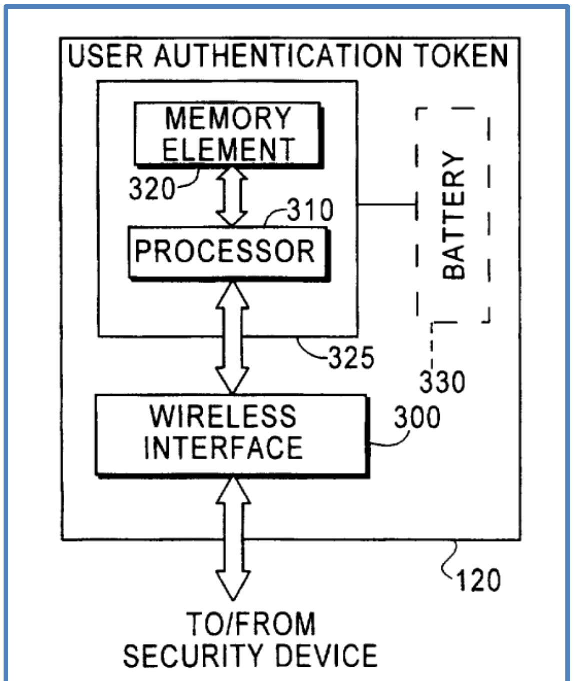


FIG. 4

Element 11[a]: provide a certificate to a first device identifying said second device, said certificate comprising the public key.

224. I have already explained with reference to limitation 1[a] that Maillard discloses providing a certificate from device 60 ("second device") to the security

module ("first device"), that the certificate included a public key, and that the public key was a unique identifier for the device 60. I incorporate this discussion into my analysis of limitation 11[a] by reference.

Element 11[b]: receive a first signal from a first device after the first device determines, based on information obtained from the certificate that the receiving device is compliant.

225. I have already explained with reference to limitation 1[b] that it would have been obvious to a POSITA to combine Maillard-Davis-Lundkvist such that after Maillard determines, based on the certificate of device 60, that device 60 is compliant, a timed challenge-response protocol based on the combined teachings of Davis and Lundkvist would initiate. The challenge of this challenge-response protocol represents the first signal. I incorporate this discussion into my analysis of limitation 11[b] by reference.

Element 11[c]: obtain a secret encrypted by the public key, wherein the secret is known by the first device;

226. Maillard discloses that the security module ("first device") transmits the session key SK, which is a secret known to the security module (because it created it), to the device 60. In Maillard, the session key SK is encrypted with a symmetric key X.

227. It would have been obvious to a POSITA that session key SK could be encrypted with an asymmetric key instead, which as I explained earlier, was a known alternative to symmetric encryption. Maillard already discloses that device 60 has a public key. It would have been obvious, and consistent with the usage of asymmetric keys in the prior art, to include the corresponding private key for device 60. The security module already receives the public key for device 60 as part of the normal protocol. A POSITA would have found it obvious to use this key to transmit session key SK back to the device 60. As there are no timing requirements expressed by Maillard, asymmetric encryption would have been an obvious alternative to symmetric encryption. A POSITA was well-aware of this alternative and would have implemented and would have been predictably and successfully able to securely transfer the session key from the first device to the second device.

Element 11[d]: use the private key to determine the secret;

228. I explained with reference to claim limitation 11[c] why it would have been obvious to encrypt the secret SK with device 60's public key. I incorporate that discussion into my analysis of 11[d] by reference.

229. Once the secret is transmitted encrypted by public key, the only way to decrypt it (which would "determine" it) is to use the private key. Thus, it would have been obvious to a POSITA for device 60 to decrypt the session key SK with

its private key. As I explained with reference to the preamble of claim 11, Davis explains that memory of the second device (described as being tamper proofed) would store the private key in the second device. Davis at 10:25-28.

Element 11[e]: derive a second signal, wherein the second signal is the first signal modified using the secret.

230. I have already explained how it would have been obvious to a POSITA to combine Maillard-Davis-Lundkvist to generate a second signal that is derived from the first signal using the secret with reference to limitation 1[c]. As derivation is a type of modification, my analysis of limitation 1[c] applies equally to limitation 11[e], and I incorporate my analysis of limitation 1[c] by reference.

Element 11[f]: send the second signal to the first device after receiving the first signal.

231. I have already explained how both Lundkvist and Davis disclose timed challenge responses with reference to limitation 1[b] and 1[c]. These timed challenge responses necessarily require the challenge (first signal) to be received before the second signal can be generated, much less sent. I incorporate my analysis of limitations 1[b] and 1[c] into my analysis of 11[f] by reference.

Element 11[g]: receive protected content after the first device has determined that at least the second signal is derived from the secret and a time difference between first device's provision of the

first signal and first device's reception of the second signal difference is less than a predetermined time.

232. I have already explained that it would have been obvious to a POSITA to receive protected content in Maillard's device 60 from the security module (first device) or from another device 60 with the security module's authorization, after successfully validating a timed challenge response as taught by Davis and Lundkvist, with reference to limitation 1[f]. The timed challenge response is validated if the response ("second signal") is successfully decrypted using the secret key ("derived from the secret") and if the time difference between the transmission of the challenge ("provision of the first signal") and receipt of the response ("reception of the second signal") is less than a predetermined time. I incorporate my analysis of limitation 1[f] into my analysis of limitation 11[g] by reference.

H. Claim 12: The second device of claim 11, wherein the secret comprises a random number.

233. My analysis of claim 9 applies equally to claim 11, and I incorporate my analysis of claim 9 by reference.

234. Thus, the combination of Maillard-Davis-Lundkvist renders obvious claim 12.

I. Claim 14: The second device of claim 11, wherein the microprocessor circuit is further arranged to receive the secret from the first device.

235. My analysis of claim 3 applies equally to claim 11 and I incorporate my analysis of claim 3 by reference.

236. Thus, the combination of Maillard-Davis-Lundkvist renders obvious claim 14.

J. Claim 15: The second device of claim 11, wherein the certificate comprise an identity of the second device.

237. I have already explained with reference to limitation 1[a] that the certificate contains a unique public key that identifies device 60 ("second device"). I incorporate my analysis of limitation 1[a] by reference.

238. Thus, the combination of Maillard-Davis-Lundkvist renders obvious claim 15.

K. Claim 16: The second device of claim 11, wherein the predetermined time is based on a communication system associated with the first device.

239. I have already explained with reference to claim 10 that the predetermined time is based on a communication system associated with the first device. I incorporate my analysis of claim 11 by reference.

240. Thus, the combination of Maillard-Davis-Lundkvist renders obvious claim 16.

L. Claim 17: The second device of claim 11, wherein the microprocessor circuit is further arranged to: use the secret to generate a secure authenticated channel between the first device and the second device; and use the secure authenticated channel to receive the protected content.

241. I have already explained with reference to limitations 1[e] and 1[f] that the session key SK is used to generate a secure authenticated channel between the security module, or the security module in combination with a device 60, ("first device") to a device 60 ("second device"). I also explained that the secure authenticated channel was used to received protected content. I incorporate my analysis of these limitations by reference.

242. Thus, the combination of Maillard-Davis-Lundkvist renders obvious Claim 17.

M. Claim 18: The second device of claim 11, wherein the modification is a XOR operation using the first signal.

243. I have previously explained with reference to limitations 1[b] and 1[c] that it would have been obvious to a POSITA to combine Maillard-Davis-Lundkvist. Within this combination, the first signal is a symmetrically encrypted challenge. I have also described the methods that would have been known to a POSITA for performing symmetric encryption. I incorporate these discussions into my analysis of Claim 18 by default.

244. If the symmetric encryption is performed by XOR'ing with the key, or if the symmetric encryption is performed by XOR'ing with key stream using AES

or DES, the plaintext data is XOR'ed to produce the ciphertext. In any of these cases, the encrypted message (the first signal) is also decrypted by XOR'ing, either with the key itself or with key stream. Schneier at 14, 197. In any event, given the limited number of options to try, it would have been obvious for a POSITA to use XOR with the encrypted challenge ("first signal") for decryption.

245. The encrypted challenge is decrypted before the response can be calculated. Thus, when XOR is used for decryption, XOR is used in the modification of the first signal to produce the second signal.

N. Claim 19: The second device of claim 11, wherein said secret is transmitted using a transfer protocol, said transfer protocol selected the group consisting of a key transport protocol, a key management protocol and a key exchange agreement.

246. I have already explained with reference to limitation 11[c] and limitation 11[d] that Maillard discloses transmitting the session key SK ("the secret") encrypted by device 60's public key. This is a key transport protocol. HAC at 490, 506-509. I incorporate my analysis of limitations 11[c] and 11[d] into this section by reference.

O. Claim 20: The second device of claim 11, wherein the microprocessor circuit is further arranged to receive the secret by using a key transfer protocol.

247. I have already explained in reference to Claim 19 that Maillard discloses a key transport protocol. I incorporate my analysis of claim 19 into this section by reference.

VIII. UNPATENTABILITY OF THE CHALLENGED CLAIMS OF THE '564 PATENT FOR GROUND 1 (THE COMBINATION OF MAILLARD, DAVIS AND LUNDKVIST)

A. Claim 1

Element 1 [Preamble]: A second device for receiving delivery of a protected content from a first device, the second device comprising a processor circuit, the processor circuit arranged to execute instructions, the instructions arranged to:

248. I understand that the preamble of a patent claim is sometimes not limiting. Nevertheless, to the extent that the preamble of Claim 1 of the '564 patent is limiting, Maillard describes a "device 60" that, as explained more fully below, is a "second device for receiving delivery of protected content from a first device" as required by Claim 1.

249. Figure 1, for example, shows an example DVD-Recorder-TV system, where the DVD provides protected audio-visual content to the recorder or the TV, with Figure 3 showing a receiving device 60 that would be the equivalent of the digital recorder or the TV that would receive the content. Maillard at 0074-0077, 0088.

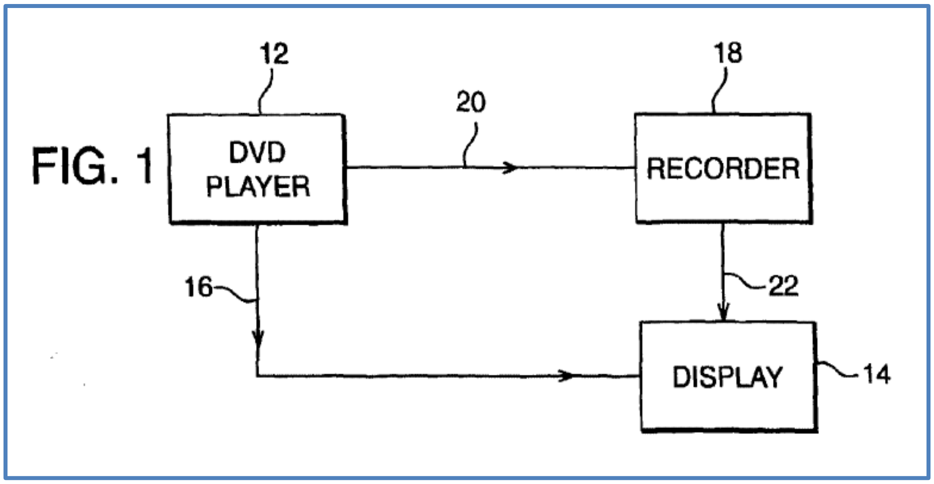
250. 

FIG. 1

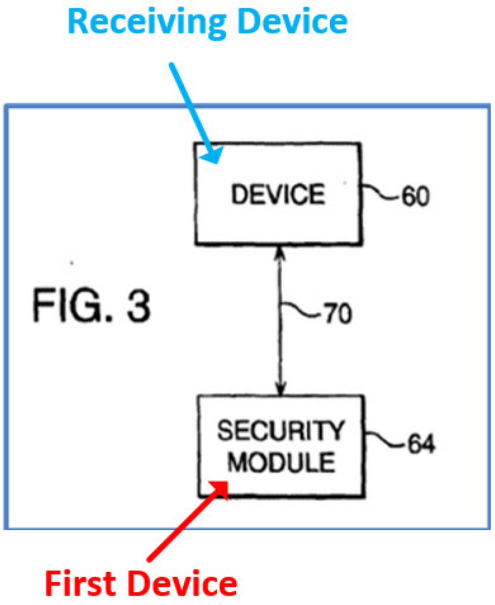


FIG. 3

251. Maillard further states, "The session key SK is thereafter used to encrypt data transferred between the device 60 and the security module 64." Maillard at [0106]. The "device 60" receives content such as media content from another device (such as a DVD player) with keys from the security module, which, as I explained earlier, can be inserted into or be part of the DVD player or source device. Maillard at [0011]). Thus collectively forming a "first device" and the

content would be transmitted from the first device to the second device. Figure 9 shows the transfer of encrypted content from one device to another device after the session key has been securely shared. Maillard at 0123-0124 and Fig. 9. Alternatively, Maillard describes protected content being sent directly from the security module to the device 60. Maillard at [0113]-[0114].

252. Maillard discloses device 60 ("second device") is an electronics device such as a DVD player or display. A POSITA would have known that that these types of devices were configured with a processor ("processor circuit arranged to execute instructions").

253. I have previously explained in section VI.E. and the associated claims of the '977 patent that it would have been obvious to a POSITA to combine Maillard, Davis, and Lundkvist and I incorporate those discussions by reference into my analysis of claim 1.

254. Within the Maillard-Davis-Lundkvist combination, Davis describes a token, such as a smart card, that includes its own memory and processor. Davis at 10:17-11:7, FIG. 4. In this combination, Davis's token ("second device") responds to challenges from the security device attached to the personal computer ("first device"). It would have been obvious to a POSITA that Davis's token ("second device") could be attached to Maillard's device 60 ("second device") as Maillard describes such devices as being able to plug in smart cards. Maillard at [0085].

255. Therefore, at the very least, it would have been obvious to a POSITA that the Maillard-Davis-Lundkvist combination renders obvious a memory and a processor in the second device.

Element 1[a]: provide a certificate to the first device prior to receiving a first signal, wherein the first signal is sent by the first device, wherein the certificate is associated with the second device.

256. Claim limitation 1[a] of the ‘977 patent and claim limitation 1[b] of the ‘977 patent collectively have similar language to limitation 1[a] of the ‘564. I incorporate my analysis of these two limitations (¶¶155-174) into my analysis of (‘564) limitation 1[a].

257. As previously explained, device 60 (“second device”) transmits an encrypted certificate in order to establish its identity with the security module (“first device”). The encrypted certificate of device 60 includes a unique public key that identifies the device. Thus, the certificate is associated with the second device.

258. In the incorporated-by-reference analyses of limitations 1[a] and 1[b] of the ‘977 patent (¶¶155-174), and in section VI.E, I also explained that it would have been obvious to a POSITA to combine Maillard, Davis, and Lundkvist. This analysis applies equally to this limitation of the ‘564. As with the ‘977 limitations, in this combination, Lundkvist’s encrypted challenge of a challenge-response

protocol is the first signal. And, as I explained with reference to limitation 1[b] of the ‘977, this challenge is not sent until after the security module has shared session key SK with device 60, and that the security module cannot share the session key SK with device 60 until after receiving device 60’s certificate. The device 60 certificate is sent with a random number X to enable the security module to securely transmit the session key SK and it cannot transmit the session key SK until it receives random number X (with the certificate). Thus, certificate is provided to the security module (“first device”) before the encrypted challenge (“first signal”) is received by device 60 (“second device”).

Element 1[b]: receive the first signal when the certificate indicates that the second device is compliant with at least one compliance rule.

259. Claim limitation 1[b] of the ‘977 patent has similar requirements to this (‘564) limitation 1[b] with respect to certifying that the second device is compliant with at least one compliance rule. I incorporate my analysis of claim limitation 1[b] of the ‘977 patent (¶¶159-174) into my analysis of this (‘564) limitation 1[b] by reference.

260. In my analysis of limitation 1[b] of the ‘977 patent, I explained that the security module (“first device”) determines if the public key in the certificate associated with device 60 (“second device”), is on a preauthorization list, or is not on an unauthorized list, and/or is signed by the appropriate authority. Thus, the

certificate indicates (i.e., by its public key and/or signature) that it is compliant with at least one compliance rule.

261. Although not required by the ‘977 limitation 1[b], I also explained in my analysis of that limitation that only after the validation checks on the certificate (“the certificate indicates that it is compliant with at least one compliance rule”) the security module (“first device”) transmits a session key SK. And, as also explained with respect in my analysis of the ‘977 limitation 1[b], the encrypted challenge of the Maillard-Davis-Lundkvist combination (“first signal”) is not sent until after the security module has shared session key SK with device 60, and that the security module cannot share the session key SK with device 60 until after receiving device 60’s certificate. The device 60 certificate is sent with a random number X to enable the security module to securely transmit the session key SK and it cannot transmit the session key SK until it receives random number X (with the certificate). Thus, the encrypted challenge cannot be received by device 60 until after the security module has determined that the certificate of device 60 indicates that it is compliant with at least one compliance rule.

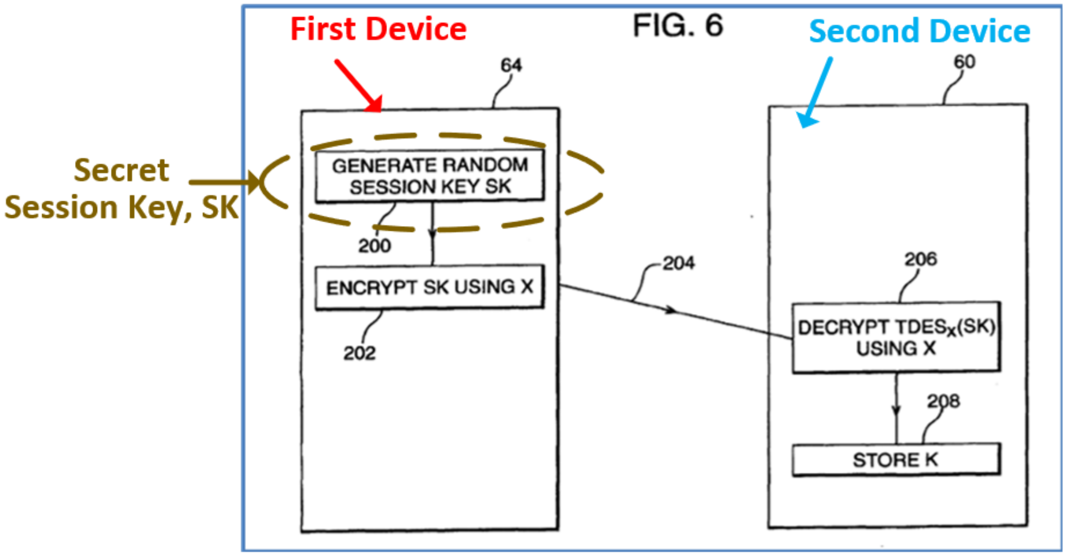


FIG. 6

Element 1[d]: create a second signal, wherein the second signal is derived from a secret known by the second device.[8]

262. Claim limitation 1[c] of the ‘977 patent has similar requirements to this (‘564) limitation 1[d] with respect to certifying that the second device is compliant with at least one compliance rule. I incorporate my analysis of claim limitation 1[c] of the ‘977 patent (¶¶175-188) into my analysis of this (‘564) limitation 1[d] by reference.

263. In my analysis of the ‘977 patent’s limitation 1[c], I discussed that it would have been obvious to a POSITA to combine Maillard, Davis, and Lundkvist. Within this combination, Lundkvist’s challenge-response protocol sends an

[8] Counsel for TCL has asked me to label this limitation as 1[d] for consistency with their petition.

encrypted challenge X, which is the encrypted value of O_RND (and potentially additional data such as O_ID). See annotated Figure 2 below.

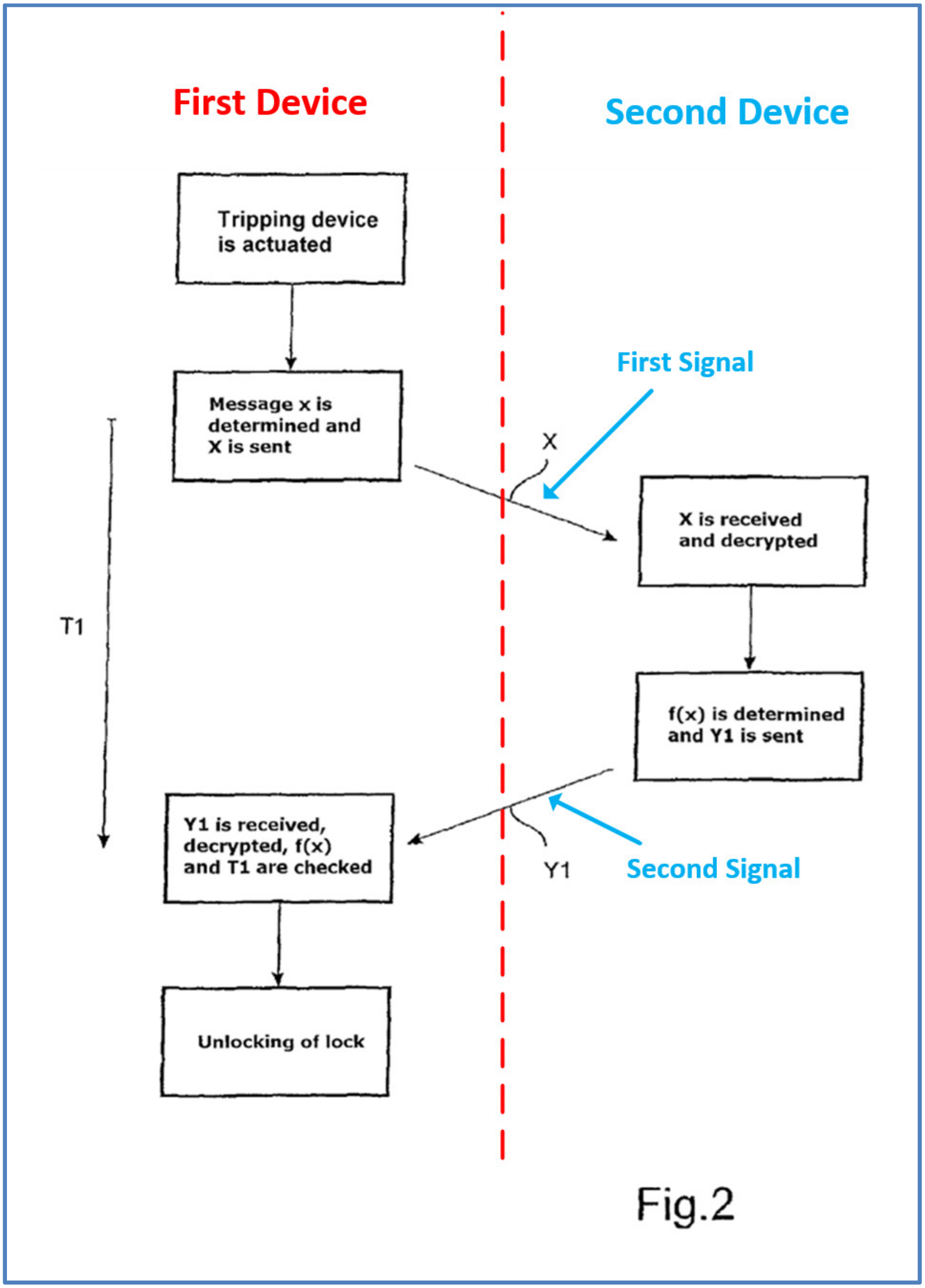

First Device
Second Device
Tripping device is actuated
Message x is determined and X is sent
First Signal
X
X is received and decrypted
T1
f(x) is determined and Y1 is sent
Y1 is received, decrypted, f(x) and T1 are checked
Y1
Second Signal
Unlocking of lock

Fig.2

264. As previously explained, in this combination, Maillard's security module (comparable to Lundkvist's control unit, and the "first device") transmits the encrypted challenge X to device 60 (comparable to Lundkvist's portable unit, and the "second device").

265. Moreover, I also explained that Lundkvist's Y1 ("second signal") is derived from the first signal using the session key SK. I explained that:

$$Y1 = E_{sk}(f(D_{sk}(X)))$$

Where "E_{SK}" represents encryption using SK and "D_{SK}" represents decryption using SK and f is Lundkvist's transformation function. X, as previously discussed, is the first signal, but also the encrypted information O_RND.

266. In my explanation of the '977's limitation 1[c], I explained that this expression of Y1 as a function of both X and SK illustrated that Y1 was derived from X ("the first signal") and session key SK. With respect to the 977's limitation 1[c], I explained that session key SK was a secret known to the first device as required by that limitation. With respect to this limitation, SK is also a secret known to the second device, because SK is a shared secret. Maillard at [105]-[106].

Element 1[e]: provide the second signal to the first device after receiving the first signal, wherein the second signal is received by the first device.

267. I have previously explained in Section VI.E and with reference to Claim 1 of the ‘977 that it would have been obvious to a POSITA to combine Maillard, Davis, and Lundkvist. In incorporate that previous analysis into my analysis of the 564’ claim limitation 1[e].

268. In the Maillard-Davis-Lundkvist combination, Lundkvist’s encrypted challenge-response protocol collectively transmits the X challenge (“first signal”) from the controller unit (security module, or “first device”) to the portable unit (device 60 or “second device”). See annotated Figure 2 above. In response, the portable unit (device 60 or “second device”) transmits the encrypted response Y1 (“second signal”) back to the controller unit (security module or “first device”). Lundkvist at 8:11-23.

269. The response is received by the controller unit (security module or “first device”) for subsequent processing. Lundkvist at 8:23-28.

Element 1[f]: receive the protected content from the first device when the first device determines that the second signal is derived from the secret and a time between the sending of the first signal and the receiving of the second signal is less than a predetermined time.

270. Limitation 1[f] of the ‘977 patent is similar in language and content to this (‘564) Limitation 1[f]. Therefore, I incorporate my analysis of limitation 1[f]

of the ‘977 patent (¶¶194-204) into my analysis of this (‘564) Limitation 1[f] by reference.

271. In my analysis of limitation 1[f] of the ‘977 patent, I explained that a POSITA would have been motivated to combine Maillard, Davis, and Lundkvist. In the Maillard-Davis-Lundkvist system, the security module of Maillard (“first device”) does not transmit content unless the device 60 (“second device”) is in an expected location (e.g., “home network”), which is enforced by the Lundkvist timed challenge-response.

272. As I have already explained, this timed challenge-response performs two operations. As disclosed by Lundkvist, the protocol verifies the Y1 (“second signal”) is the expected encrypted value $E_{sk}(f(D_{sk}(X)))$ using the shared key SK (“the secret”) (“determines that the second signal is derived from the secret). The protocol also verifies that the round-trip time of the challenge response is within a predetermined time. Lundkvist at 8:19-28.

B. Claim 2: The second device of claim 1, wherein the secret is securely provided to the second device by the first device.

Claim 28: The second device of claim 1, the secret is known by the first device.

273. As I explained with respect to claim 1[b] of the ‘977 patent, the security module (“first device”) securely transmits the session key SK (“the secret”) to device 60 (“second device”). I incorporate my discussion of claim 1[b]

of the ‘977 patent (¶¶159-174) into this (‘564) claim 2 and (‘564) claim 28 by reference.

274. Because the security module (“first device”) securely transmits the session key SK (“the secret”) to device 60 (“second device”), the secret is securely provided to the second device by the first device.

275. Furthermore, by transmitting the secret to the second device from the first device, the secret is known to the first device.

C. Claim 5: The second device of claim 2, wherein the predetermined time is based on a communication system associated with the first device.

Claim 17: The second device of claim 1 , wherein the predetermined time is based on a communication system associated with the first device.

276. I have already explained with respect to claim 10 of the ‘977 patent that the predetermined time is based on a communication system associated with the first device. I incorporate my discussion of claim 10 of the ‘977 (¶¶213-217) into this (‘564) claim 5 and claim 17 by reference.

277. Thus, the combination Maillard-Davis-Lundkvist renders obvious claim 5 and claim 17.

D. Claim 6: The second device of claim 2, further comprising instructions arranged to receive the secret from the first device.

Claim 18: The second device of claim 1, further comprising instructions arranged to receive the secret from the first device.

278. As I explained with respect to limitation 1[b] of the '977 patent, the security module ("first device") securely transmits the session key SK ("the secret") to device 60 ("second device"). I incorporate my discussion of claim 1[b] of the '977 patent (¶¶159-174) into this ('564) claim 6 and ('564) claim 18 by reference.

279. Because the first device transmits the session key SK to the second device, the second device receives the secret from the first device.

280. Thus, the Maillard-Davis-Lundkvist combination renders obvious claim 6 and claim 18.

E. Claim 7: The second device of claim 2, wherein the second signal comprises the first signal modified by the secret.

Claim 19: The second device of claim 1, wherein the second signal comprises the first signal modified by the secret.

281. I have already explained with reference to limitation 1[c] of the '977 patent that Y1 ("the second signal") comprises X ("the first signal") modified by session key SK ("the secret") (e.g., $Y1 = G_{SK}(X)$). I incorporate my analysis of limitation 1[c] of the '977 patent (¶¶175-188) into this ('564) claim 7 and claim 19 by reference.

282. Thus, the combination of Maillard-Davis-Lundkvist renders obvious claim 7 and claim 19.

F. Claim 8: The second device of claim 2, wherein the secret comprises a random number.

Claim 20: The second device of claim 1, wherein the secret comprises a random number.

283. I have already explained with reference to claim 9 of the ‘977 patent that the session key SK (“the secret”) comprises a random number. I incorporate my analysis of claim 9 of the ‘977 patent (¶¶211-212) into this (‘564) claim 8 and claim 20 by reference.

284. Thus, the combination of Maillard-Davis-Lundkvist renders obvious claim 8 and claim 20.

G. Claim 9: The second device of claim 2, wherein the secret is encrypted with a public key.

Claim 21: The second device of claim 1, wherein the secret is encrypted with a public key.

285. I have already explained with reference to limitation 11[c] that the combination of Maillard-Davis-Lundkvist renders obvious encrypting the secret with a public key. I incorporate my analysis of limitation 11[c] of the ‘977 (¶¶175-188) into my analysis of this (‘564) claim 9 and claim 21.

286. Thus, the combination of Maillard-Davis-Lundkvist renders obvious claim 9 and 21.

H. Claim 10: The second device of claim 2, wherein the first signal comprises a random number.

Claim 22: The second device of claim 21, wherein the first signal comprises a random number.

287. I have already explained with reference to limitation 1[b] of the '977 patent that in the Maillard-Davis-Lundkvist combination that Lundkvist's X ("first signal") is the encrypted random number O_RND. I incorporate my analysis of limitation 1[b] of the '977 patent (¶¶159-174) into this ('564) claim 10 and claim 22.

288. Therefore, the combination of Maillard-Davis-Lundkvist renders obvious claim 10 and claim 22.

I. Claim 14: The second device of claim 2, wherein the secret is used for generating a secure channel between the first device and the second device.

Claim 25: The second device of claim 1, wherein the secret is used for generating a secure channel between the first device and the second device.

289. I have already explained with reference to limitations 1[e] and 1[f] of the '977 patent that the session key SK is used to generate a secure authenticated channel between the security module, or the security module in combination with a device 60, ("first device") to a device 60 ("second device"). I incorporate my analysis of limitation 1[e] and 1[f] of the '977 patent (¶¶190-203) into this ('564) claim 14 and claim 25 by reference.

290. Thus, the combination of Maillard-Davis-Lundkvist renders obvious of claim 14 and claim 25.

IX. UNPATENTABILITY OF THE CHALLENGED CLAIMS OF THE ’564 PATENT FOR GROUND 2 (MAILLARD, DAVIS, CHAUM AND LUNDKVIST)

291. I have explained in Section VIII that the ‘564 patent is rendered obvious by the combination of Maillard, Davis, and Lundkvist. The ‘564 patent is also rendered obvious by the combination of Maillard, Davis, Chaum, and Lundkvist.

292. I have already explained in Section VI.E that a POSITA would have been motivated to combine Maillard, Davis, Chaum, and Lundkvist and that it would have been obvious to do so. I incorporate this analysis into this section by reference.

293. As previously explained, Chaum, like Lundkvist, describes a mechanism for symmetric-encryption based challenge-response protocols. Chaum teaches a slightly different version from Lundkvist. Whereas Lundkvist sends an encrypted challenge and an encrypted response that is decrypted, Chaum discloses sending an unencrypted random challenge and an encrypted response that can be directly compared to a precomputed value. Chaum’s challenge-response is faster, both in not requiring the challenge to be encrypted, and in permitting the response to be compared to a pre-computed value. See paragraph XX.

294. In the following paragraphs, I detail how this Maillard-Davis-Lundkvist-Chaum combination meets various claims. Because almost all of the explanation mirrors explanation of Section VIII, I will only discuss the small number of limitations and claims that require additional explanation.

A. Claim 1

Element 1[a]: provide a certificate to the first device prior to receiving a first signal, wherein the first signal is sent by the first device, wherein the certificate is associated with the second device:

295. I have already explained with reference to this same limitation that it would have been obvious in view of Maillard combined with Davis and Lundkvist. I incorporate my analysis regarding both motivation and the combination of Maillard-Davis-Lundkvist by reference.

296. As I explained previously, Davis details a small number of challenge-response protocols that use asymmetric cryptography, but these are listed as examples only. Davis at 11:8-13:16. Davis clearly emphasizes the exemplary nature of these illustrations. "The periodic Challenge/Response message may be performed in a number of ways as shown in Figures 6A-6C. ***These are shown purely for clarification***; other means of authentication may be used without deviating from the spirit of this invention." Davis at 12:10-13, emphasis added.

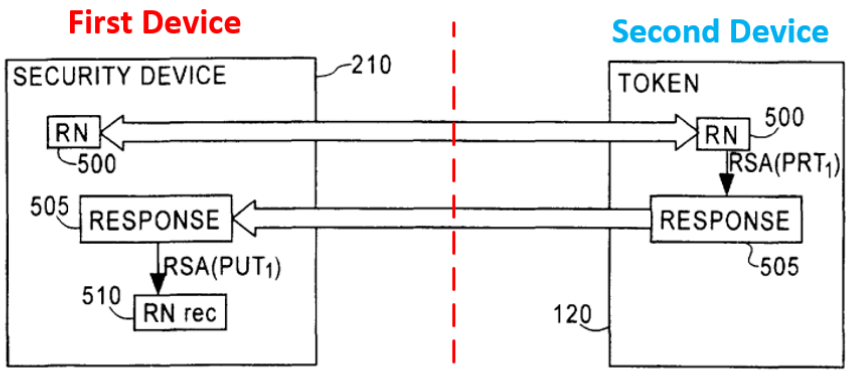


FIG. 6A

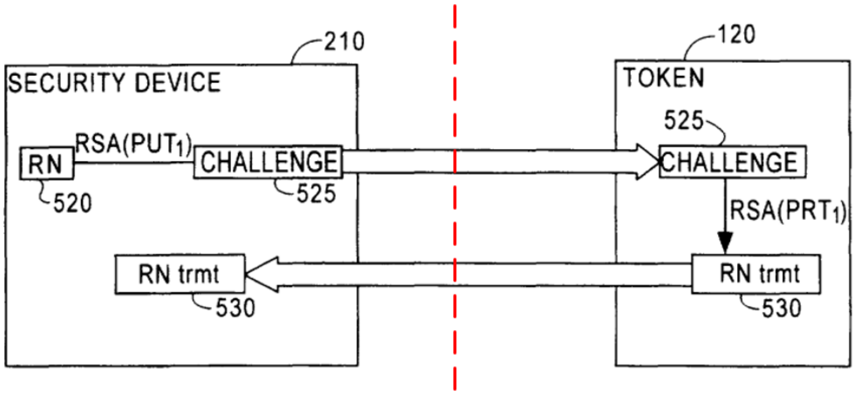


FIG. 6B

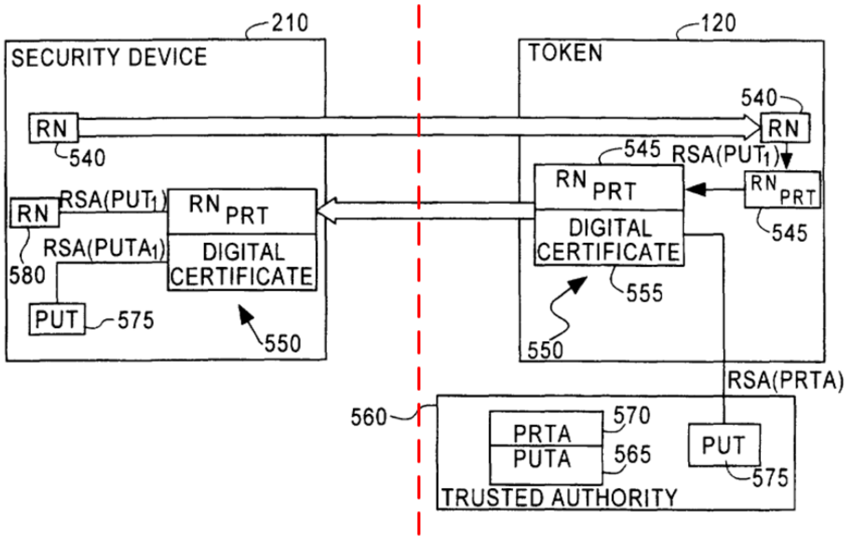


FIG. 6C

297. As also previously explained, Lundkvist teaches a challenge-response protocol that uses an encrypted challenge and an encrypted response. The

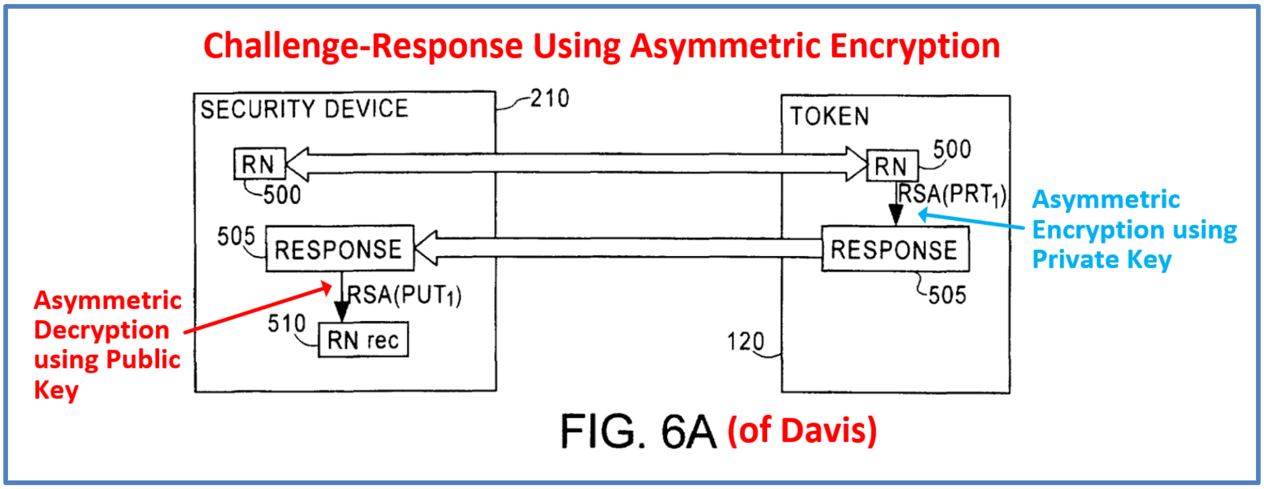
encrypted challenge is decrypted and processed, and the result of that process is encrypted and sent back as an encrypted response. Lundkvist describes decrypting the encrypted response for analysis. Lundkvist at 8:19-28.

298. Chaum discloses yet another challenge-response protocol that, while also based on symmetric encryption, has a few construction differences from Lundkvist. In Chaum's construction, the challenge is *not* encrypted. The response is encrypted, but does not need to be decrypted when received by the challenger. Instead, the challenging party can compute the same response in advance and compare the precomputed and received (encrypted) responses directly. Chaum, 11:11-26.

299. A POSITA would have found it obvious to use Chaum's symmetric-encryption challenge-response instead of Lundkvist's symmetric challenge-response because Chaum's variant is faster. Chaum's variant is compatible with Lundkvist's timing mechanisms and appropriate for distance measurement.

300. Moreover, Chaum's construction is suggested by Davis's example asymmetric challenge-response. That is, in one of Davis's examples, a challenge is sent unencrypted, the response is calculated, and the response sent back encrypted by *an asymmetric key*. Davis at 12:13-22, FIG 6A. It would have been obvious to a POSITA that Chaum's recipe for using a symmetric response would perform the same function.

301. Davis's original FIG 6A is reproduced below:


Challenge-Response Using Asymmetric Encryption
SECURITY DEVICE
210
TOKEN
RN
500
RN
500
RSA(PRT1)
Asymmetric Encryption using Private Key
505
RESPONSE
RESPONSE
505
Asymmetric Decryption using Public Key
RSA(PUT1)
510
RN rec
120
FIG. 6A (of Davis)

302. Davis combined with Chaum is a very minor modification and is visualized in the ***modified*** and annotated FIG 6A below.

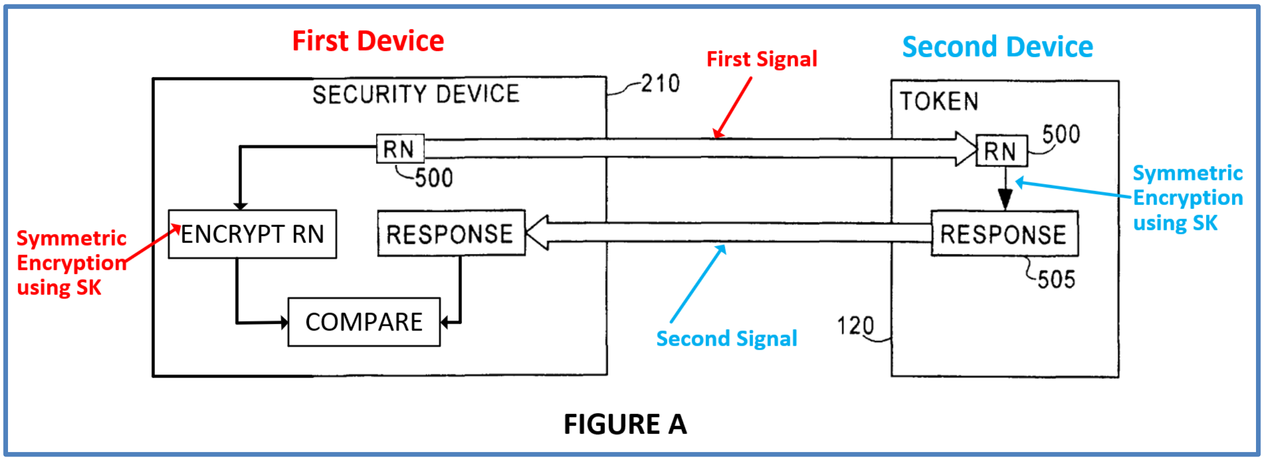


FIGURE A

303. In the Maillard-Davis-Lundkvist-Chaum combination, the transmission of the unencrypted random number is the first signal.

Element 1[d]: create a second signal, wherein the second signal is derived from a secret known by the second device.

304. As I previously explained with reference to limitation 1[c], in the Maillard-Davis-Lundkvist-Chaum combination, a random number is sent as the challenge ("first signal"). I incorporate my discussion of limitation 1[c] by reference.

305. After the second device receives the unencrypted challenge, it encrypts it with the random number RN (analogous to the session key SK, or "the secret") to produce the response ("second signal"). Therefore, the second signal (the "response") is derived from a secret that was known by the second device because it was encrypted using SK that was known to the second device. The second signal is sent back to the first device for processing.

Element 1[e]: provide the second signal to the first device after receiving the first signal, wherein the second signal is received by the first device.

306. As I previously explained with reference to limitation 1[c] and 1[d], in the Maillard-Davis-Lundkvist-Chaum combination, a random number is sent as the challenge ("first signal"). The challenge is encrypted with the random number RN (session key SK, or "the secret") to produce the response ("second signal"). The second signal is sent back to the first device. I incorporate my discussion of limitation 1[c] and limitation 1[d] by reference.

307. The response ("second signal") in Chaum, like with Lundkvist, is created based on the values of the challenge ("first signal"). It is, therefore, only

sent after receiving the first signal. The challenge is sent back to the challenging device ("first device").

Element 1[f]: receive the protected content from the first device when the first device determines that the second signal is derived from the secret and a time between the sending of the first signal and the receiving of the second signal is less than a predetermined time."

308. I have already explained that a POSITA would have found this limitation to be obvious in the Maillard-Davis-Lundkvist combination and I incorporate this explanation by reference. My analysis of Maillard-Davis-Lundkvist is identical to Maillard-Davis-Lundkvist-Chaum except for "determining that the second signal is derived from the secret."

309. Chaum's method for "determining that the second signal is derived from the secret" is conceptually similar to Lundkvist's, but with a slight modification. Lundkvist proposes sending an encrypted response that is subsequently decrypted and processed. Lundkvist at 8:19-28. Chaum teaches that for a predictable response, the encrypted response can be computed (or precomputed) and compared directly. Chaum, 11:11-26.

B. Claim 3: The second device of claim 2, wherein determining that the second signal is derived from the secret comprises: modifying the first signal, wherein the modifying requires the secret; and determining that the modified first signal is identical to the second signal.

Claim 4: The second device of claim 2, wherein determining that the second signal is derived from the secret comprises: modifying the first signal; and determining that the modified first signal is identical to the second signal.

Claim 15: The second device of claim 1, wherein determining that the second signal is derived from the secret comprises: modifying the first signal, wherein the modifying requires the secret; and determining that the modified first signal is identical to the second signal.

Claim 16: The second device of claim 1, wherein determining that the second signal is derived from the secret comprises: modifying the first signal; and determining that the modified first signal is identical to the second signal.

310. I have already explained that a POSITA would have found the limitations of claim 1 obvious in view of the Maillard-Davis-Lundkvist-Chaum combination. I incorporate this previous analysis by reference.

311. I have already explained that the response ("second signal") described by Chaum is the challenge ("first signal") encrypted by the random number RN (the session key SK, or "the secret"). Thus, the second signal is derived from the first signal and the secret. Moreover, the second signal requires the secret or it cannot be created.

312. I have also explained that Chaum determines that the response ("second signal") is derived from the secret by pre-computing the expected value of the response by encrypting the challenge value ("modifying the first signal") and then comparing that the precomputed response and received response are the same ("determining that the modified first signal is identical to the second signal").

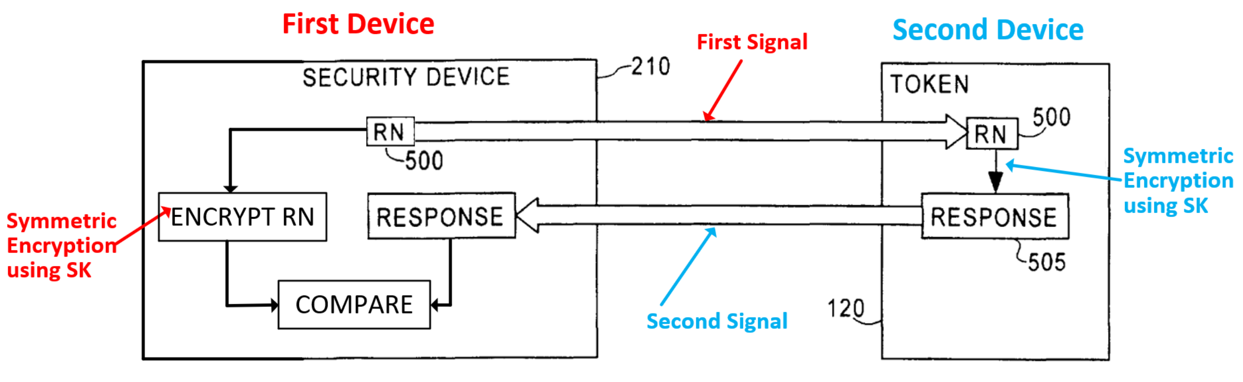


FIGURE A

313. Thus Maillard-Davis-Lundkvist-Chaum renders obvious claims 3, 4, 15, and 16.

C. Claim 7: The second device of claim 2, wherein the second signal comprises the first signal modified by the secret.

Claim 19: The second device of claim 1, wherein the second signal comprises the first signal modified by the secret.

314. I have already explained that a POSITA would have found the limitations of claim 1 obvious in view of the Maillard-Davis-Lundkvist-Chaum combination. I incorporate this previous analysis by reference.

315. I have already explained that the response ("second signal") described by Chaum is the challenge ("first signal') encrypted by the random number RN (the session key SK, or "the secret"). Thus, the second signal is the first signal encrypted ("modified") by the secret.

316. Thus, claim 19 is rendered obvious by Maillard-Davis-Lundkvist-Chaum.

X. UNPATENTABILITY OF THE CHALLENGED CLAIMS OF THE '564 PATENT FOR GROUND 2 (MAILLARD, DAVIS, LUNDKVIST AND SCHNEIER)

317. I have already opined that certain challenged claims of the '564 patent are obvious in view of Maillard-Davis-Lundkvist in Section VI.E. I incorporate that analysis into this section by reference.

A. Claim 11: The second device of claim 2, wherein [creating] the second signal comprises an XOR operation of the first signal with the secret.

Claim 23: The second device of claim 1, wherein [creating] the second signal comprises an XOR operation of the first signal with the secret.

318. I have already explained that Schneier disclosed using XOR as a weak, but fast, mechanism for encrypting data with reference to limitation 1[c] of the '977 patent. I incorporate this discussion of limitation 1[c] of the '977 patent (¶¶175-188) into this (564's) claims 11 and 23. The analysis remains the same; the only difference is that, in this section, Schneier is a named reference.

319. Thus, I conclude that Maillard, Davis, Lundkvist, and Schneier render claims 11 and 23 obvious.

XI. DECLARATION

320. I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; that these statements were made with knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under 18 U.S.C. § 1001; and further that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

321. I declare under the penalty of perjury that the foregoing is true and correct.

Executed on 10th day of February 2021 in Austin, TX .

Seth James Nielson, Ph.D.

APPENDIX A

CRIMSON VISTA
CYBERSECURITY ENGINEERING

Seth James Nielson's Vita

(512) 387-4310
seth@crimsonvista.com

February 2021

Academic Degrees

2010	**Ph.D. Computer Science,** Rice University, Houston, TX
2004	**M.S. Computer Science,** Brigham Young University, Provo, UT
2000	**B.S. Computer Science,** Brigham Young University, Provo, UT

Current Appointments

Founder and Chief Scientist	Crimson Vista, Inc.
Founder and Partner	Source Code Discovery, LLC
Adjunct Assistant Professor	University of Texas at Austin Computer Science
Cybersecurity Fellow	Robert Strauss Center for International Security and Law

Professional Advising

Cyberlab Advisory Board	The SEED School of Maryland
Advisory Board	Syccure Inc.

Grants and Research Awards

Mitigation of Ransomware, DoD 2018.B STTR solicitation (Army), Topic: A18B-T010, Crimson Vista, $150,000 (2019)

An Enabling Repository of Cryptographic Knowledge (Co-PI), Cisco Grant, Johns Hopkins University Information Security Institute, $150,000 (2017-2021)

Certifications

CISSP

Subject Matter Expertise and Selected Projects

Applied Cryptography

"Crypto Done Right", Cryptographic Knowledge Base with Cisco (Johns Hopkins, 2017-2020)

Simulated implementation of Vehicle-to-Vehicle authentication (Crimson Vista, 2018-2019)

Cryptographic protocol analysis and design for Syccure Inc. (Crimson Vista, 2017-)

Timing attacks on Hardware Security Modules with Diamond Key Security (Johns Hopkins, 2018)

Technical consulting for U.S. Dpt. of Justice (Antitrust Division, Transportation, Energy & Agriculture) (Crimson Vista, 2018)

Aircraft/drone anti-collision protocol security with OnBoard Security (Johns Hopkins, 2017)

Cryptographic communication library for Security First Corp (ISE, 2010-2011)

Encryption library, file system encryption, GPU for Security First Corp (ISE, 2005-2011)

IoT Devices and IoT Security

Contributor, reviewer for the Institute of Assured Autonomy (Johns Hopkins, 2019)

IoT data aggregation security with Armored Things (Johns Hopkins, 2018-2019)

Automated IoT device profiling with the Physics Lab (Johns Hopkins, 2018)

Physical forensics from IoT devices (Johns Hopkins 2017)

Network Security

Information risk assessment with respect to data breach (Crimson Vista, 2020-)

Data breach analysis with respect to password authentication (Crimson Vista, 2020-)

Network security curriculum design for Law and Policy (University of Texas at Austin, 2020-)

Network security curriculum design for undergraduates (University of Texas at Austin, 2020-)

Network security curriculum design (Johns Hopkins, 2013-2019)

Security analysis of financial services vendor for Fortune-100 client (Crimson Vista, 2018-2019)

Analysis of secure e-mail transmission for confidential client (Crimson Vista, 2018-2019)

Cloud storage system security evaluation for confidential client (Harbor Labs, 2015)

Secure gateway security and compliance evaluation for SecurityFirst Corp (Harbor Labs, 2015)

Gateway caching security/privacy engineering (Google, 2005)

Malware and Viruses

Ransomware recovery research and development (Crimson Vista, 2018-2019)

Advanced malware and malware defenses analysis (Johns Hopkins, 2018)

Malware Analysis for Confidential Client (Crimson Vista, 2016)

Privacy

Development of an auditing tool for cookies and other web tracking (Crimson Vista, 2020-)

Audit of a large hospital network for cookies and web tracking (Confidential client, 2020-)

Development of a privacy curriculum for data architects (Crimson Vista, 2019)

Anonymization of threat indicators with the Applied Physics Lab (Johns Hopkins, 2017)

Analysis of anti-piracy vendor for Center for Copyright Information (Harbor Labs, 2013-2014)

Miscellaneous Security Technologies

Blockchain technology analysis for a start-up (Confidential Client, 2019-2020)

Analysis of cyber security norms for a confidential Fortune 100 client (Crimson Vista, 2018-2019)

Analysis of vulnerabilities in portable chemical manufacturing system (Johns Hopkins, 2018)

Open-Source Contribution and Sponsorship for The PyPy Project (Crimson Vista, 2018-2019)

Blockchain and smart contract design for confidential start-up (Crimson Vista, 2018)

Medical device system security evaluation for confidential client (Harbor Labs, 2015)

Software Engineering and Software Analysis

Software theft analysis and tool design for various clients (Crimson Vista, 2018-)

Cyber insurance investigations for Clyde & Co (Crimson Vista, 2018-2019)

Distributed code coverage analysis tool for a confidential Fortune 100 client (ISE, 2011)

Software engineering for Metrowerks Inc. (2001-2003, formerly Lineo Inc.)

Publications

Books

Seth James Nielson, Christopher K. Monson, Practical Cryptography in Python, Learning Correct Cryptography by Example. (October 2019).

Refereed Publications

Joseph Kosturko, Eric Schlieber, Sean Futch, Seth James Nielson, *Cracking a Continuous Flow Reactor: A Vulnerability Assessment for Chemical Additive Manufacturing Devices*. In *Proceedings of the 2018 IEEE International Symposium on Technologies for Homeland Security*. (September 2018)

Chanyang Shin, Prerit Chandok, Ran Liu, Seth James Nielson, Timothy Leschke, *Potential Forensic Analysis of IoT Data: An Overview of Amazon Echo, Z-wave, and Home Router Data Extraction and Analysis*. In *Proceedings of the 2017 IEEE International Conference on Internet of Things (iThings)*, Exeter, UK, pp. 705-710 (June 2017).

Seth James Nielson, *PLAYGROUND: Preparing Students for the Cyber Battleground*. *Computer Science Education*, volume 26, issue 4, pp. 255-276, (January 2017).

Seth James Nielson and Charles D. Knutson, *Design Dysphasia and the Design Patterns Maintenance Cycle*. *Information & Software Technology,* volume 48, number 8, pp. 660- 675, (August 2006)

Seth James Nielson, Scott S. Crosby, and Dan S. Wallach, *A Taxonomy of Rational Attacks*. In

Proceedings of the Fourth International Workshop on Peer-to-Peer Systems (IPTPS '05), Ithaca, New York, (February 2005)

Seth J. Nielson and Charles D. Knutson. *OO++: Exploring the Multiparadigm Shift*. *Proceedings of the Workshop on Multiparadigm Programming with Object-Oriented Languages* (MPOOL 2004), Oslo, Norway, (June 2004)

Rob Kunz, Seth Nielson, Mark Clement, Quinn Snell, *Effective Bandwidth for Traffic Engineering*, *Proceedings of the IEEE Workshop on High Performance Switching and Routing (HPSR 2001),* Dallas, TX, (May 2001)

Testimony Before Government Bodies

Testimony in support of *Maryland SB 151/HB 211 Criminal Law - Crimes Involving Computers – Ransomware*. Testimony given to the Judicial Proceedings Committee of the Maryland Senate and the Judiciary Committee of the Maryland House of Delegates (January/February 2019)

Invited Talks, Panels, and Technical Training

Seth James Nielson, *Securing Your Data Assets Against Hackers*, 2021 Enterprise Data World, Boston, MA. (April 2021)

Seth James Nielson, *The Cybersecurity of IP*, 2020 Essential Cybersecurity Law, Austin, TX (Virtual). (July 2020)

[**CANCELED DUE TO COVID 19**] Seth James Nielson and Ellie Daw, *Correct Cryptography in Python*, a Tutorial for Cryptography Beginners, PyCon 2020, Philadelphia, PA. (April 2020)

[**CANCELED DUE TO COVID 19**] Seth James Nielson, *Protecting Your Data Assets*, 2020 Enterprise Data World, Boston, MA. (March 2020)

Seth James Nielson, *Security and Privacy for Data Architects*, 2019 Data Architecture Summit, Chicago, IL (October 2019)

Seth James Nielson, Crypto Done Right, One Year In. Lessons Learned and Next Steps. Presented in the International Cryptographic Knowledge Base 2019 (ICMC 2019), Vancouver, Canada (May 2019)

Panel Discussion, *Hardening US Unmanned Systems Against Enemy Counter Measures*. Accepted invitation to 7th Annual DoD Unmanned Systems Summit, Alexandria, VA (April 2019).

Seth James Nielson, *A Gentle Introduction to Blockchain*. 2019 Enterprise Data World, Boston, MA (March 2019).

Seth James Nielson, *A Gentle Introduction to Blockchain*. 2018 Data Architecture Summit, Chicago, IL (October 2018).

Seth James Nielson, *Detecting Malicious Sandboxes*. Workshop on Defensive Deception and Trust in Autonomy, San Diego, CA (August 2018).

Seth James Nielson and Debra Baker, *Towards a Crowd-Sourced Cryptographic Knowledge Base*. Presented in the International Cryptographic Knowledge Base 2018 (ICMC18), Ottawa, Canada (May 2018).

Seth James Nielson, *The PyPy Sandbox*. Presented in the National Centers of Academic Excellence Tech Talk, Online (March 2018).

Theses

PhD Thesis: Designing Incentives for Peer-to-Peer Systems (defended 10/2009)
Master's Thesis: OO++ Design Patterns, GOF Revisited (defended 8/2004)

Technical Reports

Seth James Nielson and Dan S. Wallach, The BitTorrent Anonymity Marketplace, arXiv Technical Report 1108.2718, (August 2011)

Seth James Nielson, Caleb E. Spare, and Dan S. Wallach, Building Better Incentives for Robustness in BitTorrent, arXiv Technical Report 1108.2716, (August 2011)

Seth James Nielson, Seth J. Fogarty, and Dan S. Wallach, *Attacks on Local Searching Tools*, arXiv Technical Report 1108.2704 (Originally produced in December, 2004, available on arXiv as of August 2011)

White Papers and Trade Publications

Aviel D. Rubin, Seth J. Nielson, Christopher K. Monson, *Evaluation of the MarkMonitor AntiPiracy System*, Produced for the Center for Copyright Information (December 2013)

Aviel D. Rubin, Seth J. Nielson, Sam Small, Christopher K. Monson, *Guidelines for Source Code Review in Hi-Tech Litigation*, Harbor Labs White Paper (September 2013)

Graduate Advising

Capstones for Masters of Science in Security Informatics (Johns Hopkins)

Weizhou Wang, Runjie Zhang, *Research on defense next-generation malware on the Windows platform* (December 2018)

Yu-Tsern Jou, Ying Liu, Menghan Bai, *Open Source HSM Side Channel Analysis* (December 2018)

Bowen Shi, Hong Ma, Mengqi Qin, *Side Channel Attack on HSM Based on Machine Learning* (December 2018)

Dylan Richmond, Matthew Shonman, Jingcheng Yang, *An Exploration of the Usability of HTTPS* (December 2018)

Steven Cheng, Venkata Aditya Bollapragada, and Antara Sargam, *Using Selective RAM Journaling to Fight Ransomware* (December 2018)

Yongqiang Fan, Haiwen Sun*, Fault Tolerance System for IoT* (December 2018)

Weike Chen, Harry Luo, Prashanth Venkateswaran, *IoT Discovery* (August 2018)

Joseph “Jay” Kosturko, Eric Schlieber, Sean Futch, *Cracking a Continuous Flow Reactor: A Vulnerability Assessment for Chemical Additive Manufacturing Devices* (May 2018)

Chao Lei, Wenjun Li, *Anti-Honeypot Detection in Advanced Botnet Attacks* (December 2017)

Ritvik Sachdev, Purushottam Kulkarni, Praveen Malhan, *Securing ADS-B Based Airborne Collision Avoidance Systems* (December 2017)

Ningyuan Bao, Mengying Hu, *Security-Testing-Orientated Internet of Things(IoT) Simulator* (December 2017)

Zehuan Li, Shanshan Yang, Liangjia Fu, *AIS Data De-anonymization* (December 2017)

Chanyang Shin, Prerit Chandok, Aurin Chakravarty, *Forensic Data Collection from IoT Devices* (December 2017)

Kevin Manzotti, Kashif Memon, Rahul Durgad, *Replication of CryptoDrop* (December 2016)

Harshneel More, Jingmiao Wang, Yuanqi Zhu, *Detecting XSS attacks using BRO IDS* (December 2016)

Asmaa Aljohani, Gyan Namdhari, Yue Zhu, *Feasibility, Security and Privacy Analysis of EMVCo Payment Tokenization Technology for Identity Enabled Transactions* (December 2016)

Richard Eaton, *The Emperor Has No Friends: Identifying Botnet Customers and Mapping Out Botnets on Twitter* (May 2015)

Jingru Chen, Yaning Liu, Yifan Yu, Zhiyue Zu, *Investigating the Heartbleed Vulnerability* (2015)

Kartik Thapar, *Security Techniques for Developing iOS Applications* (February 2015)

Jie Feng, Jianxiang Peng, Likai Zhang, *De-anonymizing BitCoin* (January 2014)

Teaching

University of Texas at Austin

Cybersecurity Tech/Law Policy (2020-)
Network Security and Privacy (2020-)

Johns Hopkins University

Advanced Network Security (2017 - 2019)
Network Security (2014 – 2019)

Rice University

Data Structures and Algorithms (Spring 2008)

Professional Service

International Cryptographic Module Conference, 2020 Program Committee
International Cryptographic Module Conference, 2019 Program Committee
International Cryptographic Module Conference, 2018 Program Committee

Patents

Co-inventor: Orsini, R. 2014. Systems and methods for security data in motion. U.S. Patent 8,745,372 filed November 24, 2010 and issued June 3, 2014.

Co-inventor: Orsini, R. 2014. Systems and methods for security data in motion. U.S. Patent 8,745,379 filed August 20, 2012 and issued June 3, 2014.

Co-inventor: O'Hare, R. 2014. Systems and methods for security data. U.S. Patent 8,677,148 filed January 27, 2012 and issued March 18, 2014.

In the News

"Johns Hopkins Researchers and OnBoard Security Team Up to Protect Drones," *Robotics Tomorrow*, 3/16/2018

Academic Awards

Brown Fellowship
John and Eileen Tietze Fellowship

Employment History

University of Texas at Austin	
2020-	**Adjunct Assistant Professor**
Source Code Discovery, LLC.	
2019-Present	**Founder and Partner**
Crimson Vista, Inc.	
2016-Present	**Founder and Chief Scientist**
Johns Hopkins Applied Physics Lab	
2018-2019	**Senior Professional Staff (Temp/On-Call)**
Johns Hopkins University	
2016-2019	**Director of Advanced Research Projects**
2015-2019	**Adjunct Associate Research Scientist**
2014	**Lecturer**
Harbor Labs, LLC	
2014-2015	**Principal**
2011-2014	**Research Scientist**
Independent Security Evaluators	
2010-2011	**Senior Security Analyst**

2005-2009 **Security Analyst**

Google, Inc.
2005 **Summer Intern**

Metrowerks (Formerly Lineo, Inc.)
2001-2003 **Software Engineer II**

Expert Testimony

On behalf of Magic Micro in Monsoon v. Magic Micro (2020-Present)

Counsel: Gutnicki, LLP
Venue: International Centre for Dispute Resolution
Subject Matter: Blockchain, file storage
Reports: 1 declaration submitted October 2020
1 declaration submitted November 2020
Testimony: In arbitration December 2020

On behalf of Juniper in Juniper Networks Inc. v. Huawei Digital Technologies Co. (2020-Present)

Counsel: Irell and Manella, LLP
Venue PTAB Petition for Inter Partes Review
Subject Matter: Network perimeter defense against malware
Reports: 1 declaration submitted June 2020

On behalf of DivX in Netflix v. DivX (2020-Present)

Counsel: Lowenstein & Weatherwax, LLP
Venue: PTAB Petition for Inter Partes Review
Subject Matter: Encryption of streaming media
Reports: 1 declaration submitted May 2020
1 declaration submitted December 2020
Deposition January 2021

On behalf of Juniper in Juniper Networks Inc. v. Implicit LLC (2019-2020)

Counsel: Irell and Manella, LLP
Venue: PTAB Petition for Inter Partes Review
Subject Matter: Packet classification
Reports: 1 declaration submitted February 2020

On behalf of the Dealership class in Dealer Management Systems Antitrust Litigation (2019-Present)

Counsel: Milberg Phillips Grossman LLP
Venue: Case No. 18-cv-00864 (Northern District of Illinois)
Subject Matter: Cybersecurity, risk assessment, third-party access
Reports: 1 report submitted November 2019
Deposition: January 2020

On behalf of Symantec in the Trustees of Columbia University in the City of New York v. Symantec Corporation. (2018-Present)

Counsel: Quinn Emanuel Urquhart & Sullivan, LLP

Venue: Case No. Civil Action No. 3:13-cv-00808-JRS (Eastern District of Virginia)
Subject Matter: Deception technologies, Honeypots
Reports: 1 report submitted October 2019
Deposition: January 2020

On behalf of Proofpoint in Proofpoint Inc. v. Vade Secure (2019-Present)
Counsel: Quinn Emanuel Urquhart & Sullivan, LLP
Venue: Case No. 3:19-cv-04238 (Northern District of California)
Subject Matter: Cloud-based email, malware detection, AI
Reports: 1 report submitted September 2019, 1 report submitted December 2019

On behalf of Bitdefender in Finjan Inc. v. BitDefender Inc. (2018-Present)
Counsel: Susman Godfrey LLP
Venue: Case No. 4:17-cv-4790-HSG (Northern District of California)
Subject Matter: Web threat detection, malware, gateways
Reports: 1 report submitted July 2019, 1 report submitted August 2019
Depositions: October 2019

On behalf of HTC in Koninklijke Philips N.V., US Philips Corporation v. HTC Corp., HTC America Inc. (2019-Present)
Counsel: Perkins Coie LLP
Venue: Case No. 4:18-cv-01887-HSG (Northern District of California)
Subject Matter: Cryptographic protocols
Reports: 1 report submitted May 2019, 1 report submitted June 2019
Depositions: August 2019

On behalf of Juniper in Juniper Networks Inc. v. Finjan Inc. (2018-Present)
Counsel: Irell and Manella, LLP
Venue: PTAB Petition for Inter Partes Review
Subject Matter: Malware, firewall/gateway, network security
Reports: 4 declarations submitted October 2018
Depositions: June 2019

On behalf of Redbox in Disney Enterprises, Inc., LucasFilm Ltd, LLC, and MVL Film Finance LLC v. Redbox Automated Retail, LLC. (2018)
Counsel: Robins Kaplan LLP
Venue: Case No. 2:17-cv-08655-DDP (AGRx) (Central District of California, Western Division)
Subject Matter: DRM, cloud security, media streaming
Reports: 1 declaration submitted May 2018

On behalf of Trend Micro in Trend Micro, Inc. v. Security Profiling, LLC (2017-2018)
Counsel: The Marbury Law Group, PLLC
Venue: Inter Partes Review Case No. IPR2017-02191 and IPR2017-02192
Subject Matter: Automated security patches
Reports: 2 declarations submitted September 2017
Depositions: June 2018

On behalf of TeleSign in Twilio Inc. v. TeleSign Corporation (2016-Present, multiple cases)

Counsel: Shook, Hardy & Bacon LLP
Venue: Inter Partes Review Case No. IPR2016-01688, IPR2016-00360, IPR2017-01976 IPR2017-01977, IPR2017-01978
Case No. 5:16-cv-6925-LHK (Nor. District of California, San Jose Div.)
Subject Matter: Telecommunications security
Reports: 1 declaration submitted August 2016, May 2017, and July 2017
Depositions: November 2016, July 2017, August 2017

On behalf of Blue Coat in Finjan Inc. v. Blue Coat Systems LLC (2017-2018)
Counsel: Originally Wilson, Sonsini, Goodrich and Rosati PC; later Morrison & Foerster LLP
Venue: Case No. 15-cv-03295-BLF-SVK (Nor. District of California, San Div.)
Subject Matter: Firewalls, gateway, security devices, malware
Reports: 1 report submitted April 2017
Depositions: April 2017
In Court: Tech tutorial February 2017, expert testimony November 2017

On behalf of Sedosoft in Sedosoft Inc. v Mark Burchett LTD and NFSx9 LLC (2016)
Counsel: McInnes and McLane LLP
Venue: Case No. Civil Action No. 1:15-cv-10244-RGS (District of Massachusetts)
Subject Matter: Code theft
Reports: 1 report submitted May 2016 and August 2016

On behalf of USAA in Asghari-Kamrani, et al. v. United Services Automobile Association, Inc.
Counsel: Fish and Richardson PC
Venue: Inter Partes Review Case No. IPR2015-01842, and CBM2016-00063, CBM2016-00064
Subject Matter: Applied cryptography, authentication
Reports: 1 report submitted September 2015, 1 report submitted April 2016
Depositions: March 2017

On behalf of WTS Paradigm in WTS PARADIGM, LLC v EDGEAQ, LLC (2015-2016)
Counsel: Quarles & Brady LLP
Venue: Case No. 3:15-CV-330 (Western District of Wisconsin)
Subject Matter: Software-assisted product configuration, presentation, and ordering
Reports: 1 report submitted March 2016
Depositions: April 2016

On behalf of Mr Leon Stambler in Mr. Leon Stambler v Mastercard IPR (2015-2016)
Counsel: Flachsbart & Greenspoon, LLC
Venue: CBM2015-00044
Subject Matter: Authentication, authentication codes
Reports: 1 declaration submitted October 2015
Deposition: December 2015

On behalf of Trusteer/IBM in Trusted Knight Corporation v. International Business Machines Corporation and Trusteer, Inc. (2015)
Counsel: Quinn Emanuel Urquhart & Sullivan LLP
Venue: C.A. No. 14-1063 LPS-CJB (District of Delaware)

Subject Matter:	Key logging, protections against
Reports:	1 declaration submitted July 2015

On behalf of Sensus in Sensus USA, Inc. v. Certified Measurement, LLC (2015)

Counsel:	Feldman Gale PA
Venue:	Inter Partes Review Case No. IPR2015-01262, IPR2015-01311, IPR2015-01439, IPR2015-01454 Case number 3:14-cv-01069 (District of Connecticut)
Subject Matter:	Applied cryptography, certification of data
Reports:	1 declaration submitted June 2015, 4 declarations submitted July 2015

On behalf of Chad Eichenberger in Chad Eichenberger v. ESPN (2015)

Counsel:	Edelson PC
Venue:	Case No. 2:14-cv-00463 (Western District of Washington)
Subject Matter:	Consumer privacy
Reports:	1 declaration submitted January 2015

On behalf of Optimum Content Protection in Microsoft Corporation v. Optimum Content Protection LLC. (2014)

Counsel:	Sidley Austin LLP
Venue:	Inter Partes Review No. IPR2015-00048
Subject Matter:	DRM, data streaming, protection
Reports:	1 declaration submitted October 2014

On behalf of Fortinet in Fortinet v. Sophos (2014-2015, multiple cases)

Counsel:	Quinn Emanuel Urquhart & Sullivan LLP
Venue:	Inter Partes Review No. IPR2015-00618 Case No. 3:13-cv-05831-EMC (DMR) (Nor. District of California, SF div.) Case No. 1:14-cv-00100-GMS (District of Delware)
Subject Matter:	Antivirus, anti-malware
Reports:	2 declarations submitted September 2014, 1 submitted September 2015 1 report submitted October 2015, November 2015
Depositions:	October 2014, October 2015
In Court:	Tech tutorial December 2015

On behalf of Afilias in Afilias PLC v Architelos Inc and Alexa Raad (2015)

Counsel:	Haynes Boone LLP
Venue:	Case No. 1:15-cv-00014-LMB-JFA (Eastern District of Virginia)
Subject Matter:	Domain name system anti-abuse
Reports:	1 report submitted April 2015, May 2015
Depositions:	June 2015
In Court:	Expert testimony August 2015

On behalf of Telit in M2M v. Motorola, Telit (2015-2016)

Counsel:	Pearl Cohen Zedek Latzer Baratz LLP
Venue:	Case No. 12-033-RGA (District of Delaware)
Subject Matter:	Authentication
Reports:	1 declaration submitted May 2014, 1 report submitted July 2014, August 2014

Depositions: June 2015

On behalf of Rmail/Rpost in Rmail Limited, v. Amazon.com, Inc., and Paypal and Rmail Limited, Rpost Communications Limited, and Rpost Holdings Inc., v.Docusign, Inc. (2013-2014)

Counsel: Hudnell Law Group PC
Venue: Case No. 2:10-CV-258-JRG (Lead Case)
Case No. 2:11-CV-299-JRG (Member Case) (E.D. Texas)
Subject Matter: Email security
Reports: 2 reports April 2013, 1 declaration June 2013
Depositions: May 2013 (2 days)

On behalf of Via Vadis in Via Vadis v. Skype (2012-2014)

Counsel: WTP Law
Venue: Via Vadis v. Skype, Case No. 11-507 (RGA) (District of Delaware)
Subject Matter: P2P communications
Reports: 1 affidavit December 2012

APPENDIX B

APPENDIX B

TECHNICAL ANALYSIS OF MAILLARD'S CERTIFICATE HIERARCHY

By way of technical explanation, Maillard certificates are organized into a hierarchy.

1. CA – The Certification Authority. See, Maillard at [0080]. The CA maintains a private key CA_Kpri that it never discloses and a CA_Kpub that is installed into appropriate devices, such as the security module. See, Maillard at [0086]. It is also installed in device 60. See, Maillard at [0084]. The public key can be used to decrypt any data encrypted by CA_Kpri.

2. CE manufacturer – The manufacturer of device 60. See, Maillard at [0076]-[0081]. The CE manufacturer maintains a CEman_kpri that it never discloses and a CEman_kpub that is included in an encrypted certificate that is encrypted by the Certification Authority after certification is complete. See, Maillard at [0079]-[0081]. This encrypted certificate is known as CertCA(CEman_kpub). See, Maillard at [0081]. The "CA" subscript indicates that the data is encrypted by the private key of the CA. It can be decrypted by the CA's public key CEman_Kpub.

3. (CE) device 60 – The CE manufacturer creates a unique public key for the device designated Device_Kpub. The manufacturer further creates a certificate with this device public key encrypted by the manufacturer's CEman_kpri

CertCEman(Device_Kpub). This encrypted certificate is installed into the device. See, Maillard at [0084].

2. This "hierarchy" allows a security module to validate the encrypted certificate of the device. In particular:

1. The security module receives both the manufacturer's encrypted certificate (CertCA(CEman_kpub)) and the device's encrypted certificate (CertCEman(Device_Kpub)). See, Maillard at Fig. 5.

2. The security module has the CA's public key CA_Kpub installed as described above.

3. The security module decrypts the manufacturer's encrypted certificate with CA_Kpub to get CEman_Kpub. It knows this is an authorized manufacturer public key because only the CA has CA_Kpri and is the only party that could have encrypted the data.

4. The security module decrypts the device certificate with CEman_Kpub. It knows this is an authorized device public key because only the manufacturer has the CEman_Kpri and is the only party that could have encrypted the data.

APPENDIX C

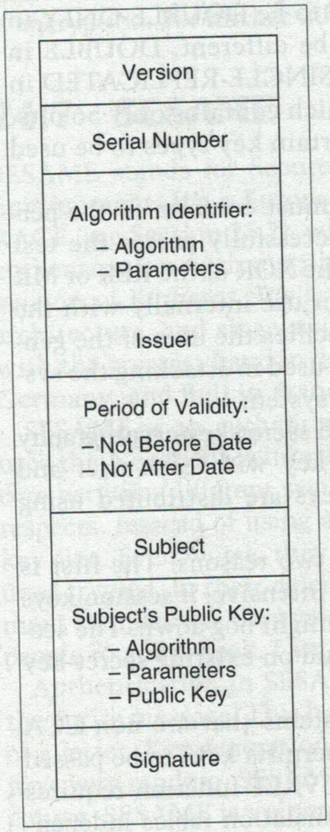
dently. Such dual control provides a high degree of assurance concerning the integrity and pedigree of any keys introduced into the system.

A key of type DATA is provided for compatibility with other systems. A DATA key is stored with a CV that identifies the key as a DATA key. DATA keys can have broad uses and as such must be regarded with suspicion and used with care. DATA keys may not be used for any key management functions.

The Commercial Data Masking Facility (CDMF) provides an exportable version of CCA. It has a special feature that reduces DES keys to an effective 40 bits for export (see Section 15.5) [785].

24.9 ISO Authentication Framework

Public-key cryptography has been recommended for use with the ISO authentication framework, also known as the X.509 protocols [304]. This framework provides for authentication across networks. Although no particular algorithms are specified for either security or authentication, the specification recommends RSA. There are



Version

Serial Number

Algorithm Identifier:
– Algorithm
– Parameters

Issuer

Period of Validity:
– Not Before Date
– Not After Date

Subject

Subject's Public Key:
– Algorithm
– Parameters
– Public Key

Signature

Figure 24.2 An X.509 certificate.

nce concerning

r systems.
DATA keys
ed with care.

xportable
ve 40 bits

he ISO
ramework
ithms are
nds RSA.

provisions, however, for multiple algorithms and hash functions. X.509 was initially issued in 1988. After public review and comment, it was revised in 1993 to correct some security problems [1100,750].

Certificates

The most important part of X.509 is its structure for public-key certificates. Each user has a distinct name. A trusted Certification Authority (CA) assigns a unique name to each user and issues a signed certificate containing the name and the user's public key. Figure 24.2 shows an X.509 certificate [304].

The version field identifies the certificate format. The serial number is unique within the CA. The next field identifies the algorithm used to sign the certificate, together with any necessary parameters. Issuer is the name of the CA. The period of validity is a pair of dates; the certificate is valid during the time period between the two. Subject is the name of the user. The subject's public key information includes the algorithm name, any necessary parameters, and the public key. The last field is the CA's signature.

If Alice wants to communicate with Bob, she first gets his certificate from a database. Then she verifies its authenticity. If both share the same CA, this is easy. Alice simply verifies the CA's signature on Bob's certificate.

If they use different CAs, it's more complicated. Think of a tree structure, with different CAs certifying other CAs and users. On the top is one master CA. Each CA has a certificate signed by the CA above it, and by the CAs below it. Alice uses these certificates to verify Bob's certificate.

Figure 24.3 illustrates this. Alice's certificate is certified by CA_A; Bob's is certified by CA_B. Alice knows CA_A's public key. CA_C has a certificate signed by CA_A, so Alice

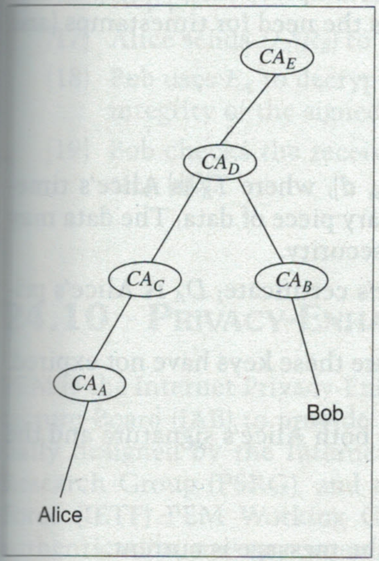


Figure 24.3 Sample certification hierarchy.

can verify that. CA_D has a certificate signed by CA_C. CA_B has a certificate signed by CA_D. And Bob's certificate is signed by CA_B. By moving up the certification tree to a common point, in this case CA_D, and then down to Bob, Alice can verify Bob's certificate.

Certificates can be stored on databases around the network. Users can send them to each other. When a certificate expires, it should be removed from any public directories. The issuing CA, however, should maintain a copy of the certificate. Should a dispute arise later, it will be required.

Certificates can also be revoked, either because the user's key has been compromised, the CA's key has been compromised, or because the CA no longer wants to certify the user. Each CA must maintain a list of all revoked but not expired certificates. When Alice receives a new certificate, she should check to see if it has been revoked. She can check a database of revoked keys on the network, but more likely she will check a locally cached list of revoked certificates. There are certainly possible abuses to this system; key revocation is probably its weakest part.

Authentication Protocols

Alice wants to communicate with Bob. First she goes to a database and obtains what is called a **certification path** from Alice to Bob, and Bob's public key. At this point Alice can initiate either a one-way, two-way, or three-way authentication protocol.

The one-way protocol is a single communication from Alice to Bob. It establishes the identities of both Alice and Bob and the integrity of any information communicated by Alice to Bob. It also prevents any replay attacks on the communication.

The two-way protocol adds a reply from Bob. It establishes that Bob, and not an imposter, sent the reply. It also establishes the secrecy of both communications and prevents replay attacks.

Both the one-way and two-way protocols use timestamps. A three-way protocol adds another message from Alice to Bob and obviates the need for timestamps (and therefore authenticated time).

The one-way protocol is:

(1) Alice generates a random number, R_A.

(2) Alice constructs a message, $M = (T_A, R_A, I_B, d)$, where T_A is Alice's timestamp, I_B is Bob's identity, and d is an arbitrary piece of data. The data may be encrypted with Bob's public key, E_B, for security.

(3) Alice sends $(C_A, D_A(M))$ to Bob. (C_A is Alice's certificate; D_A is Alice's private key.)

(4) Bob verifies C_A and obtains E_A. He makes sure these keys have not expired. (E_A is Alice's public key.)

(5) Bob uses E_A to decrypt $D_A(M)$. This verifies both Alice's signature and the integrity of the signed information.

(6) Bob checks the I_B in M for accuracy.

(7) Bob checks the T_A in M and confirms that the message is current.

a certificate signed by
e certification ...
e can verify Bob's ce-

Users can send them
oved from any public
py of the certificate

key has been compro-
A no longer wants to
ut not expired certifi-
k to see if it has been
work, but more likely
here are certainly pos-
kest part.

database and obtains
's public key. At this
e-way authentication

e to Bob. It establishes
nformation communi-
he communication.
s that Bob, and not an
communications and

A three-way protocol
d for timestamps (and

ere T_A is Alice's time-
of data. The data may

cate; D_A is Alice's pri-

keys have not expired.

ice's signature and the

age is current.

(8) As an option, Bob can check R_A in M against a database of old random numbers to ensure the message is not an old one being replayed.

The two-way protocol consists of the one-way protocol and then a similar one-way protocol from Bob to Alice. After executing steps (1) through (8) of the one-way protocol, the two-way protocol continues with:

(9) Bob generates another random number, R_B.

(10) Bob constructs a message $M' = (T_B, R_B, I_A, R_A, d)$, where T_B is Bob's timestamp, I_A is the identity of Alice and d is arbitrary data. The data may be encrypted with Alice's public key, E_A, for security. R_A is the random number Alice generated in step (1).

(11) Bob sends $D_B(M')$ to Alice.

(12) Alice uses E_B to decrypt $D_B(M')$. This verifies both Bob's signature and the integrity of the signed information.

(13) Alice checks the I_A in M' for accuracy.

(14) Alice checks the T_B in M' and confirms that the message is current.

(15) As an option, Alice can check the R_B in M' to ensure the message is not an old one being replayed.

The three-way protocol accomplishes the same thing as the two-way protocol, but without timestamps. Steps (1) through (15) are identical to the two-way protocol, with $T_A = T_B = 0$.

(16) Alice checks the received version of R_A against the R_A she sent to Bob in step (3).

(17) Alice sends $D_A(R_B)$ to Bob.

(18) Bob uses E_A to decrypt $D_A(R_B)$. This verifies both Alice's signature and the integrity of the signed information.

(19) Bob checks the received version of R_B against the R_B he sent to Alice in step (10).

24.10 Privacy-Enhanced Mail (PEM)

PEM is the Internet Privacy-Enhanced Mail standard, adopted by the Internet Architecture Board (IAB) to provide secure electronic mail over the Internet. It was initially designed by the Internet Research Task Force (IRTF) Privacy and Security Research Group (PSRG), and then handed over to the Internet Engineering Task Force (IETF) PEM Working Group. The PEM protocols provide for encryption, authentication, message integrity, and key management.

A well designed P&A technique is necessary to protect assets

Techniques for Privacy and Authentication in Personal Communication Systems

DAN BROWN

Personal Communication Systems (PCS) are anticipated to bring ubiquitous wireless telephony into widespread public use. To help further this goal, system designers are pursuing solutions to a number of challenges. These include accommodations for terminal (handset) mobility, personal mobility, universal roaming, access control, and the protection of user information sent via the airwaves. Of these, the latter two are often considered as a single subsystem called "Privacy and Authentication" (P&A). A well-designed P&A technique is necessary to protect assets. Network assets are protected by the access control (authentication) portion which enables legitimate users to utilize network services for which they have subscribed, while denying service to "hackers" who would steal services and monopolize resources. Subscriber assets (e.g., confidential information) are protected by encryption of traffic (privacy) on the radio link.

Authentication and privacy are generally linked together because the derivation of a "session key" for an encryption algorithm is often an integral part of the authentication process. From a designer's perspective, the access control and derivation of a session key form a single activity called Authentication and Key Agreement (AKA). The subsequent use of this session key to achieve encryption of user traffic can then be treated as a separate topic. This includes the selection of a cryptographic algorithm having properties that are compatible with the air interface to be protected.

This work describes progress to date in the development of AKA processes for PCS. A conceptual framework is first established; this is a three-part general model that characterizes all AKA techniques. Then three proposed AKA methods are compared using this model. These methods are the so-called secret key method of GSM, the secret key method of United States Digital Cellular (IS-54, IS-95), and a public key/secret key method that has been recently described in technical literature. Finally, a summary is presented that indicates the AKA method of preference for some proposed PCS air interfaces that are currently under development by standards bodies.

A General Model for Access Control (Authentication and Key Agreement)

Figure 1 depicts the general AKA process. The user's handset is shown on the upper-left side as a "flip-phone," and the service provider's network is shown as a cloud-like shape in the upper-right corner. Most security methods are initiated when the user purchases a phone, and continue toward the goal of protecting the user's traffic through encipherment over the wireless media. These endpoints are depicted at the top and bottom, respectively, in Fig. 1. The three-part security model that connects these endpoints provides the logical steps necessary to accomplish this process. These parts are described below. Three composite P&A methods are later compared through the use of this model.

The first part of the general security model is Provisioning. This is the means by which a handset or user acquires the bona fides that will enable the network to subsequently recognize him as a legitimate user. It is essential that these bona fides permit the user access to the network while frustrating any "hacker" who attempts access "replays" or false interrogation of the handset.

Part two of the model is the means by which a handset establishes credibility when the user registers with a local service provider who is generally not the "home" network. In such cases, a local service provider should only acquire a byproduct or subset of handset credentials. This is necessary in a well-planned P&A method because any promulgation of handset secrets will eventually result in their compromise. However, it is still necessary that the local network be capable of distinguishing a legitimate user, based on partial security information.

The third part of the model is the protocol that is executed to permit network access and establish a key for protection of channel traffic. In secret key systems, this is generally a simple challenge/response mechanism. Public-key systems typically use the exchange of "certificates" and modulo-exponentiation to complete the AKA transaction.

Comparison Process

These three processes will each be examined for PCS P&A proposals that are based upon GSM, IS-41, and a Public Key method. The purpose of this comparison is to highlight some differences of the three methods without attempting to assign quantitative values. The reader is encouraged to consult the references for more detailed information.

Because the GSM and IS-41 proposals are quite similar, the comparison process will be completed for these methods on a side-by-side basis. This is illustrated in Figs. 2, 3, and 4. Then the Public Key method will be described in order to contrast it to both secret key methods. The public-key method is shown in condensed form as Fig. 5.

1070-9916/95/$04.00 © 1995 IEEE

Provisioning in Secret Key Systems

Refer to Fig. 2. The upper portion shows that in GSM-style systems, the service provider controls the security process by issuing a "Subscriber Identity Module" (SIM) to the user. A SIM often takes the form of a credit card-like device intended for insertion in the handset. The SIM contains information about the services that have been purchased, and it also contains a 128-bit number called "Ki" that is unique for each SIM. Ki enables the SIM to authenticate itself to the network. When the service provider issues the SIM to the user, he also must store Ki in a secure manner at the network. A loss of Ki's at the network could result in widespread fraudulent access due to user impersonations. In GSM-style systems, Ki's never leave the network of the "home" service provider.

In the United States, IS-41-based digital wireless telephony evolved from the AMPS analog cellular system. IS-41 refers to the network signaling protocol; its companion digital air interface standards in the cellular spectrum are IS-54 (TDMA) and IS-95 (CDMA). In all current IS-41-based systems, subscriber-specific information is downloaded into a user's handset by electronic means, generally by a service shop that is authorized to perform this function by the service provider.

The practice of using a removable SIM has not been a component of the analog-to-digital evolution in the United States The introduction of security features into IS-54, IS-95, and later into AMPS and NAMPS has instead relied upon a method by which the user enters a security parameter called the "A-Key" into his handset via the keypad. This technique begins when the service provider sends the 64-bit A-Key to the user in a confidential manner, such as through the U.S. mail. This direct link between the user and the service provider is intended to bypass the service shop, which can be a source of fraud through either intentional or careless mishandling of security information. The user's correct entry of the A-Key is verified by security software within the handset. It is also necessary that the service provider store the user's A-Key at the "home" network. Provisioning of the A-Key is shown in the lower section of Fig. 2. The A-Key never leaves the "home" network, just as Ki never leaves its GSM "home" network.

Establishment of an A-Key is the first of two components of IS-41 security provisioning. A second security variable called the "Shared Secret Data" is derived from the A-Key by means of an over-the-air protocol that can be initiated by only the home service provider. SSD is intended to be shared between a home network and a visited network in order that the handset can be autonomously authenticated by the visited network. This feature is further discussed in the next section.

Access Control

The previous section describes how the "home" service providers in both GSM and IS-41 systems establish unique secrets with each subscriber. Mutual knowledge of these secrets enables the respective network to authenticate its

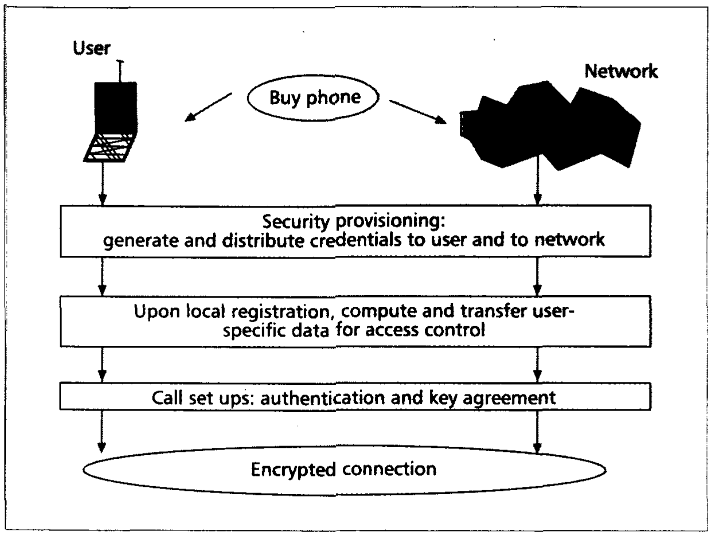

User
Buy phone
Network
Security provisioning:
generate and distribute credentials to user and to network
Upon local registration, compute and transfer user-specific data for access control
Call set ups: authentication and key agreement
Encrypted connection

Figure 1. *AKA process: a general model.*

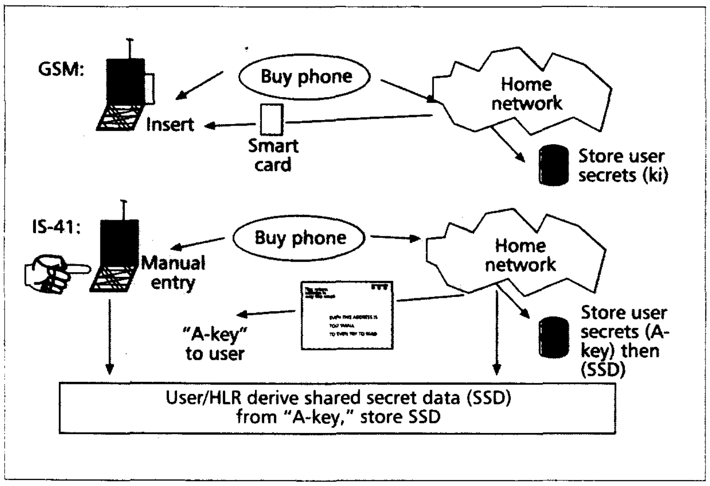


Figure 2. *P&A provisioning of secret key systems.*

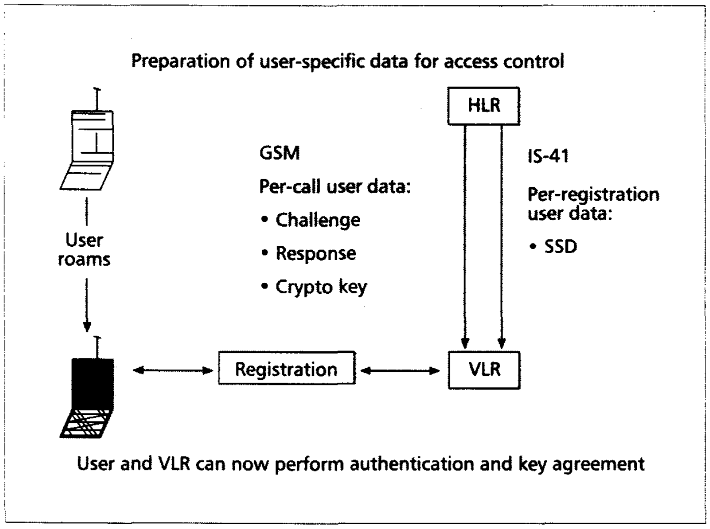

Preparation of user-specific data for access control
HLR
GSM
Per-call user data:
• Challenge
• Response
• Crypto key
IS-41
Per-registration user data:
• SSD
User roams
Registration
VLR
User and VLR can now perform authentication and key agreement

Figure 3. *Roaming support: secret key systems.*

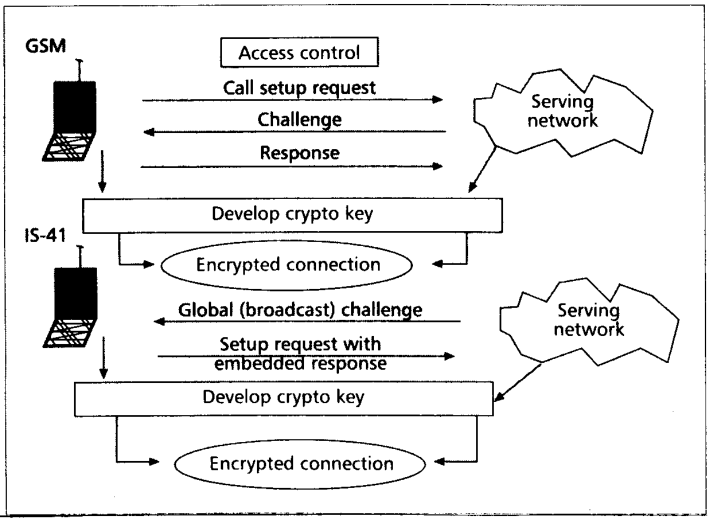


Figure 4. *AKA protocols: secret key systems.*

users when service is requested. However, terminal mobility permits the handset to be carried into the service areas of other, "visited" networks. This is also called "roaming," and business agreements are generally negotiated between service providers to support each other's roaming users. This results in a dilemma for the authentication process. A sufficient amount of information must be supplied to the visited network to authenticate a roamer, but this information must not be adequate to enable someone at the visited site to permanently impersonate a legitimate subscriber. The local network must be capable of performing authentication, but the process must be controlled by the home network.

The process of access control in a roaming situation is depicted in Fig. 3. A handset shown on the left side of the figure has roamed from its home network, served by its Home Location Register (HLR), to another network, where it will be served by the (other network's) Visited Location Register (VLR). An authentication will be performed upon handset registration with the VLR. Two flows of information are shown from the HLR to the VLR in order to support the authentication. The left-hand side depicts the information content that is supplied in GSM-style systems, while the right-hand side shows the information that is provided in IS-41-style systems.

In GSM-style systems, roaming subscribers are supported by their HLR in the form of "triplets" that provide security information to the VLR without revealing the secret Ki. Each set of triplets consists of a subscriber-unique random challenge RAND, an expected response SRES, and a resulting cipher key Kc. The number of triplets sent in a package may vary, but sets of five are common. Triplets are sent upon registration and thereafter as needed for the duration of the user's visit to the VLR's service area. The use of triplets permits the VLR to authenticate the roamer and establish a cipher key, but unauthorized interception of triplets does not enable permanent impersonation of a legitimate subscriber.

IS-41-style systems support roaming subscribers by transporting SSD from the HLR to the VLR. Knowledge of SSD enables the VLR to perform autonomous authentication of the user because the challenges and responses can be derived locally. This eliminates the need for additional HLR/VLR communications to provide further security information when needed. In addition, the visited system may utilize a "global" random challenge that can be broadcast on a system-wide information channel. The use of a global challenge enables the handset to respond to the challenge as a component of the service access request, thereby making efficient usage of bandlimited PCS channels.

An unauthorized interception of SSD upon transport to the VLR could result in a long-term impersonation of a user. IS-41-style systems employ a "Call Count" for protection from both this event and from general handset duplication, or "cloning." The call count is incremented in the handset upon a command from the network, generally during a call. The network also maintains the count. Later, during a subsequent access attempt, the handset sends its call count back to the network. If multiple handsets are sharing an identity, the network will accumulate a count that will likely exceed that of the legitimate user. Once a clone is suspected, network personnel can intervene. The preferred method of eliminating a clone is to request that the home network initiate the protocol to change the handset's SSD, as described above.

Authentication and Key Agreement Protocol in Secret Key Systems

Figure 4 illustrates simplified call flow models for AKA in GSM-based systems (upper portion) and in IS-41-based systems (lower portion). In either case, the goals are to assure the serving network that the handset is entitled to service and to develop a set of cipher bits for protection of user traffic over the RF link.

GSM-based AKA utilizes a challenge/response protocol to perform authentication and to generate cipher bits. This protocol is executed at the discretion of the serving network; a typical occurrence would be during a call setup. The network begins the procedure by either generating or selecting a challenge/response pair, called RAND/SRES, respectively. If the handset is being served by its "home" network, the 128-bit RAND is generated locally and then combined with the user's Ki to form the 32-bit SRES. RAND is then sent to the handset, where the handset will combine RAND with Ki to produce SRES. SRES will be returned to the network for comparison with the SRES that was calculated internally. If the two match, the handset will be considered to be authentic. Additional processing on RAND and Ki will produce the cipher bits called "Kc." This will occur at both the handset and the network; Kc can then be applied at both ends to protect traffic. In GSM, the computational algorithms that are used to combine RAND and Ki are selected by the home service provider and need not be common throughout the system.

In a roaming situation, the above scenario is unchanged, except that the RAND/SRES/Kc triplets are precomputed by the home network.

Triplets are transferred to the visited network to support a roaming user as a result of handset registration and/or a request for additional security information. The VLR performs the SRES comparison and provides Kc to the encipherment function, based on inputs from the home network.

IS-41-based systems also utilize a challenge/response method to perform authentication and derive cipher bits. As discussed in the previous section, the user's security variable is SSD; this is intended to be passed from a home network to a visited network upon registrations. Hence, the challenge/response mechanism is identical for both home users as well as roaming users.

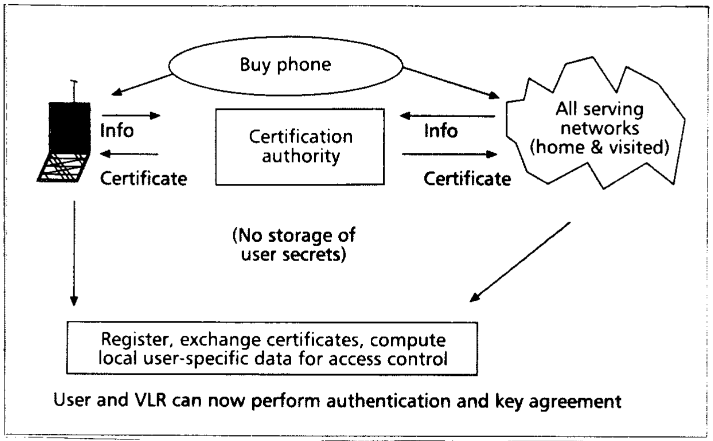
In an IS-41-style network, a single 32-bit "global" challenge is generated at frequent intervals and broadcast throughout the service area on a system information channel. Handsets that attempt a system access will compute an 18-bit authentication response by means of an authentication algorithm operating on their individual SSDs and the current global challenge. The access request package concatenates the registration/call setup information with the user's authentication response and call count value. For a registration, the response (and challenge) are sent to the home network for verification. If the handset is found to be authentic, SSD will be transported to the serving network along with other pertinent user data. During a call setup, receipt of the user's identity triggers a local data base lookup at the serving network to retrieve his SSD and call count. The authentication response is then verified when the serving network confirms that the retrieved SSD/global challenge combination can be applied to the authentication algorithm to produce the same response as that received from the handset. In addition, the call count is checked for accuracy. Further processing of the SSD/global challenge at both the handset and the local network then produces cipher bits for the protection of user traffic.

User Confidentiality in Secret Key Systems

Both secret key methods provide a means for user authentication and subsequent protection of user traffic. However, the registration and/or call setup process must include an identification of the user in order that the network may retrieve unique security information assigned to the subscriber. A subscriber ID that is available over the airwaves may be a security risk, especially in low-mobility settings, because it reveals knowledge of a subscriber's location.

GSM systems have dealt with this problem by the practice of using "temporary mobile station identities" (tmsi). This scheme requires that a subscriber reveal his true identity upon initial access. During the first call, the network then assigns to him, under encryption, an identity that is only known to him and to the serving network. "tmsi" may be reassigned by the serving network at its discretion. Anonymous roaming is accommodated when the user sends tmsi and the ID of the (previous) serving network to the current network. The use of a clear subscriber ID is permitted during network failures.

IS-41-based systems, such as the "PACS" air interface, are adopting similar schemes for the protection of user identities.


Buy phone
Info
Certificate
Certification authority
Info
Certificate
All serving networks (home & visited)
(No storage of user secrets)
Register, exchange certificates, compute local user-specific data for access control
User and VLR can now perform authentication and key agreement

■ **Figure 5.** *Provisioning and roaming support: Public Key/Secret Key hybrid.*

Introduction of Public Key AKA into PCS

PCS networks that emerge at 1.8 GHz are expected to adopt P&A techniques that are derived from some combination of existing GSM and/or IS-41 standards. However, additional security benefits may be realized by the introduction of public key techniques. The "PACS" air interface has already been designed with the capability of supporting a migration to a public key AKA method.

Public Key first appeared in mathematics journals in the mid-'70s, but has not been widely adopted for use in wireless systems. This is because most implementations required both excessive computations and the time-consuming transfer of large bit fields across noisy, bandlimited channels. However, three factors have enhanced the desirability of Public Key for PCS applications. First, some PCS air interfaces offer increased bandwidths over conventional cellular and two-way radio systems. Also, a low-mobility environment decreases the effects of channel fading. These considerations enable a quicker, more error-free transfer of large public key bit fields across a PCS channel. Second, the computational ability of low-cost processors continues to increase, which makes public-key mathematics less formidable. Third, recent studies [2] have proposed techniques that split the computational load unevenly between the PCS infrastructure and the handset. This enables the handset to perform relatively simple calculations, while the land-based infrastructure performs the intensive calculations.

The proposed public key method is summarized in Fig. 5, where the steps of provisioning and roaming support are combined for clarity. Once the access control information has been established, the AKA protocol for call setups can be performed using a secret key method, as in GSM or IS-41-based networks. This technique is referred to as the "Public Key/Secret Key Hybrid."

Handset provisioning begins when the user purchases a phone and requests service, as in the

> *As of the last quarter of 1994, seven air interfaces were under development by PCS standards bodies: PACS, DCS, IS-136-based, IS-95-based, the composite CDMA/TDMA/FDMA system, wideband CDMA, and the DECT-based proposal.*

secret key case. The user will then approach a "Certification Authority" (CA) with his credentials and some identity information about his handset and/or SIM-like detachable User Identity Module. The CA will verify the accuracy of the information and "sign" a coded version of this information. The digital signature uses the private portion of the CA's public key pair; this signature is returned to the user as a "certificate." Any PCS network may check the validity of the certificate by applying the public portion of the CA's public key pair.

In a similar manner, the CA issues certificates to all PCS network Access Controllers after verifying essential information. This permits the subsequent validity check of a network by the handset, by applying the CA's public key as described above.

This simplified model of the public key method assumes that a single CA serves all PCS handsets and all PCS networks. It is possible to utilize multiple CAs in peer-to-peer or hierarchical arrangements, but this adds complexity. The goal is to use a minimum of CAs that are trusted by many service providers.

After the CA has issued a certificate to provision the handset, local security credentials are established with the serving network at the time of registration. These credentials are generated at the handset and sent to the serving network under public key encryption. This eliminates the need to send "triplets" or "SSD" via a network-to-network transfer.

A further application of public key in the hybrid protocol occurs when the handset performs a per-registration digital signature on access-specific information. This is done to prevent access through the use of stolen certificates.

An attractive feature of this hybrid method over secret key techniques is that all "private" keys are never distributed beyond their source. The CA signs certificates with its private key but distributes its public key for certificate validation. The Access Controller generates a private key for usage during a portion of the protocol, but broadcasts its public key for handsets to perform encryption. The handset generates a private key for use in its digital signature calculations, but sends its public key to the network for digital signature validation. The practice of not distributing a private key means that network "hackers" will not be able to infiltrate a data base of handset secret numbers at the network.

The hybrid method offers some protocol advantages as well. Since the Access Controller's public key is broadcast, a registration can be anonymous. This eliminates the need for network management of user confidentiality, as through a "tmsi" method. Also, there should never be a requirement that a clear ID be sent due to administrative difficulties.

A further benefit of the hybrid method is that the serving network establishes security credentials for the handset upon registration, instead of though an information transfer from the home network. The registration process involves a mutual validity check by both the handset and the network, based upon credentials that have been prevalidated by the Certification Authority. This reassignment of the source of trust enables a savings in network resources, since neither "triplets" nor "SSD" are required to be sent to a visited network to support a roaming user. However, it will still be necessary that the user's service profile and credit status be maintained at his home network and be available to visited networks in order to provide continuous service.

Security Mechanisms of Proposed PCS Air Interfaces

As of the last quarter of 1994, seven air interfaces were under development by PCS standards bodies: PACS, PCS-1900, IS-136-based, IS-95-based, the composite CDMA/TDMA/FDMA system, wideband CDMA, and the DECT-based proposal. PACS security uses an IS-41-like AKA technique with the ability to migrate to the hybrid public key method. PCS-1900 uses a GSM-like security process. The composite system supports both GSM and IS-41 methods. Wideband CDMA, IS-136-based, and IS-95-based air interfaces rely on the IS-41-style P&A method. Various options remain under consideration for other air interfaces.

References

[1] M. J. Beller, L. Chang, and Y. Yacobi, "Privacy and Authentication on a Portable Communications System," *Proc. IEEE Global Telecommun. Conf.*, Dec. 2-5, 1991, pp. 1922-1927.

[2] M. J. Beller and Y. Yacobi, "Fully-Fledged Two-Way Public Key Authentication and Key Agreement for Low-Cost Terminals," *Electronic Letters*, 27th May 1993, vol. 29, no. 11, pp. 999-1001.

[3] European Telecommunications Standards Institute (ETSI), "European Digital Cellular Telecommunications System (phase 1); Recommendation GSM 03.20, Security Related Network Functions, version 3.3.3, Jan. 1991.

[4] Electronic Industries Association, EIA Interim Standard IS-54, Rev B, "Dual-Mode Mobile Station-Base Station Compatibility Standard," 1992.

Biography

DAN BROWN is a principal staff engineer in the Corporate Systems Research Labs at Motorola, Inc. Since joining Motorola in 1970, he has been involved in the development of commercial secure equipment. Recent activities include studies of authentication and voice privacy techniques for cellular phones and personal communications systems. He holds B.S.E.E. and M.S.E.E. degrees from the University of Illinois at Chicago.

Creating Web Representations for Places

Deborah Caswell and Philippe Debaty

Internet & Mobile Systems Lab
Hewlett-Packard Laboratories
Palo Alto, CA
(caswell@hpl.hp.com, debaty@hpl.hp.com)

Abstract. We believe that the future consists of nomadic people depending upon mobile appliances using World Wide Web protocols to communicate with services offered in real world places. Use of web protocols provides a ubiquitous communication infrastructure and allows interaction with the multitude of existing web-based services. Part of the challenge to realize our vision is to bridge the physical and virtual worlds by creating web representations for people, places, and things that interact virtually as they interact physically. We believe that an interesting set of new services can be provided by bridging the virtual and physical worlds in this way. This paper describes our experience with building a general place manager infrastructure useful for creating web representations for physical places. Although we leverage a general web presence architecture for building all different types of web presence, this paper focuses on the specific needs for building web representations for places.

1 Introduction

We are seeing the convergence of several trends: increasing availability of highly functional portable devices, deployment of wireless networking options, and the explosion in the number of services offered over the World Wide Web. The proliferation in use of mobile devices has increased the number of people who are always connected (or want to be) to the world wide web. Wherever they are, they have access to both a physical reality, and the virtual one presented through their mobile browser (also called its *web representation* throughout this paper). We believe that many useful services can be offered by creating a tighter link between the physical and virtual worlds. The HP Labs Cooltown project (http://www.cooltown.hpl.hp.com) explores this future based on a vision where people, places and things all have web-based representations [9].

In [2] we describe a general infrastructure for building web-present people, places, and things. The infrastructure provides the following capabilities that are necessary for creating the web representations we desire:

- Generates a custom web page for a user considering the user's circumstances.

P. Thomas and H.-W. Gellersen (Eds.): HUC 2000, LNCS 1927, pp. 114-126, 2000.

- Provides an authoring environment for easily creating templates that contain the look and feel of the web representation as well as instruction for dynamic customization.
- Provides a way to store and retrieve information concerning the relationships among web present entities. We call the component that does this the *directory*.
- Provides a configuration interface for defining security policy.

In this paper, we delve into more detail concerning our experience with building web representations for places using this general infrastructure. Whereas [2] concentrates on the similarities among the necessary infrastructures for people, places, and things, this paper focuses on the specific functionality needed in a place manager.

2 Creating a Place Web Representation

A *place* has physical boundaries and semantics assigned to its use. Thus, a virtual chat room is not a place because there are no physical boundaries. A church recreation room might provide the physical boundaries for multiple places, each one with a different use (scout meeting, 12 step program meeting, wedding reception).

A *place manager* is an infrastructure component that provides a web representation for a place. Our goal is to create a general infrastructure from which many different places and kinds of places can be built. We assume that there is a provider of the place with an interest in controlling the functionality offered and how the place appears to its users. It is the combination of provider perspective with the set of user circumstances that are considered for custom web page generation that makes this class of application different from other related work.

A bookstore, museum, conference room, home, and bus stop are some examples of places that become more useful and convenient with a web representation. The majority of web pages today that represent real or physical entities simply describe the place. For example, many retail stores have a web page that describes the merchandise they offer, directions to the store, and store hours. Others might also provide easy email access for asking questions, and still others might offer on-line ordering. Interaction is limited. There is no particular advantage to being physically present in the place and on-line at the same time. We believe that there is great value in providing a dynamic, interactive, and custom web representation for a physical place. It is the bridging of the virtual and physical worlds that makes this vision compelling.

A place's web representation is automatically offered to a browsing device as that device enters the place. The place's web page is generated as a function of the services offered in the place, as well as the devices and people physically in the place at the time of access. The integration of the physical (real) world with the virtual representation of that world is very useful, but not often done today. Imagine a bookstore trying to compete with one of the large online bookstores. In today's world, a connected mobile user could browse the physical book store, read parts of a book to determine whether or not she wanted it, then compare prices online with a vendor who can keep prices lower because of the lower overhead. Now imagine the same physical bookstore integrates the customer's experience with an online presence. If a book is out of stock, it can be ordered and shipped immediately without waiting in line for a store clerk to help. It can give suggestions of related books depending on the section of the bookstore where the user is physically. Some of the search criteria that would otherwise have to be entered manually can be inferred based on the section of the store in which the user is browsing. Another example is a coffee shop whose value to customers, in addition to great coffee, is the ability to telecommute from the shop. Access to office devices such as printers and scanners in the shop can be given to those buying coffee.

There is a good reason why the integration of physical locality and virtual reality has not become pervasive yet, despite the promise it holds for shop owners and other proprietors of physical places. Without a generic infrastructure for building such interactive and dynamic web representations, application building is very time consuming and requires an inter-disciplinary approach of sophisticated programming and web design. This section shows how the generic infrastructure described earlier can be used to create a *Place Manager* that can be customized for a variety of different kinds of places.

The literature is populated with reported efforts to build *context managers.* C*ontext* can refer to any kind of information including location, and the proposals for architecture and prototype applications have different requirements depending on specifically what context is being used. The place manager is a context manager that considers very specific kinds of context for building a specific class of applications: In addition to the user's location, we want the web place representation to be able to reflect the current time, the identity of the user accessing the place, and the capability of the device being used to access the web representation of the place. As a result of these requirements, a place manager is a combination of resource inventory manager and specialized web server that provides the web representation of a place taking all these specific notions of context into consideration.

2.1 Place-Specific Requirements

A place manager has certain requirements that stress a general implementation infrastructure in ways that are different from a person or thing manager.[1]

- *Relationships with other entities*: The web representation of a place needs to represent the people and things present within the place. There is a containment relationship, and for each person and thing within the place, there is a corresponding entry in the place's directory. People and mobile devices enter and leave places, and the current state of the directory must reflect the current state of the place at any given point in time (relative to some small time period of inconsistency). Thus, the implementation of the directory must be able to handle frequent updates.There are several different

[1] A person manager is an infrastructure component that provides a web representation for a person. A thing manager is an infrastructure component that provides a web representation for a thing.

service discovery protocols already in existence and several under development. The place manager should be able to support multiple discovery protocols for the purpose of registering entities that offer a service or are physically present within the place.

In addition to having relationships with people and things, places also have relationships with services and other places. Typically services that are particularly relevant to the use of the place are offered as links from the place's web representation. These services can be registered and unregistered dynamically. Therefore, there must be a way to store service information as well as people and thing entities. Inter-place relationships is a topic for further research.

- *Open-ended entity description for programmatic search*: A service might want to dynamically discover other resources it needs to operate. For example, a personal communication service might want to find all the devices in a place with phone, fax, and/or paging capability and then choose one to use as the endpoint for setting up a point-to-point communication. This requires that the directory must support programmatic search where the query is not known ahead of time. We cannot anticipate a priori what information a service might want about the entities within a place. Although standards organizations such as UPnP will be standardizing on entity descriptions that will dictate what attributes can be searched on in the arbitrary queries mentioned above, we believe that new devices and capabilities will be put into use faster than a standards body can update the standard to accommodate new features. Thus, it is important to be able to handle entity descriptions that include both standard and non-standard attributes. Although the place manager will not understand non-standard attributes, it must be able to search for them. Only the services searching on such attributes need know what they are.

- *Security policy*: It is valuable for a place to be able to authenticate a person/device's presence in the place. For purposes of user convenience, revenue protection, and place resource protection, there are times when services should only be offered to those who are physically present in a place, and withhold those services from those viewing the place's web representation remotely. Because there are semantics associated with the use of a place, some services, people, things, and places will be relevant to its web representation, and others will not be. Using the church recreation room as an example, the Karaoke machine that is relevant to a wedding reception place would not be relevant to the AA meeting. It becomes more challenging to specify and control what kinds of relationships are allowed in a place and with which other entities. The place provider needs a way of specifying a policy to determine which other entities may register a relationship in the place's directory, and the implementation must enforce that policy.

2.2 Discovery and Registration

In this section we discuss how relationships are created between a place and people, things, services, and other places. More specifically, we address how descriptions of these entities are entered into the place's directory. There are two parts to the process of establishing a relationship. First is noticing that a relationship exists and second is recording it. *Discovery* is the term we use for the process of noticing that a relationship exists. For a place, discovery usually involves noticing that a person or thing is located in the place and that a service or other place is relevant to the semantics of the place. *Registration* is the term we use for entering the entity description into the directory as a means of recording the relationship. Most industry standard and well known discovery protocols such as SLP [11] and SSDP [12] are standalone services. That is, they contain both discovery and registration mechanisms. Each provides its own silo of information concerning the entities discovered through its protocol, but does not lend itself well to sharing information across discovery mechanisms. Since we decouple discovery and registration, multiple discovery protocols can coexist, and the discovered entity descriptions are recorded in a common directory that is searchable independent of how the entity was discovered.

The place manager provides a registration interface for adding and deleting entity descriptions for entities with a relationship to the place. Registration is accomplished by invoking an *HTTP Get* operation and supplying as an argument the URL of the XML description of the entity to be registered. The place manager uses the URL to contact the entity's description that will be cached in the directory. This same interface is used regardless of how the relationship was discovered.

Now let's examine the various methods that can be used to discover entities in a place.

Manual Registration. There is an html form that allows an administrator to type in the URL of the XML description of an entity that the administrator wants to be in the place's directory. This is especially useful for services or other places that are chosen because of the relevance to the semantics of the place. We do not yet have an automated way to discover such relationships.

Tag Scanning and Recognition. For those things that have a web representation already and which are labeled with a passive tag, an administrator can create an association between the tagged thing and the place by scanning its tag, resolving the tag identifier to its URL, and having associated software automatically invoke the registration interface with its tag. Relationships are determined when an administrator decides which things to scan.

Device and User Self-Registration. The discovery mechanism we assume will be most commonly used is a scheme where a device offering a service registers itself with the place or a user's personal device registers the user in the place on the user's behalf. A place is assumed to have a short-range wireless beacon that announces the URL of the place manager's home page. Alternatively, there could be a passive id tag located in the place that when scanned would yield an id that is resolved to the URL of the place's home page. A device entering the place picks up the place's URL automatically, and uses it to construct the http-based registration invocation. To register, the device sends the URL of the XML description of itself or of the person it is registering.

This mechanism is preferable over manual and tag scanning methods because it requires no administrative intervention at registration time. This protocol also has several advantages over other industry accepted or emerging discovery protocols:

Point-to-point registration. The device contacts the place manager directly. The URL for bootstrapping the registration process is either provided by a short-range broadcast medium that is chosen to match the boundaries of the physical place or by scanning a passive tag whose id can only be read within the place. In either case, the search for a place manager is guaranteed to be confined to the boundaries of the place.

Single point of contact for finding place resources. A service that needs to identify other services, people, and things in a place does not need to broadcast a "is anyone out there? message" and then collate responses as is required by some discovery protocols. Instead, entities can be identified by contacting the place's directory in which information about the entities in the place is cached. That directory can be queried directly.

Caching an XML description avoids the need for fixed schemas. Arbitrary resources can be registered provided their description has the minimal set of information needed by the place manager for storage and retrieval. The XML attributes in the description need to be agreed upon between the entity and the services that will search for that resource based on the attributes. Not every attribute of an entity needs to be standardized across the industry before it can be used.

Other industry standard or emerging discovery systems [1][11][12] make different assumptions about the nature of the environment in which discovery will take place. Their use of client-side multicasting can lead to a mismatch between the network reachability of the multicast medium and the physical boundaries of place that should define the scope of the search. As a result of this mismatch in boundaries, a resource could end up being registered in multiple adjoining places and/or discovering other resources across adjoining places. Avoiding client-side multicasting is also important because mobile devices are likely to be low power. Enabling them to simply listen for a beacon then interact in a simple, point-to-point registration takes less power than multicasting and communicating with individual resources to learn their capabilities.

In order for a device to participate directly in the discovery protocol we define, it must have a beacon receiver or tag scanner, the ability to initiate the http-based registration invocation (e.g. a web browser, programmatic http client capability, or connection to a wap gateway), and the software to construct the registration invocation. That might be a tall order for small, limited-capability handheld devices. Providing they have some way of being discovered by a proxy, the proxy can register such a device on their behalf. Furthermore, the URL of the device's or user's description can be served up by any web server. There is no requirement that the device itself must run the web server that will provide the description.

Discovery Adapters. Some devices will enter a place with client software for some other discovery protocol. These devices can be discovered in a place by providing discovery adapters. A discovery adapter provides the server side of a discovery protocol to establish the relationship with a device, then uses the http-based registration invocation to record the relationship in the place's directory. In this way, industry standard and emerging discovery protocols such as SLP, SSDP, and Jini [7]discovery can coexist within a place, and client devices do not need to have additional discovery software installed that is specific to participating in the kinds of places we describe.

However, due to the limitations described above concerning the potential mismatch of networking technology and place boundaries, the burden is on the place administrator when configuring such adapters to ensure that the discovery mechanism is consistent with the policy of what may enter into a relationship with the place.

2.3 Security

Places have different security requirements. Some public places such as a bus stop will need to offer access without requiring a user to divulge their identity. These places must not rely on any kind of trust relationship. These places primarily offer information services where the user's identity does not affect the presentation or the information they receive. Other public places such as a library will have both information services provided to anonymous users and check-out services offered to authenticated members of the library.

Other places such as conference rooms and general use areas within a private company require more security. Most companies will have a private internal network with a firewall protecting places from access from people without access to the private net. There are many services for which access could be given to anyone on the private network regardless of who they are. However, if meetings are held in a conference room in which outside visitors attend, it might be desirable to provide place services to these guests without having to grant them access to the private network. To do that, the place manager must be cognizant of which network the user is on and issue URL's with the appropriate addresses that can be reached by that user. A proxy that sits on the corporate firewall would recognize the public URLs issued by the place manager and reissue the request to the corresponding private URLs in order to access place resources on the private network. The content available to visitors would be limited to the set of public URLs issued by the place manager and the translation table configured into the proxy.[10]

Then, a reverse proxy would serve the public URLs by translating certain ones to the internal address and let the request pass through the firewall.

Some services must be reserved for designated people who must authenticate themselves to be so designated. For example, a conference room place might establish the policy that only the current speaker is authorized to change the current slide on the overhead projector whereas all other people in the room may view the slide on their personal device. The need for services to be able to provide different authorization levels certainly is not new to places.

2.4 Location Authentication

There is much written about location-based web services: those services whose output is customized based on where the user is located. Examples include mapping programs and yellow pages. Because a place exists in a physical location, it can also offer services that are relevant to its location. For example, imagine that a conference has links to a map showing the floor plan of the building with the conference in the center or directions to the nearest bathroom, coffee station, copier, etc. A place administrator might want to provide such location-specific links only to users who are physically in the conference room and not those who are only virtually present. Those who participate in a meeting remotely would access the place's web representation to view the slides being presented, but would not need directions to the nearest copier. However, there is no adverse affect on the place if remote people were to have access to these links. In this section, we describe a set of scenarios that motivate the need to authenticate a person's physical presence in the place because of the adverse effects remote access could have on the place. There are at least three situations in which a place provider might want to generate a different place web representation depending on the presence or absence of the user in the place: convenience to the user, protecting revenues associated with the use of the place, and protecting place resources. Here we explain these motivations and give examples. Then we explain our basic approach to implementing location authentication.

2.5 Convenience

As explained before, the place manager generates a custom, on-demand web page for the user in his/her current circumstances that include the user's identity, browsing device capability, and time of day. Location is just another user circumstance. Not all web servers use a user's identity for restricting access or protecting resources Some web sites use the user's identity to make the content feel more personal. Similarly, location information can be used to customize the generated web pages. This is accomplished by hiding links to services that require physical presence.

One example is a library place's web page. It is reasonable and common for an employee to access the online card catalogue from their own office. However, it only makes sense to check out a book if one is physically present to take possession of the book. Many times books are expected to be on the shelf according to the inventory system, but cannot be found due to theft or mis-shelving. So, it is convenient to only include the check-out service for a user who is physically present in the library. It also provides some peace of mind to the library staff to encourage the consistency of their inventory system (do not want a book that has been checked out to be on the shelf). However, a user who can spoof their physical location (i.e. can assert that they are in the library when they are really at their desk) risks becoming responsible for a book not in their possession and the library has to deal with the potential inconsistency in their inventory system.

Another example might be selecting the play-list for a communal internet radio. (this is analogous to the old coin-operated jukebox in the local diner). Imagine a casual restaurant allowing their patrons to add selections to be played on the music system in the restaurant and wanting to restrict access to people physically in the restaurant. The reason for such a policy is to raise the probability that real patrons get to hear their own selection before they are finished eating. However, nothing dire happens if someone from outside of the restaurant can add selections to the play-list. It might be annoying, but then if that person were in the restaurant, their selections might be equally annoying.

2.6 Protect Revenues

For places that require an admission fee, providing special services in the place provides motivation for people to enter the place. For example, a rock concert might offer the service of viewing close-ups from different angles that the user chooses. If people not attending the concert were allowed access to this service, there would not be as much incentive to pay to attend the concert in person.

Another example is an amusement park that allows guests inside the park to monitor line lengths but prevents outsiders from doing so. This makes it more difficult for a competitor to gather statistics that could be used in negative advertising against the park.

For both of these examples, the consequences of spoofing presence are more important than the previous examples. However, the negative results of a security breach are not catastrophic. The rock concert will always draw a crowd of people who just enjoy the experience of seeing the band in person and would not find a web-based representation to be an adequate substitute. For the amusement park, word of mouth can already dissuade others from attending the park if the lines are too long. If the park is managed properly, there is nothing to fear from negative advertising.

Protect Place Resources. Many governments are considering making a move towards internet-based voting. Imagine a half-way solution where people must physically come to the polling place in order to be authenticated by a human verifying a picture id but the voting is then done on line, and the counts can be tallied in real time. Imagine that now a polling place can be much more comfortable and informal. One can vote from a mobile handheld device specially configured by the polling place staff that enables voting once and only once. Voting could be done while lounging on a sofa and sipping an iced tea. This step on the path to total internet voting has the advantage of getting results faster and more accurately without having to trust internet security to prevent fraudulent votes[2]. Furthermore, the comfortable atmosphere would make voting a more pleasant experience. In this case, we only want users who are present to access the voting system. The consequences of a breach could invalidate the entire election.

Another example is when supervision of place resources is needed. If a guest comes into Carol's office and she wants to share the contents of her file on the guest's PDA, she might not want the guest to be able to save or print the file. She wants to

[2] Voters might even prefer this method to complete on-line voting because the proving identity is divorced from the voting itself. Thus, there should be less worry that there will be a record of how an individual voted.

maintain ownership. She also wants access to the file to go away once the guest leaves her office, where she can no longer supervise their use. The consequence of a security breach could compromise her file system.

Notice that in both examples, identity authentication was very important. Critical place resources were issued only after the user established the trust of someone responsible for protecting the resources. As the owner of her office, Carol gets to discern which guests can access her file system, and she suffers the consequences if she exercises poor judgment.

Our Approach to Authenticating Location. The basic idea is to establish a user's presence by proving proximity to a known reference point within the place. There are several ways to do this. One way is to associate the place manager with a short-range networking access point and require all accesses desiring local status to be invoked over the short-range wireless network. Only users within the reachability of the short-range network could obtain local status, and its assumed that the reachability of the network does not extend outside of the boundaries of the place. However, this solution is too limiting because of the requirements each place would put on the networking infrastructure supported by a user's personal device and the potential mismatch of network reachability and place boundaries. To observe place boundaries, small places would be limited to using very short range networking technologies such as IR or Bluetooth whose bandwidth is quite limited. In a place where a higher speed wireless network coexists with the short range network confined to place boundaries, it is better to limit the use of short-range network to the discovery process and allow normal http traffic to take place over a higher speed medium.

It is desirable to separate the act of obtaining proof of location from the act of requesting the place's web representation. That requires that there be some token of information that can be presented at access time to prove proximity to the place's reference point. However, it is not sufficient for that token to be static because such information can be saved and replayed later at a time when the user is no longer in the place. Therefore, there must be information that can only be obtained by being in the place, and that information must change over time to prove presence within a certain time interval.

The approach we advocate does not require the act of proving location to be atomic with requesting access. A short-range wireless beacon emits a token that contains an encrypted timestamp along with the URL of the place manager. The token is placed in a cookie by the beacon receiver and is presented to the place manager on each access. The cookie is valid until the current time passes the time in the timestamp. When the cookie becomes invalid, the beacon receiver must get a new token and create a new cookie from it.

If location authentication is being used for convenience, we do not care that a user could retransmit the token over a wider-area network and those receiving the token could spoof presence in the place. However, if we are trying to protect revenues, the place manager would require that all communication be done over a secure link, and that individual tokens be generated for specific users. By encrypting the token in a secret shared between the user and the place manager, retransmitting the token will not enable outsiders to spoof location. Since server-only authentication can be used to secure the communication link and the beacon receiver can choose a random secret key for the personal device and place manager to share, the identity of the user can remain anonymous. It is important to note, however, that although the user's identity is not disclosed, the use of the secret key maps all accesses using that key to a single session. The key can be tied to other information about the access such as an internet address. The place manager can use this information to limit the extent of a security breach by only acknowledging one IP address associated with a secret key. If the IP address changes while still in the same place, a new secret key would have to be negotiated through authentication location again.

The remaining problem needing to be solved to protect place resources is that a person inside the place with malicious intent could retransmit both the token and the secret key to an outsider. If it is important to guard against an inside attack, then client authentication must be required as well, and the secret key is tied to the user's identity. In that way, although the place resources were violated by the outsider, the security breach would be traceable to the accomplice.

2.7 Scalability

Places come in various sizes and purposes. We have thought about all different kinds of places from personal office places to amphitheatre places where large concerts are held. The directory needs to scale from recording descriptions of just a few entities to recording several thousand entities. In addition, a general place manager infrastructure must be able to scale from serving a single isolated place (it might be embedded in a special purpose device) to having a centrally run and administrated place manager serving up the web representation of many places simultaneously. An example of that is having the IT department support a single place manager deployment serving all the conference rooms in the building.

We have built a stand-alone office place manager consisting of printing and personal communication services. We are in the process of building a single place manager with industrial database support to serve up an entire building's conference room web representation. We expect to deploy this version internally in Fall '2000.

3 Related Work

The team of Dey, Abowd, and Salber at Georgia Tech has been working to define architecture and toolkits for building context-sensitive applications where context is defined very broadly. In their paper *A Context-Based Infrastructure for Smart Environments*[0], the emphasis is on the architecture for context sensing. The important concept is the abstraction and aggregation of context information sensed in the environment. The architecture they describe could provide an infrastructure for implementing a place manager, as it meets the requirement of hiding the discovery details from the application. However, they do not describe the coexistence of multiple discovery schemes or manual and automatic discovery. Another paper from Georgia Tech [4] describes a context sensitive application that they built using their context toolkit. The Conference Assistant they describe requires application-specific code running on the user's mobile device.

In their paper Scalable and Flexible Location-Based Services for Ubiquitous Information Access [8], Jose and Davies from University of Minho and Lancaster University respectively present a generic architecture for building location-based applications. Their goal is providing information associated with a location context. The emphasis is on how to translate location into the semantic information of place. The location hierarchy they describe might be useful for providing a place manager that has relationships with other place managers (where the relationship is based on proximity or containment). However, there could be other kinds of relationships among places that would not be represented by a location hierarchy. Jose and Davies describe their tourist guide application as an example of the kind of service that can be built using their location-based service infrastructure. The tourist guide considers the user's location when displaying relevant information. However, all users are treated the same by the tourist guide. Thus, user identity is not used as context, and the infrastructure does not support describing and enforcing access control rules. In addition, it assumes a homogeneous browsing device. They do not deal with accommodating differences in the capabilities of the user's browsing device.

The Microsoft-sponsored Universal Plug and Play Forum has the notion of proximity networking. Proximity networking allows a mobile device to enter a new logical or geographic area and be able to interact with local resources without requiring pre-loaded applications specific to the area.[5][6] Unfortunately, there is not enough concrete information available yet on proximity networking to determine whether it will meet our requirements for supporting or providing place management.

4 Future Work

The work described in the paper is currently in progress. We have demonstrated the use of a place manager in an embedded, isolated application. Next, we will demonstrate the scalability of the place manager by using it to manage all the conference rooms in our building, several of which are large auditoria. In addition, we will implement a secure place that offers services to visitors not on the private company network and use location authentication for creating custom content on the basis of a user's presence or absence in the place.

Automatic discovery of relevant services and places is an area that requires future work as well.

5 Conclusions

We have built rudimentary place managers using a general implementation infrastructure. The specific requirements for creating a web representation for places have emphasized specific implementation issues. Our place managers interact with person managers and thing managers to create the powerful and useful web-based services envisioned in Cooltown.

6 Acknowledgements

We wish to recognize and thank the following people for their contributions to this paper. Our reviewers Tim Kindberg, John Schettino, Jeff Morgan, and Gary Herman gave us invaluable suggestions for improvement and support. Our technical consultants included Jean Tourilles, Bill Serra, Jeff Morgan, John Schettino, Marcos Frid, Venky Krishnan, Tim Kindberg, and John Barton.

References

1. Czerwinski, S.E. et al. *An Architecture for a Secure Service Discovery Service*, Mobicom '99, Seattle Washington.
2. Debaty, P. and D. Caswell. *Uniform Web Presence Architecture for People, Places, and Things.* Hewlett Packard Laboratories Technical Report HPL-2000-67, June 2000.
3. Dey, A.K., G.D. Abowd, and D. Salber. *A Context-Based Infrasructure for Smart Environments.* Proceedings of the 1st International Workshop on Managing Interactions in Smart Environments, Dublin, Ireland; Dec. 13-14, 1999.
4. Dey, A.K., M. Futakawa, D. Salber & G. D. Abowd. *The Conference Assistant: Combining Context-Awareness with Wearable Computing.* Proceedings of the 3rd International Symposium on Wearable Computers, San Francisco, CA, October 20-21, 199
5. http://www.microsoft.com/HOMENET/uPnP.htm
6. http://www.upnp.org/resources/UPnPbkgnd.htm
7. Jini Discoveryhttp://developer.java.sun.com/developer/Books/CoreJini/chapter6.html
8. Jose, R. and N. Davies. *Scalable and Flexible Location-Based Service for Ubiquitous Information Access*, Handheld and Ubiquitous Computing; Proceeding from the First International Symposium, HUC'99, Karlsruhe, Germany, September 1999.
9. Kindberg, T. et al. *People, Places, Things: Web presence for the Real World,* http://www.cooltown.hpl.hp.com HPLabs Technical Report HPL-2000-16.
10. Morgan, J. Internal technical memo entitled *Security in Cooltown.*
11. RFC 2165: Service Location Protocol, J. Veizades, E. Guttman, C. Perkins, S. Kaplan: ftp://ftp.isi.edu/in-notes/rfc2165.txt..
12. Simple Service Discovery Protocol, IETF Internet Draft, Y.Y. Golund, T. Cai, P. Leach, Y. Gu, S. Albright: http://www.upnp.org/download/spec-ssdp.doc.

Distance-Bounding Protocols

(Extended abstract)

Stefan Brands[1] and David Chaum[2]

[1] CWI, Amsterdam, email: brands@cwi.nl
[2] CWI & DigiCash, Amsterdam, email: david@digicash.nl

Abstract. It is often the case in applications of cryptographic protocols that one party would like to determine a practical upper-bound on the physical distance to the other party. For instance, when a person conducts a cryptographic identification protocol at an entrance to a building, the access control computer in the building would like to be ensured that the person giving the responses is no more than a few meters away.
The "distance bounding" technique we introduce solves this problem by timing the delay between sending out a challenge bit and receiving back the corresponding response bit. It can be integrated into common identification protocols. The technique can also be applied in the three-party setting of "wallets with observers" in such a way that the intermediary party can prevent the other two from exchanging information, or even developing common coinflips.

1 Introduction

A prover convincing a verifier of some assertion is a frequently recurring element in many applications of cryptography. One potentially useful such assertion is that the prover is within a certain distance. It seems that this problem has not been specifically adressed, let alone solved in the literature. We introduce a technique called "distance bounding" that enables the verifying party to determine a practical upper-bound on the physical distance to a proving party.

In the literature, so-called "mafia frauds" have been adressed in which a party identifies himself to a verifying party using the identity of a third party, without that third party being aware of it. With our distance-bounding technique we can prevent these frauds as a special case.

Our distance-bounding protocols can be integrated with known public-key identification schemes, such that the verifier cannot obtain information that he could not have computed himself.

In the recently proposed setting of "wallets with observers," distance bounding can be incorporated in such a way that the verifying party can determine a practical upper-bound to the observer, whereas the intermediary party can prevent the other two parties from exchanging or developing information which can be used to compromise privacy.

This paper is organized as follows: In Section 2 we introduce the distance-bounding principle. We introduce our solution in parts and then unify them.

T. Helleseth (Ed.): Advances in Cryptology - EUROCRYPT '93, LNCS 765, pp. 344-359, 1994.

In Section 3, we describe how distance bounding can be integrated into known public key identification schemes. In Section 4, we describe a problem in the setting of wallets with observers. We then show how to use the distance-bounding technique to solve it. A final section ends this paper with some open problems.

2 Distance-bounding protocols

In this section, we first present the basic distance-bounding principle. We then discuss mafia frauds and previously proposed countermeasures. We show how distance bounding can be used to prevent these frauds. We go on to show how distance bounding can prevent frauds in which a party having access to the secret keys convinces a verifying party that he is within a certain distance whereas he is not. Both protocols are then merged into one protocol that prevents both attacks.

2.1 The distance-bounding principle

The essential element of a distance-bounding protocol is quite simple. It consists of a single-bit challenge and rapid single-bit response. In practice, a series of these rapid bit exchanges is used, the number being indicated by a security parameter k. Each bit of the prover $\mathcal{P}$ is to be sent out immediately after receiving a bit from the verifier $\mathcal{V}$. The delay time for responses enables $\mathcal{V}$ to compute an upper-bound on the distance.

What makes this approach really practical is that today's electronics can easily handle timings of a few nanoseconds, and light can only travel about 30cm during one nanosecond. For instance, even the timing between two consecutive periods of a 50 Mghz clock allows light to travel only three meters and back. (Later on we introduce exclusive-or operations on the bits exchanged, but 10113 chips have several such gates each with a throughput of two nanoseconds.)

2.2 Mafia frauds

A *mafia fraud*, first described in [9], is a real-time fraud that can be applied in zero-knowledge or minimum disclosure identification schemes by fraudulent prover $\overline{\mathcal{P}}$ and verifier $\overline{\mathcal{V}}$, cooperating together. It enables $\overline{\mathcal{P}}$ to convince an honest verifier $\mathcal{V}$ of a statement related to the secret information of an honest prover $\mathcal{P}$, without actually needing to know anything about this secret information. To this end, when $\mathcal{P}$ is about to perform the protocol with $\overline{\mathcal{V}}$, the latter establishes, say, a radio link with $\overline{\mathcal{P}}$, and will send any information transmitted to him by $\mathcal{P}$ straight on to $\overline{\mathcal{P}}$, who in turn sends it on to $\mathcal{V}$. The same strategy is applied by $\overline{\mathcal{P}}$, who sends information received from $\mathcal{V}$ on to $\overline{\mathcal{V}}$, who in turn sends it on to $\mathcal{P}$. In effect, $\overline{\mathcal{V}}$ and $\overline{\mathcal{P}}$ act as a single transparent entity, sitting in the middle between $\mathcal{P}$ and $\mathcal{V}$. This enables $\overline{\mathcal{P}}$ to identify himself to $\mathcal{V}$ as being $\mathcal{P}$, without any of $\mathcal{P}$ and $\mathcal{V}$ noticing the fraud.

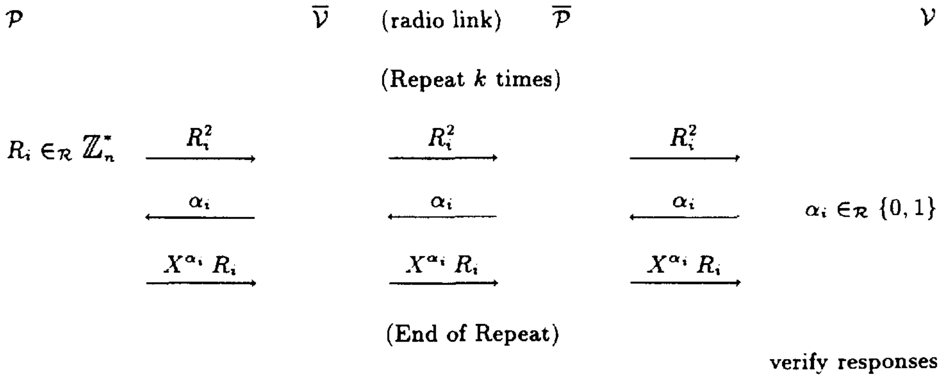


Fig. 1. A mafia fraud in the basic Fiat-Shamir identification scheme.

In Figure 1, a mafia fraud is shown as it would be applied in the basic Fiat-Shamir identification scheme (see [12]). In order to enhance readability of the figures, we define the subscript i to run over the set $\{1, \ldots, k\}$, and computations are modulo n. In the most basic form of the Fiat-Shamir scheme $\mathcal{P}$ identifies himself to $\mathcal{V}$ by proving knowledge of a square root X of $X^2 \bmod n$, where $X^2 \bmod n$ in some way is related to $\mathcal{P}$'s identity or has been published in a trusted directory. As usual, n is the product of two distinct primes.

In [9], Desmedt proposed a countermeasure to mafia frauds which requires $\mathcal{P}$ to sign a message that contains his physical location on earth, and then prove to $\mathcal{V}$ knowledge of the signature. Usually in an identification scheme, $\mathcal{P}$ will be represented by some user-module, so it will be impracticable to implement this solution without requiring position detection or cooperation of the user. Also, it cannot guarantee that the verifier in the long run does not learn anything about the secret key of the prover.

In [2], Beth and Desmedt propose that all transmission times be accurately measured. This seems to be useless owing to the significant possible variations in speed of computation.

In [1], Bengio *et al* suggested that $\mathcal{V}$ shield $\mathcal{P}$'s module from the outside world (e.g., in a Faraday cage) when the protocol takes place. This countermeasure requires trust by $\mathcal{P}$ that $\mathcal{V}$ does not secretly modify his module in some way while shielded. One would rather like to identify oneself in such a way that the module remains visible (an infrared channel would be even better, the user-module never needing to leave the hands of the user). Futhermore, it requires special hardware equipment.

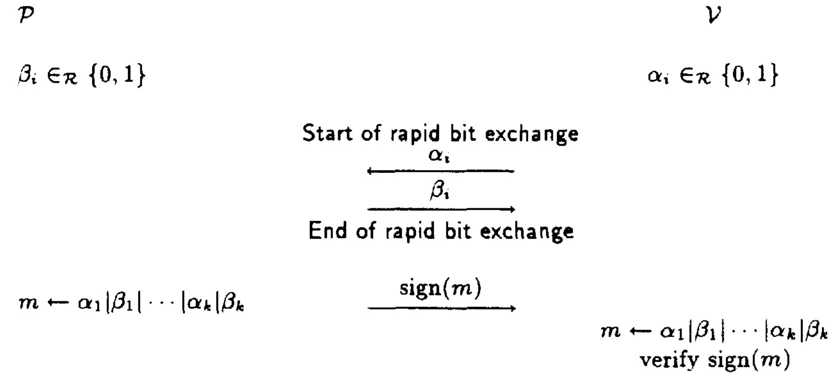

$\mathcal{P}$
$\mathcal{V}$
$\beta_i \in_{\mathcal{R}} \{0,1\}$
$\alpha_i \in_{\mathcal{R}} \{0,1\}$
Start of rapid bit exchange
α_i
β_i
End of rapid bit exchange
$m \leftarrow \alpha_1|\beta_1|\cdots|\alpha_k|\beta_k$
sign(m)
$m \leftarrow \alpha_1|\beta_1|\cdots|\alpha_k|\beta_k$
verify sign(m)

Fig. 2. Distance bounding to prevent mafia frauds.

2.3 Preventing mafia frauds using distance bounding

Consider how the distance-bounding principle can be used to prevent mafia frauds. We can assume that the distance between $\mathcal{P}$ and the fraudulent parties is not less than the accuracy that can be achieved with the apparatus being used, since otherwise obvious countermeasures can be taken. To ensure that the distance between $\mathcal{V}$ and the party $\mathcal{P}$ having access to the secret keys is measured, after the rapid bit exchanges have taken place the message formed by concatenating all the $2k$ bits sent back and forth in the distance-bounding stage is signed by $\mathcal{P}$, using his secret key (see Figure 2):

Step 1 $\mathcal{V}$ generates uniformly at random k bits α_i, and $\mathcal{P}$ generates uniformly at random k bits β_i. (Note: this can take place well beforehand.)

Step 2 Now the low-level distance-bounding exchanges can take place. The following two steps are repeated k times, for $i = 1, \ldots, k$.
- $\mathcal{V}$ sends bit α_i to $\mathcal{P}$.
- $\mathcal{P}$ sends bit β_i to $\mathcal{V}$ *immediately after* he receives α_i.

Step 3 $\mathcal{P}$ concatenates the $2k$ bits α_i and β_i, signs the resulting message m with his secret key, and sends the signature to $\mathcal{V}$. We denote concatenation by the symbol"|."

Now $\mathcal{V}$ can determine an upper-bound on the distance to $\mathcal{P}$ using the maximum of the delay times between sending out bit α_i and receiving bit β_i back, for $i = 1, \ldots, k$. $\mathcal{V}$ accepts if and only if $\mathcal{P}$ is close by, and the received signature is a correct signature of $\mathcal{P}$ on $m = \alpha_1|\beta_1|\cdots|\alpha_k|\beta_k$.

Proposition 1. *If the signature scheme is secure and $\mathcal{P}$ is not close by to $\mathcal{V}$, then a mafia fraud has probability of success at most $1/2^k$.*

That is. the probability of successful cheating decreases exponentially in the number of repetitions of the rapid bit exchange. The simple proof of this proposition is very similar to the proof of Proposition 3 in the next section.

2.4 Preventing the prover from sending bits out too soon

In this subsection we study a setting in which $\mathcal{P}$ has access to the secret keys, and $\mathcal{V}$ wants to be ensured that $\mathcal{P}$ is close by. A remarkable thing about the distance bounding stage in the protocol of the previous subsection is that the bits that $\mathcal{P}$ sends to $\mathcal{V}$ do not have to depend on the bits that $\mathcal{V}$ sends to $\mathcal{P}$. If $\mathcal{P}$ knows at what times $\mathcal{V}$ will send out bits, he can have $\mathcal{V}$ accept by sending β_i out to $\mathcal{V}$ at the correct time before he receives α_i, regardless of the distance to $\mathcal{V}$. Hence, the protocol we described for preventing mafia frauds does not prevent this fraud.

Two solutions suggest themselves. The first solution consists of $\mathcal{V}$ sending bits out with randomly chosen delay times. Since $\mathcal{P}$ cannot anticipate when $\mathcal{V}$ expects to have received back a bit. he cannot send out bits β_i before he has received bit α_i (since $\mathcal{V}$ will not accept if a response bit β_i arrives before he has sent out bit α_i). In fact, it is sufficient if $\mathcal{V}$ sends out bit α_i at random at one of two discrete times, say, at the rising edge of clock pulse $3i$ or $3i+1$, for $1 \leq i \leq k$. The probability of the strategy having success is at most $1/2^k$ if the choices are made independently.

The second solution consists of ensuring $\mathcal{V}$ that $\mathcal{P}$ must choose bits β_i depending on α_i. One way to do this involves creating a public bitstring $m_1|\cdots|m_k$ once (the choice of the bits m_i is irrelevant). The following protocol implements this (see Figure 3):

Step 1 $\mathcal{V}$ generates uniformly at random k bits α_i.

Step 2 Now the low-level distance-bounding exchanges can take place. The following steps are repeated k times, for $i = 1, \ldots, k$.

- $\mathcal{V}$ sends bit α_i to $\mathcal{P}$.
- $\mathcal{P}$ sends bit $\beta_i = \alpha_i \oplus m_i$ to $\mathcal{V}$ *immediately* after receiving bit α_i from $\mathcal{V}$.

$\mathcal{V}$ verifies whether the bit-string $(\alpha_1 \oplus \beta_1)|\cdots|(\alpha_k \oplus \beta_k)$ equals the public bitstring. If so, $\mathcal{V}$ computes an upper-bound on the distance to $\mathcal{P}$ using the maximum of the delay times between sending out bit α_i and receiving bit β_i back, for $i = 1, \ldots, k$. $\mathcal{V}$ accepts if and only if $\mathcal{P}$ is close by.

As before, it is easy to see that the probability that $\mathcal{V}$ accepts when $\mathcal{P}$ is not close by is at most $1/2^k$.

2.5 Preventing both types of fraud

By combining the two protocols, we can prevent both types of fraud. As before, it is assumed that a bit-string $m_1|\cdots|m_k$ is published. The following protocol can be used (see Figure 4):

Step 1 $\mathcal{V}$ generates uniformly at random k bits α_i.

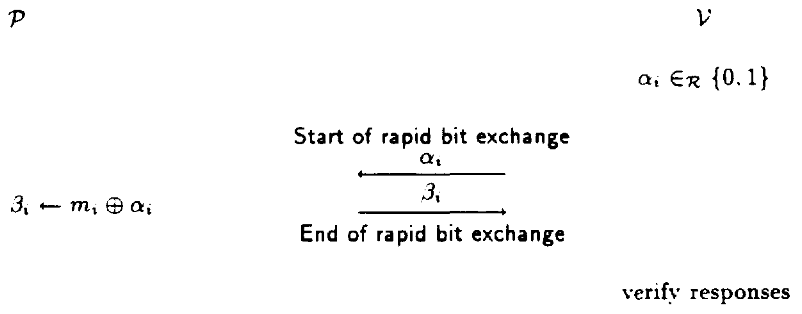

$\mathcal{P}$
$\mathcal{V}$
$\alpha_i \in_{\mathcal{R}} \{0,1\}$
Start of rapid bit exchange
α_i
$\beta_i \leftarrow m_i \oplus \alpha_i$
β_i
End of rapid bit exchange
verify responses

Fig. 3. Preventing the response bits from being sent out too soon.

Step 2 $\mathcal{P}$ generates uniformly at random k bits m_i. As before, both $\mathcal{P}$ and $\mathcal{V}$ can do so well beforehand. $\mathcal{P}$ commits to k bits m_i using a secure commitment scheme.

Step 3 Now the low-level distance-bounding exchanges can take place. The following steps are repeated k times, for $i = 1, \ldots, k$.

- $\mathcal{V}$ sends bit α_i to $\mathcal{P}$.
- $\mathcal{P}$ sends bit $\beta_i = \alpha_i \oplus m_i$ to $\mathcal{V}$ *immediately* after he receives α_i.

Step 4 $\mathcal{P}$ opens the commitment(s) on the bits β_i by sending the appropriate information to $\mathcal{V}$. $\mathcal{P}$ concatenates the $2k$ bits α_i and β_i, signs the resulting message m with his secret key and sends the resulting signature to $\mathcal{V}$.

With the information received in Step 4, $\mathcal{V}$ verifies whether the bits $\alpha_i \oplus \beta_i$ are indeed those commited to in Step 2. If this holds, then $\mathcal{V}$ computes m in the same way as $\mathcal{P}$ did and verifies whether the signature he received is indeed a correct signature of $\mathcal{P}$ on m. If so, he computes an upper-bound on the distance to $\mathcal{P}$ using the maximum of the delay times, and accepts if and only if $\mathcal{P}$ is close by.

3 Integration with public key identification schemes

The fact that a secure signature scheme must be used in the protocols of Subsection 2.3 and 2.5 can be a problem when the prover wishes only to identify himself by for example proving knowledge of a square root X of X^2 mod n (as in the basic Fiat-Shamir identification scheme): it is not clear how he should sign the message by using his secret information X; also, since $\mathcal{V}$ receives information that he could not have computed himself, it is not clear whether he obtains useful information for computing the secret keys. In this section, we show how to integrate distance bounding with known public key identification schemes such that no useful information is transferred.

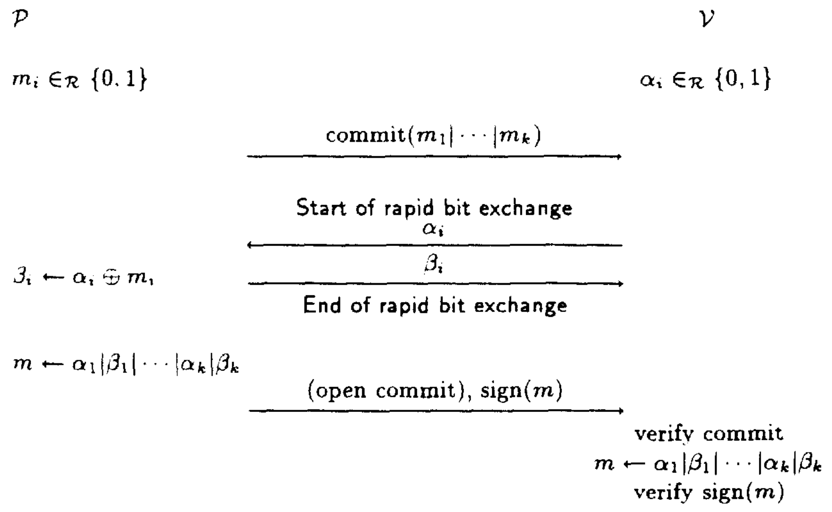

$\mathcal{P}$
$\mathcal{V}$
$m_i \in_{\mathcal{R}} \{0,1\}$
$\alpha_i \in_{\mathcal{R}} \{0,1\}$
commit$(m_1|\cdots|m_k)$
Start of rapid bit exchange
α_i
$\beta_i \leftarrow \alpha_i \oplus m_i$
β_i
End of rapid bit exchange
$m \leftarrow \alpha_1|\beta_1|\cdots|\alpha_k|\beta_k$
(open commit), sign(m)
verify commit
$m \leftarrow \alpha_1|\beta_1|\cdots|\alpha_k|\beta_k$
verify sign(m)

Fig. 4. Distance bounding to prevent both types of fraud.

3.1 Preventing mafia frauds

To prevent mafia frauds, we have the distance-bounding protocol dictate that $\mathcal{P}$ respond to challenges formed as the exclusive-or of the bits sent and received, instead of signing the concatenation. We illustrate this with the basic Fiat-Shamir scheme:

Step 1 $\mathcal{P}$ generates uniformly at random k numbers $R_i \in \mathbb{Z}_n^*$, and sends their squares R_i^2 mod n to $\mathcal{V}$. $\mathcal{P}$ also generates uniformly at random k bits β_i and commits to these bits (and their order) by sending a commitment on them to $\mathcal{V}$.

Step 2 $\mathcal{V}$ generates uniformly at random k bits α_i.

Step 3 Now the low-level distance-bounding exchanges can take place. Hereto, the following steps are repeated k times, for $i = 1, \ldots, k$.
- $\mathcal{V}$ sends bit α_i to $\mathcal{P}$.
- $\mathcal{P}$ sends bit β_i to $\mathcal{V}$ *immediately* after he receives α_i from $\mathcal{V}$.

Step 4 $\mathcal{P}$ opens the commitment on the bits β_i made in Step 1 by sending the appropriate information to $\mathcal{V}$. Furthermore, $\mathcal{P}$ determines the k responses $X^{c_i} R_i$ corresponding to challenges $c_i = \alpha_i \oplus \beta_i$, for $1 \leq i \leq k$, and sends them to $\mathcal{V}$.

$\mathcal{V}$ determines the k challenges c_i in the same way as $\mathcal{P}$ did, and verifies that the k responses are correct. Then $\mathcal{V}$ verifies whether the opening of the commitments by $\mathcal{P}$ is correct. If this holds, $\mathcal{V}$ computes an upper-bound on the distance to $\mathcal{P}$

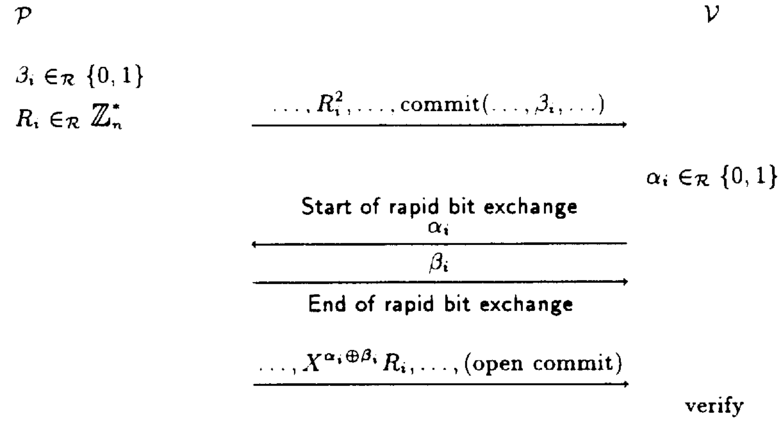

$\mathcal{P}$
$\mathcal{V}$
$\beta_i \in_{\mathcal{R}} \{0,1\}$
$R_i \in_{\mathcal{R}} \mathbb{Z}_n^*$
$\ldots, R_i^2, \ldots, \text{commit}(\ldots, \beta_i, \ldots)$
$\alpha_i \in_{\mathcal{R}} \{0,1\}$
Start of rapid bit exchange
α_i
β_i
End of rapid bit exchange
$\ldots, X^{\alpha_i \oplus \beta_i} R_i, \ldots, \text{(open commit)}$
verify

Fig. 5. Distance bounding in the Fiat-Shamir identification scheme.

using the maximum of the delay times between sending α_i and receiving β_i, for $i = 1, \ldots, k$. $\mathcal{V}$ accepts if and only if $\mathcal{P}$ is close by.

Proposition 2. *If the commitment scheme is secure, then this protocol is a proof of knowledge of a square root of* $X^2 \bmod n$ *that does not reveal any useful information for computing a root of* $X^2 \bmod n$.

Sketch of proof. In effect, this protocol is the parallel version of the basic Fiat-Shamir identification protocol. In [11] it is proven that this protocol reveals no useful information.

Since the binary challenges are chosen *mutually* random, the verifier cannot choose them as the outcome of a collision-free hash-function of the information known to him before Step 2. That is, the verifier does not receive information that he cannot compute himself. In particular, the transcript of an execution of the protocol cannot be used as a digital signature to convince others that the execution took place.

Proposition 3. *If the commitment scheme is secure,* $\mathcal{P}$ *is not close by to* $\mathcal{V}$ *and both follow the protocol, then the mafia fraud has probability of success at most* $1/2^k$.

Sketch of proof. In order to have any chance at all of having $\mathcal{V}$ accept, the fraudsters $\overline{\mathcal{P}}$ and $\overline{\mathcal{V}}$ must perform the rapid bit exchange first entirely with $\mathcal{V}$ and then with $\mathcal{P}$ (or vice versa), otherwise $\mathcal{V}$ will not accept because the computed upper-bound on the distance will not be tight enough (see Figure 6).

However, since a commitment was sent in Step 2, it is clear that with probability $1 - 1/2^k$ the fraudsters cannot prevent $\mathcal{P}$ and $\mathcal{V}$ from ending up with

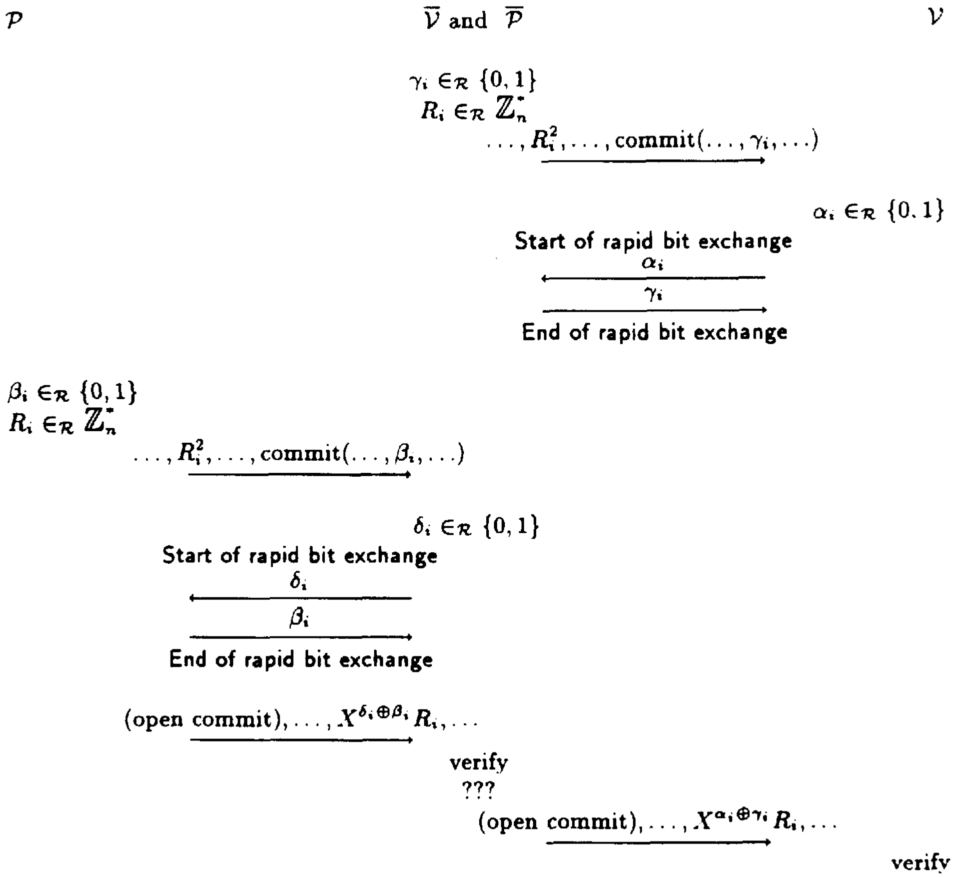

$\mathcal{P}$
$\overline{\mathcal{V}}$ and $\overline{\mathcal{P}}$
$\mathcal{V}$
$\gamma_i \in_{\mathcal{R}} \{0,1\}$
$R_i \in_{\mathcal{R}} \mathbb{Z}_n^*$
$\ldots, R_i^2, \ldots, \mathrm{commit}(\ldots, \gamma_i, \ldots)$
$\alpha_i \in_{\mathcal{R}} \{0,1\}$
Start of rapid bit exchange
α_i
γ_i
End of rapid bit exchange
$\beta_i \in_{\mathcal{R}} \{0,1\}$
$R_i \in_{\mathcal{R}} \mathbb{Z}_n^*$
$\ldots, R_i^2, \ldots, \mathrm{commit}(\ldots, \beta_i, \ldots)$
$\delta_i \in_{\mathcal{R}} \{0,1\}$
Start of rapid bit exchange
δ_i
β_i
End of rapid bit exchange
(open commit), $\ldots, X^{\delta_i \oplus \beta_i} R_i, \ldots$
verify
???
(open commit), $\ldots, X^{\alpha_i \oplus \gamma_i} R_i, \ldots$
verify

Fig. 6. Can $\overline{\mathcal{P}}$ and $\overline{\mathcal{V}}$ apply a mafia fraud?

at least one different challenge (i.e. $\beta_i \oplus \delta_i \neq \alpha_i \oplus \gamma_i$). Therefore, at least one response of $\mathcal{P}$ is correct with respect to a challenge that is complementary to the challenge $\mathcal{V}$ expects a response to. Clearly, one cannot convert $X^c R$ to $X^{c \oplus 1} R$ without knowing X.

Note that if at least one of $\mathcal{P}$ and $\mathcal{V}$ generates the challenge bits according to a distribution other than the uniform one (i.e., does not follow the protocol), this only increases the probability of successful cheating for $\overline{\mathcal{P}}$ and $\overline{\mathcal{V}}$.

Although we had the prover commit himself, it does not really matter whether the prover or the verifier commits. This holds for the protocol in the next section as well.

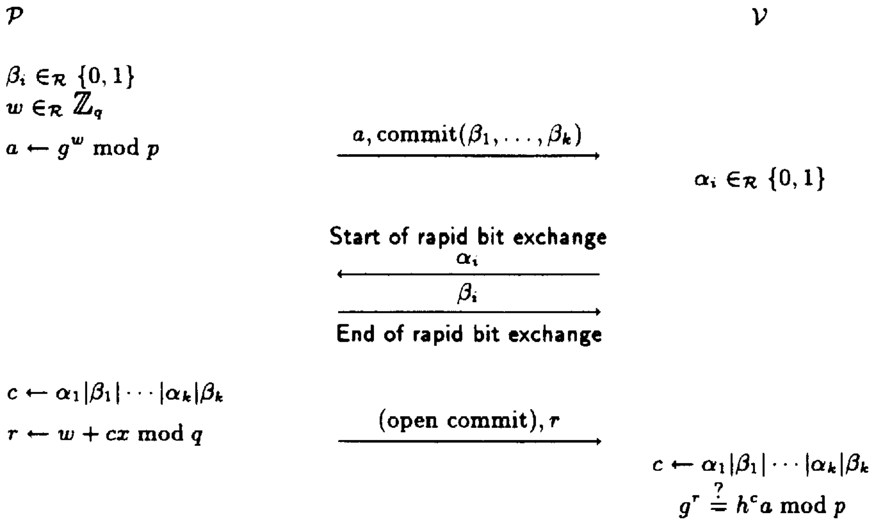

$\mathcal{P}$
$\mathcal{V}$
$\beta_i \in_{\mathcal{R}} \{0,1\}$
$w \in_{\mathcal{R}} \mathbb{Z}_q$
$a \leftarrow g^w \bmod p$
$a, \mathrm{commit}(\beta_1, \ldots, \beta_k)$
$\alpha_i \in_{\mathcal{R}} \{0,1\}$
Start of rapid bit exchange
α_i
β_i
End of rapid bit exchange
$c \leftarrow \alpha_1|\beta_1|\cdots|\alpha_k|\beta_k$
$r \leftarrow w + cx \bmod q$
(open commit), r
$c \leftarrow \alpha_1|\beta_1|\cdots|\alpha_k|\beta_k$
$g^r \stackrel{?}{=} h^c a \bmod p$

Fig. 7. Distance bounding in the Schnorr identification scheme.

3.2 Preventing both types of fraud

In order to prevent both types of frauds, as in Subsection 2.5, one can straightforwardly modify this protocol. To this end, in Step 1 $\mathcal{P}$ commits to k bits m_i, and in Step 3 $\mathcal{P}$ will reply with response bits $\beta_i = \alpha_i \oplus m_i$. Finally, the responses of $\mathcal{P}$ in Step 4 must be computed with respect to the multi-bit challenge $\alpha_1|\beta_1|\cdots|\alpha_k|\beta_k$ (using the concatenation of the xor-values does not prevent mafia frauds). This technique can be integrated in (minimum disclosure) identification schemes with multi-bit challenges, such as [5, 14, 15, 17], retaining the same security level for $\mathcal{P}$. Observe that the same propositions hold for this modified protocol; the only distinction is that the challenge bits *can* be chosen as the outcome of a collision-free hashfunction, and hence the transcript can serve as a digital signature.

Figure 7 shows how one might incorporate distance bounding into the Schnorr identification scheme. In this protocol, $(p, q, g, h = g^x \bmod p)$ is the public key of $\mathcal{P}$(as in [17]).

4 Distance bounding in wallets with observers

Up to now, we have considered distance-bounding protocols in a model with two legitimate parties. In this section, we will discuss distance bounding in a certain three-party setting. The goal of $\mathcal{V}$ is to determine an upper-bound to $\mathcal{P}$, and the task of the intermediary is to prevent undesired flow of information between

$\mathcal{P}$ and $\mathcal{V}$. Our technique allows the intermediary to prevent common coinflips between $\mathcal{V}$ and $\mathcal{P}$. This can be thought of as a generalization of the "warden's problem" (see [18]).

Recently, transaction systems based on "wallets with observers" have been proposed (see [6]). This setting can simultaneously offer privacy and security to an unprecedented extent. This is achieved by embedding within each user-module a tamper-resistant device called an observer. The observer is incorporated in a user-module in such a way that any message it sends to the outside world has to pass through the user-module. That is, the user-module acts as an intermediary party. The benefit of this setting is that one can design protocols such that the observer and the user-module both have to participate in order to have a verifier accept. In this way, a user cannot, say, double-spend the same coin in an electronic cash system since the observer will not participate a second time (see e.g. [3]).

Often, it will be sufficient to prevent outflow (any information going from the observer to the verifier not specified by the protocol) and inflow (any information going from the verifier to the observer not specified by the protocol). Inflow and in particular outflow can be a serious threat to the privacy of the user.

In [8] the privacy aspect of the wallet with observer setting has been investigated under an even more stringent requirement: even if an observer were to store all information it receives during the period it is embedded within a user-module, it still should be impossible (independent of computing resources) to link a payment to a user by examining afterwards the information inside the observer and all information gathered by the verifying parties. This possibility is not excluded by preventing inflow and inflow, since for example a single random number known to both an observer and a shop would enable linking: the fact that the user-module took part in generating it (so that no information could be encoded within it, thus preventing both inflow and outflow) is irrelevant in this matter. That is, one must also prevent "common coinflips." In [8], the term "shared information" is proposed, encompassing inflow, outflow, and common coinflips. The essential technique ("divertability") needed to prevent shared information in such a setting has been proposed earlier by Desmedt in [9], and was generalized in [16]. Prevention of shared information in some instances can be viewed as a slight generalization of divertability, in that the keys have to be shared together with the intermediary in a suitable way.

A fraud that can be applied in this three-party setting is one in which a user illegitimately uses an observer embedded within someone else's wallet. A possible motivation for doing so is that typically observers will gather (part of) negative credentials which can prevent the user from doing transactions he would like to do (see e.g. [7]). Also, another observer might have (part of) certain positive credentials the user would like to make use of. One can imagine a fraudulent organization specializing in lending, at a distance, observers with positive credentials (or without certain negative ones) to users who are willing to pay for this. In effect, when the user wants to do a transaction for which he needs certain positive credentials, he could use a radio link with the fraudulent organization and lets an appropriate observer authorize the transaction. We will call this the "observer fraud."

4.1 Preventing the observer fraud

Using our distance-bounding technique we show how the verifier in the three-party setting can determine an upper-bound on the distance to the observer, such that the user-module can prevent shared information. We only describe one protocol that meets the most stringent requirements: no shared information, no release of useful information for computing the secret key, and the verifier obtains no information that he could not have computed himself (transcripts cannot serve as proof that the protocol took place). For easy comparison with the distance-bounding protocol shown in Section 3, our discussion will be based on the three party version of the Fiat-Shamir protocol (see [9, 16]).

We need a new notion called a "xor-commitment scheme." This is a commitment scheme which enables one to commit to the exclusive-or $\alpha \oplus \beta$ of two bits α and β, whereas one only knows a commitment on β but not β itself. In addition, one should be able to open the xor-commit if and only if one knows how to open the commitment on β, and this opening information must leak no Shannon information on the bits α and β, and the random choices involved in the commitment on β.

An implementation of an xor-commitment scheme can be realized with RSA, based on the technique of probabilistic encryption (see [13]). Let n be a Blum integer. In order to encrypt a bit α, the commiter chooses $r \in \mathbb{Z}_n^*$ at random and computes $\text{commit}(\alpha) := (-1)^\alpha r^2 \bmod n$. According to the quadratic residuocity assumption (see [13]), it is infeasible to decide whether $\text{commit}(\alpha)$ is a quadratic residue or not (i.e., whether $\alpha = 1$ or 0), unless one knows the factorization of n. Given a commitment $\text{commit}(\alpha) = (-1)^\alpha r^2 \bmod n$ of a bit α and a commitment $\text{commit}(\beta) = (-1)^\beta s^2 \bmod n$ of a bit β, it follows that $\text{commit}(\alpha \oplus \beta) = (-1)^{\alpha\oplus\beta} r^2 s^2 \bmod n$ is an xor-commitment on $\alpha \oplus \beta$. When opening this commit, one reveals $rs \bmod n$, which does not contain any information on s.

We denote the observer by $\mathcal{O}$, the verifier by $\mathcal{V}$ and the user-module by $\mathcal{U}$. For clarity, we leave out the fact that to prevent shared information the secret and public keys must be shared between the observer and the user-module in a suitable way. It is not hard to see how to do this using some of the techniques suggested in [7].

In the protocol, $\mathcal{O}$ knows a square root X of $X^2 \bmod n$, and $\mathcal{O}$ wishes to convince $\mathcal{V}$ of this fact in such a way that $\mathcal{U}$ does not learn it, whereas $\mathcal{U}$ can be ensured that there is no shared information. $\mathcal{V}$ wants to be convinced not only of the fact that $\mathcal{O}$ knows a square root of $X^2 \bmod n$, but also that $\mathcal{O}$ is close by. In essence, this is the setting of the mafia fraud, with the intermediary party ($\overline{\mathcal{P}}$ and $\overline{\mathcal{V}}$ in mafia frauds, and $\mathcal{U}$ in this situation) also trying to prevent shared information.

The protocol is as follows (see Figure 8):.

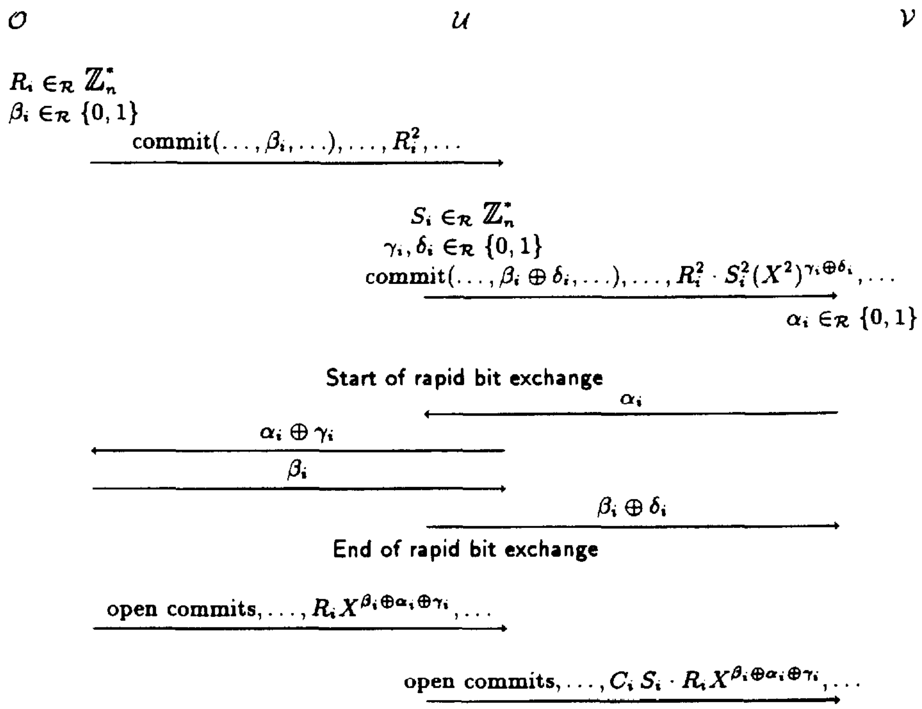

$\mathcal{O}$
$\mathcal{U}$
$\mathcal{V}$
$R_i \in_{\mathcal{R}} \mathbb{Z}_n^*$
$\beta_i \in_{\mathcal{R}} \{0,1\}$
commit$(\ldots,\beta_i,\ldots),\ldots,R_i^2,\ldots$
$S_i \in_{\mathcal{R}} \mathbb{Z}_n^*$
$\gamma_i,\delta_i \in_{\mathcal{R}} \{0,1\}$
commit$(\ldots,\beta_i \oplus \delta_i,\ldots),\ldots,R_i^2 \cdot S_i^2(X^2)^{\gamma_i \oplus \delta_i},\ldots$
$\alpha_i \in_{\mathcal{R}} \{0,1\}$
Start of rapid bit exchange
α_i
$\alpha_i \oplus \gamma_i$
β_i
$\beta_i \oplus \delta_i$
End of rapid bit exchange
open commits$,\ldots,R_i X^{\beta_i \oplus \alpha_i \oplus \gamma_i},\ldots$
open commits$,\ldots,C_i\, S_i \cdot R_i X^{\beta_i \oplus \alpha_i \oplus \gamma_i},\ldots$

Fig. 8. Diverted Fiat-Shamir identification protocol with distance bounding.

Step 1 $\mathcal{O}$ generates k random numbers $R_i \in_{\mathcal{R}} \mathbb{Z}_n^*$ and sends the squares $R_i^2 \bmod n$ of these numbers to $\mathcal{U}$. $\mathcal{O}$ also generates k bits β_i, and sends a xor-commitment on them to $\mathcal{U}$. (Clearly, if we use the specific xor-commitment just described, a commitment for each bit would be needed.)

Step 2 $\mathcal{U}$ first verifies that the numbers received from $\mathcal{O}$ all have Jacobi symbol 1. If this is the case, he generates at random k bits $\gamma_i \in_{\mathcal{R}} \{0,1\}$ as well as k bits $\delta_i \in_{\mathcal{R}} \{0,1\}$. $\mathcal{U}$ also generates k numbers $S_i \in_{\mathcal{R}} \mathbb{Z}_n^*$. He then computes the k products $R_i^2 \cdot S_i^2(X^2)^{\gamma_i \oplus \delta_i} \bmod n$ and sends them to $\mathcal{V}$. $\mathcal{U}$ also sends xor-commitment(s) on $\beta_i \oplus \delta_i$ to $\mathcal{V}$.

Step 3 $\mathcal{V}$ generates k challenge bits $\alpha_i \in_{\mathcal{R}} \{0,1\}$, which he will use for the rapid bit exchange.

Step 4 Now the rapid exchange of bits can take place. Hereto, the following four exchanges are repeated k times, for $i = 1,\ldots,k$:

- $\mathcal{V}$ sends bit α_i to $\mathcal{U}$.
- $\mathcal{U}$ sends $\alpha_i \oplus \gamma_i$ to $\mathcal{O}$ *immediately* after receiving α_i.
- $\mathcal{O}$ sends challenge bit β_i to $\mathcal{U}$ *immediately* after receiving $\alpha_i \oplus \gamma_i$.

– $\mathcal{U}$ sends $\beta_i \oplus \delta_i$ to $\mathcal{V}$ *immediately* after receiving β_i.

Step 5 $\mathcal{O}$ opens the k commits on the bits β_i to $\mathcal{U}$ $\mathcal{O}$ also computes the k responses $R_i X^{\beta_i \oplus \alpha_i \oplus \gamma_i} \bmod n$ and sends them to $\mathcal{U}$.

Step 6 $\mathcal{U}$ verifies whether the responses of $\mathcal{O}$ are correct with respect to the challenges $\alpha_i \oplus \gamma \oplus \beta_i$ and the squares received from $\mathcal{O}$ in Step 3. $\mathcal{V}$ verifies whether the bits that $\mathcal{O}$ sent to him in Step 4 are those he commited to. If all the verifications hold, then $\mathcal{U}$ computes the k responses $C_i\, S_i \cdot R_i X^{\alpha_i \oplus \gamma_i \oplus \beta_i} \bmod n$ by multiplying, for $1 \leq i \leq k$, the i-th response of $\mathcal{O}$ by $S_i \bmod n$ and a correction-factor C_i. The correction-factor is equal to $X^2 \bmod n$ if and only if $\gamma_i \oplus \delta_i = 1$ and $\alpha_i \oplus \beta_i \oplus \delta_i = 1$, otherwise it is equal to 1. $\mathcal{U}$ sends all these responses to $\mathcal{V}$. Furthermore, $\mathcal{U}$ opens the xor-commitments to the k values $\beta_i \oplus \delta_i$ to $\mathcal{V}$.

Afterwards, $\mathcal{V}$ verifies whether the responses of $\mathcal{U}$ are correct with respect to the challenges $\beta_i \oplus \delta_i \oplus \alpha_i$ and the squares received from $\mathcal{V}$ in Step 3. He also verifies whether the bits received from $\mathcal{U}$ in Step 4 are those $\mathcal{U}$ commited to. If all the verifications hold, then $\mathcal{V}$ derives an upper-bound on the physical distance to $\mathcal{O}$ by using the maximum of the delays between sending out α_i and receiving $\beta_i \oplus \delta_i$ from $\mathcal{U}$, for $1 \leq i \leq k$. $\mathcal{V}$ accepts if and only if $\mathcal{O}$ is close by.

Although we write commit$(\ldots, \beta_i, \ldots,)$ we do not mean to imply with this that a multi-bit commitment must necessarily be used: one might as well use k single-bit commitments.

It is straightforward to show that $\mathcal{V}$ accepts if all parties follow the protocol, and that Propositions 2 and 3 hold.

Since one can easily show that for each view of $\mathcal{V}$ and for each view of $\mathcal{O}$ in this protocol, there is exactly one set of random choices that could have been made by $\mathcal{U}$ such that the views are from the same execution of the protocol, there is no shared information. Clearly, for the protocol as we described it, this only holds for executions concerning proof of knowledge of the particular number $X^2 \bmod n$. However, as we noted before, if the knowledge of $X^2 \bmod n$ is divided between $\mathcal{O}$ and $\mathcal{U}$ in a suitable way (as described in [8]), the property of absence of shared information holds for the set of *all* proofs of knowledge, regardless of the particular number $X^2 \bmod n$ that the proof is concerned with.

Finally, as in Proposition 3 it is easy to see that the following must hold.

Proposition 4. *If $\mathcal{O}$ and $\mathcal{V}$ follow the protocol, then $\mathcal{U}$ cannot (with probability of success greater than $1/2^k$) trick $\mathcal{V}$ into believing that $\mathcal{O}$ is close by if this is not the case.*

As before, if at least one of $\mathcal{O}$ and $\mathcal{V}$ generates challenge bits according to a distribution other than the uniform one, then $\mathcal{U}$'s probability of successful cheating will only increase.

5 Open problems and further work

We would like to present two potentially fruitful areas for further investigation.

First is the physics and practical implementation of distance-bounding technology. We know little about the physical limits or precisely how best to use current technology. Some experimental work might also be interesting.

Second is dealing with a problem not adressed here. The techniques presented do not prevent frauds in which a distant party with access to the secret keys is *cooperating* with a party close by (without conveying the secret keys). The frauds were suggested informally under the name "the terrorist fraud" by Desmedt in [9]. We are currently working on some ideas preventing such frauds using distance bounding.

Acknowledgements

We are grateful to Niels Ferguson for some suggestions concerning distance bounding in wallets with observers, to Torben Pedersen for the particular realization we described of an xor-commitment, and to Corné Hoogendijk for discussions on the physical realizations of distance-bounding.

References

1. Bengio, S., Brassard, G., Desmedt, G., Goutier, C. and Quisquater, J., "Secure Implementation of identification schemes," Journal of Cryptology, 4 (1991), pages 175–183.
2. Beth, T. and Desmedt, Y., "Identification tokens–or: solving the chess grandmaster problem," Crypto '90, Lecture Notes in Computer Science, Springer-Verlag (1991), pages 169–176.
3. Brands, S., "An efficient off-line electronic cash system based on the representation problem," C.W.I. Technical Report CS - T9323, april 1993, The Netherlands.
4. Brassard, G., Chaum, D. and Crépeau, "Minimum Disclosure Proofs of Knowledge," Journal of Computer and System Sciences, Vol. 37 (1988), pages 156–189.
5. Brickell, E. and McCurley, K., "An interactive identification scheme based on discrete logarithms and factoring," Journal of Cryptology, Vol. 5, no. 1 (1992), pages 29–39.
6. Chaum, D., "Achieving electronic privacy," Scientific American, Aug. 1992, pages 96–101.
7. Chaum, D. and Pedersen, T., "Wallet Databases with Observers," Proceedings of Crypto '92, Abstracts, Santa Barbara, August 1992, pp. 3.1-3.6.
8. Cramer, R. and Pedersen, T., "Improved privacy in wallets with observers," in: these proceedings.
9. Desmedt, Y., "Major security problems with the 'unforgeable' (Feige)-Fiat-Shamir proofs of identity and how to overcome them," SecuriCom '88, SEDEP Paris, (1988), pages 15–17.
10. Desmedt, Y., Goutier, C., and Bengio, S., "Special uses and abuses of the Fiat-Shamir passport protocol," Crypto '87, LNCS 293, Springer-Verlag (1988), pages 16-20.
11. Feige, U., Fiat, A. and Shamir, A., "Zero-knowledge proofs of identity," Journal of Cryptology 1 (1988), pages 77–94.

12. Fiat, A. and Shamir, A., "How to prove yourself: practical solutions to identification and signature problems," Crypto '86, Springer-Verlag, (1987), pages 186–194.
13. Goldwasser, S. and Micali, S., "Probabilistic Encryption," Journal of Computer and System Sciences, Vol. 28 (1984), pages 270–299.
14. Guillou, L. and Quisquater, J.-J., "A 'paradoxical' identity-based signature scheme resulting from zero-knowledge," Crypto '88, Springer-Verlag, pages 216-231.
15. Okamoto, T., "Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes," Proceeding of Crypto '92, pages (1-15) - (1-25).
16. Okamoto, T. and Ohta, K., "Divertible zero knowledge interactive proofs and commutative random self-reducibility," Eurocrypt '89, Springer-Verlag, pages 134-149.
17. Schnorr, C.P., "Efficient Signature Generation by Smart Cards," Journal of Cryptology, Vol. 4, No. 3, (1991), pages 161–174.
18. Simmons, G., "The prisoner's problem and the subliminal channel," Crypto '83, Santa Barbera (1983), Plenum, New York, pages 51–67.

Kipp E.B. Hickman
Netscape Communications Corp

Internet Draft

April 1995 (Expires 10/95)

The SSL Protocol
<draft-hickman-netscape-ssl-00.txt>

1. STATUS OF THIS MEMO

This document is an Internet-Draft.Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

To learn the current status of any Internet-Draft, please check the "1id-abstracts.txt" listing contained in the Internet- Drafts Shadow Directories on ds.internic.net (US East Coast), nic.nordu.net (Europe), ftp.isi.edu (US West Coast), or munnari.oz.au (Pacific Rim).

2. ABSTRACT

This document specifies the Secure Sockets Layer (SSL) protocol, a security protocol that provides privacy over the Internet. The protocol allows client/server applications to communicate in a way that cannot be eavesdropped. Server's are always authenticated and clients are optionally authenticated.

3. INTRODUCTION

The SSL Protocol is designed to provide privacy between two communicating applications (a client and a server). Second, the protocol is designed to authenticate the server, and optionally the client. SSL requires a reliable transport protocol (e.g. TCP) for data transmission and reception.

The advantage of the SSL Protocol is that it is application protocol independent. A "higher level" application protocol (e.g. HTTP, FTP, TELNET, etc.) can layer on top of the SSL Protocol transparently. The SSL Protocol can negotiate an encryption algorithm and session key as well as authenticate a server before the application protocol transmits or receives its first byte of data. All of the application protocol data is transmitted encrypted, ensuring privacy.

The SSL protocol provides "channel security" which has three basic properties:

The channel is private. Encryption is used for all messages after a simple handshake is used to define a secret key. Symmetric cryptography is used for data encryption (e.g. DES, RC4, etc.)

The channel is authenticated. The server endpoint of the conversation is always authenticated, while the client endpoint is optionally authenticated. Asymmetric cryptography is used for authentication

(e.g. Public Key Cryptography).

The channel is reliable. The message transport includes a message integrity check (using a MAC). Secure hash functions (e.g. MD2, MD5) are used for MAC computations.

The SSL protocol is actually composed of two protocols. At the lowest level, layered on top of some reliable transport protocol, is the "SSL Record Protocol". The SSL Record Protocol is used for encapsulation of all transmitted and received data, including the SSL Handshake Protocol, which is used to establish security parameters.

4. SSL Record Protocol Specification

4.1 SSL Record Header Format

In SSL, all data sent is encapsulated in a record, an object which is composed of a header and some non-zero amount of data. Each record header contains a two or three byte length code. If the most significant bit is set in the first byte of the record length code then the record has no padding and the total header length will be 2 bytes, otherwise the record has padding and the total header length will be 3 bytes. The record header is transmitted before the data portion of the record.

Note that in the long header case (3 bytes total), the second most significant bit in the first byte has special meaning. When zero, the record being sent is a data record. When one, the record being sent is a security escape (there are currently no examples of security escapes; this is reserved for future versions of the protocol). In either case, the length code describes how much data is in the record.

The record length code does not include the number of bytes consumed by the record header (2 or 3). For the 2 byte header, the record length is computed by (using a "C"-like notation):

```
RECORD-LENGTH = ((byte[0] & 0x7f) << 8)) | byte[1];
```

Where byte[0] represents the first byte received and byte[1] the second byte received. When the 3 byte header is used, the record length is computed as follows (using a "C"-like notation):

```
RECORD-LENGTH = ((byte[0] & 0x3f) << 8)) | byte[1];
IS-ESCAPE = (byte[0] & 0x40) != 0;
PADDING = byte[2];
```

The record header defines a value called PADDING. The PADDING value specifies how many bytes of data were appended to the original record by the sender. The padding data is used to make the record length be a multiple of the block ciphers block size when a block cipher is used for encryption.

The sender of a "padded" record appends the padding data to the end of its normal data and then encrypts the total amount (which is now a multiple of the block cipher's block size). The actual value of the padding data is unimportant, but the encrypted form of it must be transmitted for the receiver to properly decrypt the record. Once the total amount being

transmitted is known the header can be properly constructed with the PADDING value set appropriately.

The receiver of a padded record decrypts the entire record data (sans record length and the optional padding) to get the clear data, then subtracts the PADDING value from the RECORD-LENGTH to determine the final RECORD-LENGTH. The clear form of the padding data must be discarded.

4.1.1 SSL Record Data Format

The data portion of an SSL record is composed of three components (transmitted and received in the order shown):

MAC-DATA[MAC-SIZE]
ACTUAL-DATA[N]
PADDING-DATA[PADDING]

ACTUAL-DATA is the actual data being transmitted (the message payload). PADDING-DATA is the padding data sent when a block cipher is used and padding is needed. Finally, MAC-DATA is the "Message Authentication Code".

When SSL records are sent in the clear, no cipher is used.Consequently the amount of PADDING-DATA will be zero and the amount of MAC-DATA will be zero. When encryption is in effect, the PADDING-DATA will be a function of the cipher block size. The MAC-DATA is a function of the CIPHER-CHOICE (more about that later).

The MAC-DATA is computed as follows:

MAC-DATA = HASH[SECRET, ACTUAL-DATA, PADDING-DATA, SEQUENCE-NUMBER]

Where the SECRET data is fed to the hash function first, followed by the ACTUAL-DATA, which is followed by the PADDING-DATA which is finally followed by the SEQUENCE-NUMBER. The SEQUENCE-NUMBER is a 32 bit value which is presented to the hash function as four bytes, with the first byte being the most significant byte of the sequence number, the second byte being the next most significant byte of the sequence number, the third byte being the third most significant byte, and the fourth byte being the least significant byte (that is, in network byte order or "big endian" order).

MAC-SIZE is a function of the digest algorithm being used. For MD2 and MD5 the MAC-SIZE will be 16 bytes (128 bits).

The SECRET value is a function of which party is sending the message. If the client is sending the message then the SECRET is the CLIENT-WRITE-KEY (the server will use the SERVER-READ-KEY to verify the MAC). If the client is receiving the message then the SECRET is the CLIENT-READ-KEY (the server will use the SERVER-WRITE-KEY to generate the MAC).

The SEQUENCE-NUMBER is a counter which is incremented by both the sender and the receiver. For each transmission direction, a pair of counters is kept (one by the sender, one by the receiver). Every time a message is sent by a sender the counter is incremented. Sequence numbers are 32 bit unsigned quantities and must wrap to zero after incrementing past 0xFFFFFFFF.

The receiver of a message uses the expected value of the sequence number as input into the MAC HASH function (the HASH function is chosen from the CIPHER-CHOICE). The computed MAC-DATA must agree bit for bit with the transmitted MAC-DATA. If the comparison is not identity then the record is considered damaged, and it is to be treated as if an "I/O Error" had occurred (i.e. an unrecoverable error is asserted and the connection is closed).

A final consistency check is done when a block cipher is used and the protocol is using encryption. The amount of data present in a record (RECORD-LENGTH))must be a multiple of the cipher's block size. If the received record is not a multiple of the cipher's block size then the record is considered damaged, and it is to be treated as if an "I/O Error" had occurred (i.e. an unrecoverable error is asserted and the connection is closed).

The SSL Record Layer is used for all SSL communications, including handshake messages, security escapes and application data transfers. The SSL Record Layer is used by both the client and the server at all times.

For a two byte header, the maximum record length is 32767 bytes. For the three byte header, the maximum record length is 16383 bytes. The SSL Handshake Protocol messages are constrained to fit in a single SSL Record Protocol record. Application protocol messages are allowed to consume multiple SSL Record Protocol record's.

Before the first record is sent using SSL all sequence numbers are initialized to zero. The transmit sequence number is incremented after every message sent, starting with the CLIENT-HELLO and SERVER-HELLO messages.

5. SSL Handshake Protocol Specification

5.1 SSL Handshake Protocol Flow

The SSL Handshake Protocol is used to negotiate security enhancements to data sent using the SSL Record Protocol. The security enhancements consist of authentication, symmetric encryption, and message integrity.

The SSL Handshake Protocol has two major phases. The first phase is used to establish private communications. The second phase is used for client authentication.

5.1.1 Phase 1

The first phase is the initial connection phase where both parties communicate their "hello" messages. The client initiates the conversation by sending the CLIENT-HELLO message. The server receives the

CLIENT-HELLO message and processes it responding with the SERVER-HELLO message.

At this point both the client and server have enough information to know whether or not a new "master key" is needed (The master key is used for production of the symmetric encryption session keys). When a new master key is not needed, both the client and the server proceed immediately to phase 2.

When a new master key is needed, the SERVER-HELLO message will contain enough information for the client to generate it. This includes the server's signed certificate (more about that later), a list of bulk cipher specifications (see below), and a connection-id (a connection-id is a randomly generated value generated by the server that is used by the client and server during a single connection). The client generates the master key and responds with a CLIENT-MASTER-KEY message (or an ERROR message if the server information indicates that the client and server cannot agree on a bulk cipher).

It should be noted here that each SSL endpoint uses a pair of ciphers per connection (for a total of four ciphers). At each endpoint, one cipher is used for outgoing communications, and one is used for incoming communications. When the client or server generate a session key, they actually generate two keys, the SERVER-READ-KEY (also known as the CLIENT-WRITE-KEY) and the SERVER-WRITE-KEY (also known as the CLIENT-READ-KEY). The master key is used by the client and server to generate the various session keys (more about that later).

Finally, the server sends a SERVER-VERIFY message to the client after the master key has been determined. This final step authenticates the server, because only a server which has the appropriate public key can know the master key.

5.1.2 Phase 2

The second phase is the client authentication phase. The server has already been authenticated by the client in the first phase, so this phase is primarily used to authenticate the client. In a typical scenario, the server will require authentication of the client and send a REQUEST-CERTIFICATE message. The client will answer in the positive if it has the needed information, or send an ERROR message if it does not. This protocol specification does not define the semantics of an ERROR response to a server request (e.g., an implementation can ignore the error, close the connection, etc. and still conform to this specification). In addition, it is permissable for a server to cache client authentication information with the "session-id" cache. The server is not required to re-authenticate the client on every connection.

When a party is done authenticating the other party, it sends its finished message. For the client, the CLIENT-FINISHED message contains the encrypted form of the CONNECTION-ID for the server to verify. If the verification fails, the server sends an ERROR message.

Once a party has sent its finished message it must continue to listen to its peers messages until it too receives a finished message. Once a party has

both sent a finished message and received its peers finished message, the SSL handshake protocol is done. At this point the application protocol begins to operate (Note: the application protocol continues to be layered on the SSL Record Protocol).

5.2 Typical Protocol Message Flow

The following sequences define several typical protocol message flows for the SSL Handshake Protocol. In these examples we have two principals in the conversation: the client and the server. We use a notation commonly found in the literature [10]. When something is enclosed in curly braces "{something}key" then the something has been encrypted using "key".

5.2.1 Assuming no session-identifier

```
client-hello            C -> S: challenge, cipher_specs
server-hello            S -> C: connection-id, server_certificate,
                                  cipher_specs
client-master-key       C -> S: {master_key}server_public_key
client-finish           C -> S: {connection-id}client_write_key
server-verify           S -> C: {challenge}server_write_key
server-finish           S -> C: {new_session_id}server_write_key
```

5.2.2 Assuming a session-identifier was found by both client & server

```
client-hello            C -> S: challenge, session_id, cipher_specs
server-hello            S -> C: connection-id, session_id_hit
client-finish           C -> S: {connection-id}client_write_key
server-verify           S -> C: {challenge}server_write_key
server-finish           S -> C: {session_id}server_write_key
```

5.2.3 Assuming a session-identifier was used and client authentication is used

```
client-hello            C -> S: challenge, session_id, cipher_specs
server-hello            S -> C: connection-id, session_id_hit
client-finish           C -> S: {connection-id}client_write_key
server-verify           S -> C: {challenge}server_write_key
request-certificate     S -> C: {auth_type,challenge'}
                                  server_write_key
client-certificate      C -> S: {cert_type,client_cert,
                                   response_data}client_write_key
server-finish           S -> C: {session_id}server_write_key
```

In this last exchange, the response_data is a function of the auth_type.

5.3 Errors

Error handling in the SSL connection protocol is very simple. When an error is detected, the detecting party sends a message to the other party. Errors that are not recoverable cause the client and server to abort the secure connection. Servers and client are required to "forget" any session-identifiers associated with a failing connection.

The SSL Handshake Protocol defines the following errors:

NO-CIPHER-ERROR
This error is returned by the client to the server when it cannot find a cipher or key size that it supports that is also supported by the server. This error is not recoverable.

NO-CERTIFICATE-ERROR
When a REQUEST-CERTIFICATE message is sent, this error may be returned if the client has no certificate to reply with. This error is recoverable (for client authentication only).

BAD-CERTIFICATE-ERROR
This error is returned when a certificate is deemed bad by the receiving party. Bad means that either the signature of the certificate was bad or that the values in the certificate were inappropriate (e.g. a name in the certificate did not match the expected name). This error is recoverable (for client authentication only).

UNSUPPORTED-CERTIFICATE-TYPE-ERROR
This error is returned when a client/server receives a certificate type that it can't support. This error is recoverable (for client authentication only).

5.4 SSL Handshake Protocol Messages

The SSL Handshake Protocol messages are encapsulated using the SSL Record Protocol and are composed of two parts: a single byte message type code, and some data. The client and server exchange messages until both ends have sent their "finished" message, indicating that they are satisfied with the SSL Handshake Protocol conversation. While one end may be finished, the other may not, therefore the finished end must continue to receive SSL Handshake Protocol messages until it receives a "finished" message from its peer.

After the pair of session keys has been determined by each party, the message bodies are encrypted. For the client, this happens after it verifies the session-identifier or creates a new master key and has sent it to the server. For the server, this happens after the session-identifier is found to be good, or the server receives the client's master key message.

The following notation is used for SSLHP messages:

```
char MSG-EXAMPLE
char FIELD1
char FIELD2
char THING-MSB
char THING-LSB
char THING-DATA[(MSB<<8)|LSB];
...
```

This notation defines the data in the protocol message, including the message type code. The order is presented top to bottom, with the top most element being transmitted first, and the bottom most element transferred last.

For the "THING-DATA" entry, the MSB and LSB values are actually THING-MSB and THING-LSB (respectively) and define the number of bytes of data actually present in the message. For example, if THING-MSB were zero and THING-LSB were 8 then the THING-DATA array would be exactly 8 bytes long. This shorthand is used below.

Length codes are unsigned values, and when the MSB and LSB are combined the result is an unsigned value. Unless otherwise specified lengths values are "length in bytes".

5.5 Client Only Protocol Messages

There are several messages that are only generated by clients. These messages are never generated by correctly functioning servers. A client receiving such a message closes the connection to the server and returns an error status to the application through some unspecified mechanism.

5.5.1 CLIENT-HELLO (Phase 1; Sent in the clear)

```
char MSG-CLIENT-HELLO
char CLIENT-VERSION-MSB
char CLIENT-VERSION-LSB
char CIPHER-SPECS-LENGTH-MSB
char CIPHER-SPECS-LENGTH-LSB
char SESSION-ID-LENGTH-MSB
char SESSION-ID-LENGTH-LSB
char CHALLENGE-LENGTH-MSB
char CHALLENGE-LENGTH-LSB
char CIPHER-SPECS-DATA[(MSB<<8)|LSB]
char SESSION-ID-DATA[(MSB<<8)|LSB]
char CHALLENGE-DATA[(MSB<<8)|LSB]
```

When a client first connects to a server it is required to send the CLIENT-HELLO message. The server is expecting this message from the client as its first message. It is an error for a client to send anything else as its first message.

The client sends to the server its SSL version, its cipher specs (see below), some challenge data, and the session-identifier data. The session-identifier data is only sent if the client found a session-identifier in its cache for the server, and the SESSION-ID-LENGTH will be non-zero. When there is no session-identifier for the server SESSION-ID-LENGTH must be zero.

The challenge data is used to authenticate the server. After the client and server agree on a pair of session keys, the server returns a SERVER-VERIFY message with the encrypted form of the CHALLENGE-DATA.

Also note that the server will not send its SERVER-HELLO message until it has received the CLIENT-HELLO message. This is done so that the server can indicate the status of the client's session-identifier back to the client in the server's first message (i.e. to increase protocol efficiency and reduce the number of round trips required).

The server examines the CLIENT-HELLO message and will verify that it can support the client version and one of the client cipher specs. The server can optionally edit the cipher specs, removing any entries it doesn't

choose to support. The edited version will be returned in the SERVER-HELLO message if the session-identifier is not in the server's cache.

The CIPHER-SPECS-LENGTH must be greater than zero and a multiple of 3. The SESSION-ID-LENGTH must either be zero or 16. The CHALLENGE-LENGTH must be greater than or equal to 16 and less than or equal to 32.

This message must be the first message sent by the client to the server. After the message is sent the client waits for a SERVER-HELLO message. Any other message returned by the server (other than ERROR) is disallowed.

5.5.2 CLIENT-MASTER-KEY (Phase 1; Sent primarily in the clear)

```
char MSG-CLIENT-MASTER-KEY
char CIPHER-KIND[3]
char CLEAR-KEY-LENGTH-MSB
char CLEAR-KEY-LENGTH-LSB
char ENCRYPTED-KEY-LENGTH-MSB
char ENCRYPTED-KEY-LENGTH-LSB
char KEY-ARG-LENGTH-MSB
char KEY-ARG-LENGTH-LSB
char CLEAR-KEY-DATA[MSB<<8|LSB]
char ENCRYPTED-KEY-DATA[MSB<<8|LSB]
char KEY-ARG-DATA[MSB<<8|LSB]
```

The client sends this message when it has determined a master key for the server to use. Note that when a session-identifier has been agreed upon, this message is not sent.

The CIPHER-KIND field indicates which cipher was chosen from the server's CIPHER-SPECS.

The CLEAR-KEY-DATA contains the clear portion of the MASTER-KEY. The CLEAR-KEY-DATA is combined with the SECRET-KEY-DATA (described shortly) to form the MASTER-KEY, with the SECRET-KEY-DATA being the least significant bytes of the final MASTER-KEY. The ENCRYPTED-KEY-DATA contains the secret portions of the MASTER-KEY, encrypted using the server's public key. The encryption block is formatted using block type 2 from PKCS#1 [5].

The data portion of the block is formatted as follows:

char SECRET-KEY-DATA[SECRET-LENGTH]

SECRET-LENGTH is the number of bytes of each session key that is being transmitted encrypted. The SECRET-LENGTH plus the CLEAR-KEY-LENGTH equals the number of bytes present in the cipher key (as defined by the CIPHER-KIND). It is an error if the SECRET-LENGTH found after decrypting the PKCS#1 formatted encryption block doesn't match the expected value. It is also an error if CLEAR-KEY-LENGTH is non-zero and the CIPHER-KIND is not an export cipher.

If the key algorithm needs an argument (for example, DES-CBC's initialization vector) then the KEY-ARG-LENGTH fields will be non-

zero and the KEY-ARG-DATA will contain the relevant data. For the
SSL_CK_RC2_128_CBC_WITH_MD5,
SSL_CK_RC2_128_CBC_EXPORT40_WITH_MD5,
SSL_CK_IDEA_128_CBC_WITH_MD5,
SSL_CK_DES_64_CBC_WITH_MD5 and
SSL_CK_DES_192_EDE3_CBC_WITH_MD5 algorithms the KEY-ARG
data must be present and be exactly 8 bytes long.

Client and server session key production is a function of the CIPHER-CHOICE:

SSL_CK_RC4_128_WITH_MD5
SSL_CK_RC4_128_EXPORT40_WITH_MD5
SSL_CK_RC2_128_CBC_WITH_MD5
SSL_CK_RC2_128_CBC_EXPORT40_WITH_MD5
SSL_CK_IDEA_128_CBC_WITH_MD5

KEY-MATERIAL-0 = MD5[MASTER-KEY, "0", CHALLENGE,
CONNECTION-ID]
KEY-MATERIAL-1 = MD5[MASTER-KEY, "1", CHALLENGE,
CONNECTION-ID]

CLIENT-READ-KEY = KEY-MATERIAL-0[0-15]
CLIENT-WRITE-KEY = KEY-MATERIAL-1[0-15]

Where KEY-MATERIAL-0[0-15] means the first 16 bytes of the KEY-MATERIAL-0 data, with KEY-MATERIAL-0[0] becoming the most significant byte of the CLIENT-READ-KEY.

Data is fed to the MD5 hash function in the order shown, from left to right: first the MASTER-KEY, then the "0" or "1", then the CHALLENGE and then finally the CONNECTION-ID.

Note that the "0" means the ascii zero character (0x30), not a zero value. "1" means the ascii 1 character (0x31). MD5 produces 128 bits of output data which are used directly as the key to the cipher algorithm (The most significant byte of the MD5 output becomes the most significant byte of the key material).

SSL_CK_DES_64_CBC_WITH_MD5

KEY-MATERIAL-0 = MD5[MASTER-KEY, CHALLENGE,
CONNECTION-ID]

CLIENT-READ-KEY = KEY-MATERIAL-0[0-7]
CLIENT-WRITE-KEY = KEY-MATERIAL-0[8-15]

For DES-CBC, a single 16 bytes of key material are produced using MD5. The first 8 bytes of the MD5 digest are used as the CLIENT-READ-KEY while the remaining 8 bytes are used as the CLIENT-WRITE-KEY. The initialization vector is provided in the KEY-ARG-DATA. Note that the raw key data is not parity adjusted and that this step must be performed before the keys are legitimate DES keys.

SSL_CK_DES_192_EDE3_CBC_WITH_MD5

```
 KEY-MATERIAL-0 = MD5[ MASTER-KEY, "0", CHALLENGE,
CONNECTION-ID ]
 KEY-MATERIAL-1 = MD5[ MASTER-KEY, "1", CHALLENGE,
CONNECTION-ID ]
 KEY-MATERIAL-2 = MD5[ MASTER-KEY, "2", CHALLENGE,
CONNECTION-ID ]

 CLIENT-READ-KEY-0 = KEY-MATERIAL-0[0-7]
 CLIENT-READ-KEY-1 = KEY-MATERIAL-0[8-15]
 CLIENT-READ-KEY-2 = KEY-MATERIAL-1[0-7]
 CLIENT-WRITE-KEY-0 = KEY-MATERIAL-1[8-15]
 CLIENT-WRITE-KEY-1 = KEY-MATERIAL-2[0-7]
 CLIENT-WRITE-KEY-2 = KEY-MATERIAL-2[8-15]
```

Data is fed to the MD5 hash function in the order shown, from left to right: first the MASTER-KEY, then the "0", "1" or "2", then the CHALLENGE and then finally the CONNECTION-ID. Note that the "0" means the ascii zero character (0x30), not a zero value. "1" means the ascii 1 character (0x31). "2" means the ascii 2 character (0x32).

A total of 6 keys are produced, 3 for the read side DES-EDE3 cipher and 3 for the write side DES-EDE3 function. The initialization vector is provided in the KEY-ARG-DATA. The keys that are produced are not parity adjusted. This step must be performed before proper DES keys are usable.

Recall that the MASTER-KEY is given to the server in the CLIENT-MASTER-KEY message. The CHALLENGE is given to the server by the client in the CLIENT-HELLO message. The CONNECTION-ID is given to the client by the server in the SERVER-HELLO message. This makes the resulting cipher keys a function of the original session and the current session. Note that the master key is never directly used to encrypt data, and therefore cannot be easily discovered.

The CLIENT-MASTER-KEY message must be sent after the CLIENT-HELLO message and before the CLIENT-FINISHED message. The CLIENT-MASTER-KEY message must be sent if the SERVER-HELLO message contains a SESSION-ID-HIT value of 0.

5.5.3 CLIENT-CERTIFICATE (Phase 2; Sent encrypted)

```
char MSG-CLIENT-CERTIFICATE
char CERTIFICATE-TYPE
char CERTIFICATE-LENGTH-MSB
char CERTIFICATE-LENGTH-LSB
char RESPONSE-LENGTH-MSB
char RESPONSE-LENGTH-LSB
char CERTIFICATE-DATA[MSB<<8|LSB]
char RESPONSE-DATA[MSB<<8|LSB]
```

This message is sent by one an SSL client in response to a server REQUEST-CERTIFICATE message. The CERTIFICATE-DATA contains data defined by the CERTIFICATE-TYPE value. An ERROR message is sent with error code NO-CERTIFICATE-ERROR when this request cannot be answered properly (e.g. the receiver of the message has no registered certificate).

CERTIFICATE-TYPE is one of:

SSL_X509_CERTIFICATE
The CERTIFICATE-DATA contains an X.509 (1988) [3] signed certificate.

The RESPONSE-DATA contains the authentication response data. This data is a function of the AUTHENTICATION-TYPE value sent by the server.

When AUTHENTICATION-TYPE is SSL_AT_MD5_WITH_RSA_ENCRYPTION then the RESPONSE-DATA contains a digital signature of the following components (in the order shown):

the KEY-MATERIAL-0
the KEY-MATERIAL-1 (only if defined by the cipher kind)
the KEY-MATERIAL-2 (only if defined by the cipher kind)
the CERTIFICATE-CHALLENGE-DATA (from the REQUEST-CERTIFICATE message)
the server's signed certificate (from the SERVER-HELLO message)

The digital signature is constructed using MD5 and then encrypted using the clients private key, formatted according to PKCS#1's digital signature standard [5]. The server authenticates the client by verifying the digital signature using standard techniques. Note that other digest functions are supported. Either a new AUTHENTICATION-TYPE can be added, or the algorithm-id in the digital signature can be changed.

This message must be sent by the client only in response to a REQUEST-CERTIFICATE message.

5.5.4 CLIENT-FINISHED (Phase 2; Sent encrypted)

```
char MSG-CLIENT-FINISHED
char CONNECTION-ID[N-1]
```

The client sends this message when it is satisfied with the server. Note that the client must continue to listen for server messages until it receives a SERVER-FINISHED message. The CONNECTION-ID data is the original connection-identifier the server sent with its SERVER-HELLO message, encrypted using the agreed upon session key.

"N" is the number of bytes in the message that was sent, so "N-1" is the number of bytes in the message without the message header byte.

For version 2 of the protocol, the client must send this message after it has received the SERVER-HELLO message. If the SERVER-HELLO message SESSION-ID-HIT flag is non-zero then the CLIENT-FINISHED message is sent immediately, otherwise the CLIENT-FINISHED message is sent after the CLIENT-MASTER-KEY message.

5.6 Server Only Protocol Messages

There are several messages that are only generated by servers. The messages are never generated by correctly functioning clients.

5.6.1 SERVER-HELLO (Phase 1; Sent in the clear)

```
char MSG-SERVER-HELLO
char SESSION-ID-HIT
char CERTIFICATE-TYPE
char SERVER-VERSION-MSB
char SERVER-VERSION-LSB
char CERTIFICATE-LENGTH-MSB
char CERTIFICATE-LENGTH-LSB
char CIPHER-SPECS-LENGTH-MSB
char CIPHER-SPECS-LENGTH-LSB
char CONNECTION-ID-LENGTH-MSB
char CONNECTION-ID-LENGTH-LSB
char CERTIFICATE-DATA[MSB<<8|LSB]
char CIPHER-SPECS-DATA[MSB<<8|LSB]
char CONNECTION-ID-DATA[MSB<<8|LSB]
```

The server sends this message after receiving the clients CLIENT-HELLO message. The server returns the SESSION-ID-HIT flag indicating whether or not the received session-identifier is known by the server (i.e. in the server's session-identifier cache). The SESSION-ID-HIT flag will be non-zero if the client sent the server a session-identifier (in the CLIENT-HELLO message with SESSION-ID-LENGTH != 0) and the server found the client's session-identifier in its cache. If the SESSION-ID-HIT flag is non-zero then the CERTIFICATE-TYPE, CERTIFICATE-LENGTH and CIPHER-SPECS-LENGTH fields will be zero.

The CERTIFICATE-TYPE value, when non-zero, has one of the values described above (see the information on the CLIENT-CERTIFICATE message).

When the SESSION-ID-HIT flag is zero, the server packages up its certificate, its cipher specs and a connection-id to send to the client. Using this information the client can generate a session key and return it to the server with the CLIENT-MASTER-KEY message.

When the SESSION-ID-HIT flag is non-zero, both the server and the client compute a new pair of session keys for the current session derived from the MASTER-KEY that was exchanged when the SESSION-ID was created. The SERVER-READ-KEY and SERVER-WRITE-KEY are derived from the original MASTER-KEY keys in the same manner as the CLIENT-READ-KEY and CLIENT-WRITE-KEY:

```
SERVER-READ-KEY = CLIENT-WRITE-KEY
SERVER-WRITE-KEY = CLIENT-READ-KEY
```

Note that when keys are being derived and the SESSION-ID-HIT flag is set and the server discovers the client's session-identifier in the servers cache, then the KEY-ARG-DATA is used from the time when the

SESSION-ID was established. This is because the client does not send new KEY-ARG-DATA (recall that the KEY-ARG-DATA is sent only in the CLIENT-MASTER-KEY message).

The CONNECTION-ID-DATA is a string of randomly generated bytes used by the server and client at various points in the protocol. The CLIENT-FINISHED message contains an encrypted version of the CONNECTION-ID-DATA. The length of the CONNECTION-ID must be between 16 and than 32 bytes, inclusive.

The CIPHER-SPECS-DATA define a cipher type and key length (in bits) that the receiving end supports. Each SESSION-CIPHER-SPEC is 3 bytes long and looks like this:

```
char CIPHER-KIND-0
char CIPHER-KIND-1
char CIPHER-KIND-2
```

Where CIPHER-KIND is one of:

```
SSL_CK_RC4_128_WITH_MD5
SSL_CK_RC4_128_EXPORT40_WITH_MD5
SSL_CK_RC2_128_CBC_WITH_MD5
SSL_CK_RC2_128_CBC_EXPORT40_WITH_MD5
SSL_CK_IDEA_128_CBC_WITH_MD5
SSL_CK_DES_64_CBC_WITH_MD5
SSL_CK_DES_192_EDE3_CBC_WITH_MD5
```

This list is not exhaustive and may be changed in the future.

The SSL_CK_RC4_128_EXPORT40_WITH_MD5 cipher is an RC4 cipher where some of the session key is sent in the clear and the rest is sent encrypted (exactly 40 bits of it). MD5 is used as the hash function for production of MAC's and session key's. This cipher type is provided to support "export" versions (i.e. versions of the protocol that can be distributed outside of the United States) of the client or server.

An exportable implementation of the SSL Handshake Protocol will have secret key lengths restricted to 40 bits. For non-export implementations key lengths can be more generous (we recommend at least 128 bits). It is permissible for the client and server to have a non-intersecting set of stream ciphers. This, simply put, means they cannot communicate.

Version 2 of the SSL Handshake Protocol defines the SSL_CK_RC4_128_WITH_MD5 to have a key length of 128 bits. The SSL_CK_RC4_128_EXPORT40_WITH_MD5 also has a key length of **128 bits. However, only 40 of the bits are secret (the other 88 bits are sent** in the clear by the client to the server).

The SERVER-HELLO message is sent after the server receives the CLIENT-HELLO message, and before the server sends the SERVER-VERIFY message.

5.6.2 SERVER-VERIFY (Phase 1; Sent encrypted)

```
char MSG-SERVER-VERIFY
```

```
char CHALLENGE-DATA[N-1]
```

The server sends this message after a pair of session keys (SERVER-READ-KEY and SERVER-WRITE-KEY) have been agreed upon either by a session-identifier or by explicit specification with the CLIENT-MASTER-KEY message. The message contains an encrypted copy of the CHALLENGE-DATA sent by the client in the CLIENT-HELLO message.

"N" is the number of bytes in the message that was sent, so "N-1" is the number of bytes in the CHALLENGE-DATA without the message header byte.

This message is used to verify the server as follows. A legitimate server will have the private key that corresponds to the public key contained in the server certificate that was transmitted in the SERVER-HELLO message. Accordingly, the legitimate server will be able to extract and reconstruct the pair of session keys (SERVER-READ-KEY and SERVER-WRITE-KEY). Finally, only a server that has done the extraction and decryption properly can correctly encrypt the CHALLENGE-DATA. This, in essence, "proves" that the server has the private key that goes with the public key in the server's certificate.

The CHALLENGE-DATA must be the exact same length as originally sent by the client in the CLIENT-HELLO message. Its value must match exactly the value sent in the clear by the client in the CLIENT-HELLO message. The client must decrypt this message and compare the value received with the value sent, and only if the values are identical is the server to be "trusted". If the lengths do not match or the value doesn't match then the connection is to be closed by the client.

This message must be sent by the server to the client after either detecting a session-identifier hit (and replying with a SERVER-HELLO message with SESSION-ID-HIT not equal to zero) or when the server receives the CLIENT-MASTER-KEY message. This message must be sent before any Phase 2 messages or a SERVER-FINISHED message.

5.6.3 SERVER-FINISHED (Phase 2; Sent encrypted)

```
char MSG-SERVER-FINISHED
char SESSION-ID-DATA[N-1]
```

The server sends this message when it is satisfied with the clients security handshake and is ready to proceed with transmission/reception of the higher level protocols data. The SESSION-ID-DATA is used by the client and the server at this time to add entries to their respective session-identifier caches. The session-identifier caches must contain a copy of the MASTER-KEY sent in the CLIENT-MASTER-KEY message as the master key is used for all subsequent session key generation.

"N" is the number of bytes in the message that was sent, so "N-1" is the number of bytes in the SESSION-ID-DATA without the message header byte.

This message must be sent after the SERVER-VERIFY message.

5.6.4 REQUEST-CERTIFICATE (Phase 2; Sent encrypted)

```
char MSG-REQUEST-CERTIFICATE
char AUTHENTICATION-TYPE
char CERTIFICATE-CHALLENGE-DATA[N-2]
```

A server may issue this request at any time during the second phase of the connection handshake, asking for the client's certificate. The client responds with a CLIENT-CERTIFICATE message immediately if it has one, or an ERROR message (with error code NO-CERTIFICATE-ERROR) if it doesn't. The CERTIFICATE-CHALLENGE-DATA is a short byte string (whose length is greater than or equal to 16 bytes and less than or equal to 32 bytes) that the client will use to respond to this message.

The AUTHENTICATION-TYPE value is used to choose a particular means of authenticating the client. The following types are defined:

SSL_AT_MD5_WITH_RSA_ENCRYPTION

The SSL_AT_MD5_WITH_RSA_ENCRYPTION type requires that the client construct an MD5 message digest using information as described above in the section on the CLIENT-CERTIFICATE message. Once the digest is created, the client encrypts it using its private key (formatted according to the digital signature standard defined in PKCS#1). The server authenticates the client when it receives the CLIENT-CERTIFICATE message.

This message may be sent after a SERVER-VERIFY message and before a SERVER-FINISHED message.

5.7 Client/Server Protocol Messages

These messages are generated by both the client and the server.

5.7.1 ERROR (Sent clear or encrypted)

```
char MSG-ERROR
char ERROR-CODE-MSB
char ERROR-CODE-LSB
```

This message is sent when an error is detected. After the message is sent, the sending party shuts the connection down. The receiving party records the error and then shuts its connection down.

This message is sent in the clear if an error occurs during session key negotiation. After a session key has been agreed upon, errors are sent encrypted like all other messages.

Appendix A: ASN.1 Syntax For Certificates

Certificates are used by SSL to authenticate servers and clients. SSL Certificates are based largely on the X.509 [3] certificates. An X.509 certificate contains the following information (in ASN.1 [1] notation):

```
X.509-Certificate ::= SEQUENCE {
  certificateInfo CertificateInfo,
  signatureAlgorithm AlgorithmIdentifier,
  signature BIT STRING
}

CertificateInfo ::= SEQUENCE {
  version [0] Version DEFAULT v1988,
  serialNumber CertificateSerialNumber,
  signature AlgorithmIdentifier,
  issuer Name,
  validity Validity,
  subject Name,
  subjectPublicKeyInfo SubjectPublicKeyInfo
}

Version ::= INTEGER { v1988(0) }

CertificateSerialNumber ::= INTEGER

Validity ::= SEQUENCE {
  notBefore UTCTime,
  notAfter UTCTime
}

SubjectPublicKeyInfo ::= SEQUENCE {
  algorithm AlgorithmIdentifier,
  subjectPublicKey BIT STRING
}

AlgorithmIdentifier ::= SEQUENCE {
  algorithm OBJECT IDENTIFIER,
  parameters ANY DEFINED BY ALGORITHM OPTIONAL
}
```

For SSL's purposes we restrict the values of some of the X.509 fields:

The X.509-Certificate::signatureAlgorithm and CertificateInfo::signature fields must be identical in value.

The issuer name must resolve to a name that is deemed acceptable by the application using SSL. How the application using SSL does this is outside the scope of this memo.

Certificates are validated using a few straightforward steps. First, the signature on the certificate is checked and if invalid, the certificate is invalid (either a transmission error or an attempted forgery occurred). Next, the CertificateInfo::issuer field is verified to be an issuer that the application trusts (using an unspecified mechanism). The CertificateInfo::validity field is checked against the current date and verified.

Finally, the CertificateInfo::subject field is checked. This check is optional and depends on the level of trust required by the application using SSL.

Appendix B: Attribute Types and Object Identifiers

SSL uses a subset of the X.520 selected attribute types as well as a few specific object identifiers. Future revisions of the SSL protocol may include support for more attribute types and more object identifiers.

B.1 Selected attribute types

commonName { attributeType 3 }
The common name contained in the distinguished name contained within a certificate issuer or certificate subject.

countryName { attributeType 6 }
The country name contained in the distinguished name contained within a certificate issuer or certificate subject.

localityName { attributeType 7 }
The locality name contained in the distinguished name contained within a certificate issuer or certificate subject.

stateOrProvinceName { attributeType 8 }
The state or province name contained in the distinguished name contained within a certificate issuer or certificate subject.

organizationName { attributeType 10 }
The organization name contained in the distinguished name contained within a certificate issuer or certificate subject.

organizationalUnitName { attributeType 11 }
The organizational unit name contained in the distinguished name contained within a certificate issuer or certificate subject.

B.2 Object identifiers

md2withRSAEncryption { ... pkcs(1) 1 2 }
The object identifier for digital signatures that use both MD2 and RSA encryption. Used by SSL for certificate signature verification.

md5withRSAEncryption { ... pkcs(1) 1 4 }
The object identifier for digital signatures that use both MD5 and RSA encryption. Used by SSL for certificate signature verification.

rc4 { ... rsadsi(113549) 3 4 }
The RC4 symmetric stream cipher algorithm used by SSL for bulk encryption.

Appendix C: Protocol Constant Values

This section describes various protocol constants. A special value needs mentioning - the IANA reserved port number for "https" (HTTP using SSL). IANA has reserved port number 443 (decimal) for "https". IANA has also reserved port number 465 for "ssmtp" and port number 563 for "snntp".

C.1 Protocol Version Codes

```
#define SSL_CLIENT_VERSION 0x0002
#define SSL_SERVER_VERSION 0x0002
```

C.2 Protocol Message Codes

The following values define the message codes that are used by version **2 of the SSL Handshake Protocol.**

```
#define SSL_MT_ERROR                    0
#define SSL_MT_CLIENT_HELLO             1
#define SSL_MT_CLIENT_MASTER_KEY        2
#define SSL_MT_CLIENT_FINISHED          3
#define SSL_MT_SERVER_HELLO             4
#define SSL_MT_SERVER_VERIFY            5
#define SSL_MT_SERVER_FINISHED          6
#define SSL_MT_REQUEST_CERTIFICATE      7
#define SSL_MT_CLIENT_CERTIFICATE       8
```

C.3 Error Message Codes

The following values define the error codes used by the ERROR message.

```
#define SSL_PE_NO_CIPHER                        0x0001
#define SSL_PE_NO_CERTIFICATE                   0x0002
#define SSL_PE_BAD_CERTIFICATE                  0x0004
#define SSL_PE_UNSUPPORTED_CERTIFICATE_TYPE     0x0006
```

C.4 Cipher Kind Values

The following values define the CIPHER-KIND codes used in the CLIENT-HELLO and SERVER-HELLO messages.

```
#define SSL_CK_RC4_128_WITH_MD5
        0x01,0x00,0x80
#define SSL_CK_RC4_128_EXPORT40_WITH_MD5
        0x02,0x00,0x80
#define SSL_CK_RC2_128_CBC_WITH_MD5
        0x03,0x00,0x80
#define SSL_CK_RC2_128_CBC_EXPORT40_WITH_MD5
        0x04,0x00,0x80
#define SSL_CK_IDEA_128_CBC_WITH_MD5
        0x05,0x00,0x80
#define SSL_CK_DES_64_CBC_WITH_MD5
        0x06,0x00,0x40
#define SSL_CK_DES_192_EDE3_CBC_WITH_MD5
        0x07,0x00,0xC0
```

C.5 Certificate Type Codes

The following values define the certificate type codes used in the SERVER-HELLO and CLIENT-CERTIFICATE messages.

```
#define SSL_CT_X509_CERTIFICATE         0x01
```

C.6 Authentication Type Codes

The following values define the authentication type codes used in the REQUEST-CERTIFICATE message.

```
#define SSL_AT_MD5_WITH_RSA_ENCRYPTION 0x01
```

C.7 Upper/Lower Bounds

The following values define upper/lower bounds for various protocol parameters.

```
#define SSL_MAX_MASTER_KEY_LENGTH_IN_BITS       256
#define SSL_MAX_SESSION_ID_LENGTH_IN_BYTES      16
#define SSL_MIN_RSA_MODULUS_LENGTH_IN_BYTES     64
#define SSL_MAX_RECORD_LENGTH_2_BYTE_HEADER     32767
#define SSL_MAX_RECORD_LENGTH_3_BYTE_HEADER     16383
```

C.8 Recommendations

Because protocols have to be implemented to be of value, we recommend the following values for various operational parameters. This is only a recommendation, and not a strict requirement for conformance to the protocol.

Session-identifier Cache Timeout

Session-identifiers are kept in SSL clients and SSL servers. Session-identifiers should have a lifetime that serves their purpose (namely, reducing the number of expensive public key operations for a single client/server pairing). Consequently, we recommend a maximum session-identifier cache timeout value of 100 seconds. Given a server that can perform N private key operations per second, this reduces the server load for a particular client by a factor of 100.

Appendix D: Attacks

In this section we attempt to describe various attacks that might be used against the SSL protocol. This list is not guaranteed to be exhaustive. SSL was defined to thwart these attacks.

D.1 Cracking Ciphers

SSL depends on several cryptographic technologies. RSA Public Key encryption [5] is used for the exchange of the session key and client/server authentication. Various cryptographic algorithms are used for the session cipher. If successful cryptographic attacks are made against these technologies then SSL is no longer secure.

Attacks against a specific communications session can be made by recording the session, and then spending some large number of compute cycles to crack either the session key or the RSA public key until the communication can be seen in the clear. This approach is easier than cracking the cryptographic technologies for all possible messages. Note that SSL tries to make the cost of such of an attack greater than the

benefits gained from a successful attack, thus making it a waste of money/time to perform such an attack.

There have been many books [9] and papers [10] written on cryptography. This document does not attempt to reference them all.

D.2 Clear Text Attack

A clear text attack is done when the attacker has an idea of what kind of message is being sent using encryption. The attacker can generate a data base whose keys are the encrypted value of the known text (or clear text), and whose values are the session cipher key (we call this a "dictionary"). Once this data base is constructed, a simple lookup function identifies the session key that goes with a particular encrypted value. Once the session key is known, the entire message stream can be decrypted. Custom hardware can be used to make this cost effective and very fast.

Because of the very nature of SSL clear text attacks are possible. For example, the most common byte string sent by an HTTP client application to an HTTP server is "GET". SSL attempts to address this attack by using large session cipher keys. First, the client generates a key which is larger than allowed by export, and sends some of it in the clear to the server (this is allowed by United States government export rules). The clear portion of the key concatenated with the secret portion make a key which is very large (for RC4, exactly 128 bits).

The way that this "defeats" a clear text attack is by making the amount of custom hardware needed prohibitively large. Every bit added to the length of the session cipher key increases the dictionary size by a factor of 2. By using a 128 bit session cipher key length the size of the dictionary required is beyond the ability of anyone to fabricate (it would require more atoms to construct than exist in the entire universe). Even if a smaller dictionary is to be used, it must first be generated using the clear key bits. This is a time consumptive process and also eliminates many possible custom hardware architectures (e.g. static prom arrays).

The second way that SSL attacks this problem is by using large key lengths when permissible (e.g. in the non-export version). Large key sizes require larger dictionaries (just one more bit of key size doubles the size of the dictionary). SSL attempts to use keys that are 128 bits in length.

Note that the consequence of the SSL defense is that a brute force attack becomes the cheapest way to attack the key. Brute force attacks have well known space/time tradeoffs and so it becomes possible to define a cost of the attack. For the 128 bit secret key, the known cost is essentially infinite. For the 40 bit secret key, the cost is much smaller, but still outside the range of the "random hacker".

D.3 Replay

The replay attack is simple. A bad-guy records a communication session between a client and server. Later, it reconnects to the server, and plays back the previously recorded client messages. SSL defeats this attack using a "nonce" (the connection-id) which is "unique" to the connection. In theory the bad-guy cannot predict the nonce in advance as it is based on a

set of random events outside the bad-guys control, and therefore the bad-guy cannot respond properly to server requests.

A bad-guy with large resources can record many sessions between a client and a server, and attempt to choose the right session based on the nonce the server sends initially in its SERVER-HELLO message. However, SSL nonces are at least 128 bits long, so a bad-guy would need to record approximately 2^64 nonces to even have a 50% chance of choosing the right session. This number is sufficiently large that one cannot economically construct a device to record 2^64 messages, and therefore the odds are overwhelmingly against the replay attack ever being successful.

D.4 The Man In The Middle

The man in the middle attack works by having three people in a communications session: the client, the server, and the bad guy. The bad guy sits between the client and the server on the network and intercepts traffic that the client sends to the server, and traffic that the server sends to the client.

The man in the middle operates by pretending to be the real server to the client. With SSL this attack is impossible because of the usage of server certificates. During the security connection handshake the server is required to provide a certificate that is signed by a certificate authority. Contained in the certificate is the server's public key as well as its name and the name of the certificate issuer. The client verifies the certificate by first checking the signature and then verifying that the name of the issuer is somebody that the client trusts.

In addition, the server must encrypt something with the private key that goes with the public key mentioned in the certificate. This in essence is a single pass "challenge response" mechanism. Only a server that has both the certificate and the private key can respond properly to the challenge.

If the man in the middle provides a phony certificate, then the signature check will fail. If the certificate provided by the bad guy is legitimate, but for the bad guy instead of for the real server, then the signature will pass but the name check will fail (note that the man in the middle cannot forge certificates without discovering a certificate authority's private key).

Finally, if the bad guy provides the real server's certificate then the signature check will pass and the name check will pass. However, because the bad guy does not have the real server's private key, the bad guy cannot properly encode the response to the challenge code, and this check will fail.

In the unlikely case that a bad guy happens to guess the response code to the challenge, the bad guy still cannot decrypt the session key and therefore cannot examine the encrypted data.

Appendix E: Terms

Application Protocol
An application protocol is a protocol that normally layers directly on top of TCP/IP. For example: HTTP, TELNET, FTP, and SMTP.

Authentication
Authentication is the ability of one entity to determine the identity of another entity. Identity is defined by this document to mean the binding between a public key and a name and the implicit ownership of the corresponding private key.

Bulk Cipher
This term is used to describe a cryptographic technique with certain performance properties. Bulk ciphers are used when large quantities of data are to be encrypted/decrypted in a timely manner. Examples include RC2, RC4, and IDEA.

Client
In this document client refers to the application entity that is initiates a connection to a server.

CLIENT-READ-KEY
The session key that the client uses to initialize the client read cipher. This key has the same value as the SERVER-WRITE-KEY.

CLIENT-WRITE-KEY
The session key that the client uses to initialize the client write cipher. This key has the same value as the SERVER-READ-KEY.

MASTER-KEY
The master key that the client and server use for all session key generation. The CLIENT-READ-KEY, CLIENT-WRITE-KEY, SERVER-READ-KEY and SERVER-WRITE-KEY are generated from the MASTER-KEY.

MD2
MD2 [8] is a hashing function that converts an arbitrarily long data stream into a digest of fixed size. This function predates MD5 [7] which is viewed as a more robust hash function [9].

MD5
MD5 [7] is a hashing function that converts an arbitrarily long data stream into a digest of fixed size. The function has certain properties that make it useful for security, the most important of which is it's inability to be reversed.

Nonce
A randomly generated value used to defeat "playback" attacks. One party randomly generates a nonce and sends it to the other party. The receiver encrypts it using the agreed upon secret key and returns it to the sender. Because the nonce was randomly generated by the sender this defeats playback attacks because the replayer can't know in advance the nonce the sender will generate. The receiver denies connections that do not have the correctly encrypted nonce.

Non-repudiable Information Exchange

When two entities exchange information it is sometimes valuable to have a record of the communication that is non-repudiable. Neither party can then deny that the information exchange occurred. Version 2 of the SSL protocol does not support Non-repudiable information exchange.

Public Key Encryption

Public key encryption is a technique that leverages asymmetric ciphers. A public key system consists of two keys: a public key and a private key. Messages encrypted with the public key can only be decrypted with the associated private key. Conversely, messages encrypted with the private key can only be decrypted with the public key. Public key encryption tends to be extremely compute intensive and so is not suitable as a bulk cipher.

Privacy

Privacy is the ability of two entities to communicate without fear of eavesdropping. Privacy is often implemented by encrypting the communications stream between the two entities.

RC2, RC4

Proprietary bulk ciphers invented by RSA (There is no good reference to these as they are unpublished works; however, see [9]). RC2 is block cipher and RC4 is a stream cipher.

Server

The server is the application entity that responds to requests for connections from clients. The server is passive, waiting for requests from clients.

Session cipher

A session cipher is a "bulk" cipher that is capable of encrypting or decrypting arbitrarily large amounts of data. Session ciphers are used primarily for performance reasons. The session ciphers used by this protocol are symmetric. Symmetric ciphers have the property of using a single key for encryption and decryption.

Session identifier

A session identifier is a random value generated by a client that identifies itself to a particular server. The session identifier can be thought of as a handle that both parties use to access a recorded secret key (in our case a session key). If both parties remember the session identifier then the implication is that the secret key is already known and need not be negotiated.

Session key

The key to the session cipher. In SSL there are four keys that are called session keys: CLIENT-READ-KEY, CLIENT-WRITE-KEY, SERVER-READ-KEY, and SERVER-WRITE-KEY.

SERVER-READ-KEY

The session key that the server uses to initialize the server read cipher. This key has the same value as the CLIENT-WRITE-KEY.

SERVER-WRITE-KEY
The session key that the server uses to initialize the server write cipher. This key has the same value as the CLIENT-READ-KEY.

Symmetric Cipher
A symmetric cipher has the property that the same key can be used for decryption and encryption. An asymmetric cipher does not have this behavior. Some examples of symmetric ciphers: IDEA, RC2, RC4.

References

[1] CCITT. Recommendation X.208: "Specification of Abstract Syntax Notation One (ASN.1). 1988.

[2] CCITT. Recommendation X.209: "Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1). 1988.

[3] CCITT. Recommendation X.509: "The Directory - Authentication Framework". 1988.

[4] CCITT. Recommendation X.520: "The Directory - Selected Attribute Types". 1988.

[5] RSA Laboratories. PKCS #1: RSA Encryption Standard, Version 1.5, November 1993.

[6] RSA Laboratories. PKCS #6: Extended-Certificate Syntax Standard, Version 1.5, November 1993.

[7] R. Rivest. RFC 1321: The MD5 Message Digest Algorithm. April 1992.

[8] R. Rivest. RFC 1319: The MD2 Message Digest Algorithm. April 1992.

[9] B. Schneier. Applied Cryptography: Protocols, Algorithms, and Source Code in C, Published by John Wiley & Sons, Inc. 1994.

[10] M. Abadi and R. Needham. Prudent engineering practice for cryptographic protocols. 1994.

Patent Statement

This version of the SSL protocol relies on the use of patented public key encryption technology for authentication and encryption. The Internet Standards Process as defined in RFC 1310 requires a written statement from the Patent holder that a license will be made available to applicants under reasonable terms and conditions prior to approving a specification as a Proposed, Draft or Internet Standard.

The Massachusetts Institute of Technology and the Board of Trustees of the Leland Stanford Junior University have granted Public Key Partners (PKP) exclusive sub-licensing rights to the following patents issued in the

United States, and all of their corresponding foreign patents:

Cryptographic Apparatus and Method ("Diffie-Hellman")
No. 4,200,770

Public Key Cryptographic Apparatus and Method ("Hellman-Merkle")
No. 4,218,582

Cryptographic Communications System and Method ("RSA")
No. 4,405,829

Exponential Cryptographic Apparatus and Method ("Hellman-Pohlig")
No. 4,424,414

These patents are stated by PKP to cover all known methods of practicing the art of Public Key encryption, including the variations collectively known as ElGamal.

Public Key Partners has provided written assurance to the Internet Society that parties will be able to obtain, under reasonable, nondiscriminatory terms, the right to use the technology covered by these patents. This assurance is documented in RFC 1170 titled "Public Key Standards and Licenses". A copy of the written assurance dated April 20, 1990, may be obtained from the Internet Assigned Number Authority (IANA).

The Internet Society, Internet Architecture Board, Internet Engineering Steering Group and the Corporation for National Research Initiatives take no position on the validity or scope of the patents and patent applications, nor on the appropriateness of the terms of the assurance. The Internet Society and other groups mentioned above have not made any determination as to any other intellectual property rights which may apply to the practice of this standard. Any further consideration of these matters is the user's own responsibility.

Security Considerations

This entire document is about security.

Author's Address

Kipp E.B. Hickman
Netscape Communications Corp.
501 East Middlefield Rd.
Mountain View, CA 94043
kipp@netscape.com

Context Authentication Using Constrained Channels

Tim Kindberg, Kan Zhang
Internet and Mobile Systems Laboratory
HP Laboratories Palo Alto

HPL-2001-84
April 2nd, 2001*

E-mail: timothy@hpl.hp.com, kzhang@hpl.hp.com

context-aware computing, location-based computing, security, authentication

This paper presents a paradigm shift from conventional authentication—of a principal's identity—to authentication of parameters that characterise a principal's context. Location, in particular, is a highly significant contextual parameter. It is one that features in what are known as mobile, ubiquitous, pervasive and nomadic computing systems. We present a model of context authentication based on the characteristics of communication channels. As an example, we present protocols for location authentication that are based on physical channel characteristics. We conclude with a summary and discussion of the work.

* Internal Accession Date Only

Approved for External Publication

Context authentication using constrained channels

Tim Kindberg & Kan Zhang

Internet and Mobile Systems Laboratory
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA 94304, USA
timothy@hpl.hp.com, kzhang@hpl.hp.com

Abstract

This paper presents a paradigm shift from conventional authentication—of a principal's identity—to authentication of parameters that characterise a principal's context. Location, in particular, is a highly significant contextual parameter. It is one that features in what are known as mobile, ubiquitous, pervasive and nomadic computing systems. We present a model of context authentication based on the characteristics of communication channels. As an example, we present protocols for location authentication that are based on physical channel characteristics. We conclude with a summary and discussion of the work.

1 Introduction

This paper describes a model and protocols for *context authentication*: authentication of a principal's status in a certain context; in particular, its physical location. This work is part of the CoolTown project [1,2,3,4], which is investigating 'nomadic' computing systems: ones in which users, carrying wirelessly connected devices, enter places and use local services associated with those places, as well as remote services.

Conventional authentication protocols establish the identity of a principal *p* based upon the premise that only *p* possesses some secret *K*. What is really proved is possession of *K*, and the association between *p* and *K* is a given that lies outside the protocol.

However, in some circumstances we are interested in the characteristics of a principal's context, such as their location, in addition to or instead of their identity. For example:

1. To attract customers, the Kardomah Coffee House wishes to provide a service *S* only to those who are, or who have recently been, on their premises.
2. The President of Coolania wishes to take calls on the Red Phone only from those who are physically inside the inner sanctum of Hotria's centre of government.
3. A computer that provides a service inside a company's headquarters is to cease to operate if taken outside the building.
4. A public 'kiosk' computer in an airport is to erase its memory if the user, who has downloaded personal data onto it, walks away for more than *T* seconds.
5. Only users who click an 'I agree' button on a certain web page with certain contents—i.e. users who visit that virtual location—are to be provided with service *S*.
6. A government document is not to be accessible before January 1st, 2002.

We can consider a principal's context to be characterised by a set of *contextual predicates*, such as 'the location of *p* is the Kardomah cafe', 'the date of *p*'s action is 2002/1/1 or later', 'the temperature in *p*'s environment is 15C or more'. To authenticate such a contextual predicate ϕ for a principal *p* is to verify securely that $\phi(p)$.

In particular, we wish to know that the principal who issues a given request message satisfies ϕ. For example, we may require that a principal that requests a service is in a certain location, or exists within some particular interval of time. We shall show how communication channels can be the trusted artifacts that verify contextual predicates as they apply to communicating principals.

Section 2 outlines related work. Section 3 presents a model that captures the essential features of context authentication in terms of constructs called constrained communication channels. Section 4 demonstrates the application of the model to the particular case of authenticating a principal's location. Section 5 concludes.

2 Related work

Context-awareness has been identified as a key issue in nomadic computing with location being the most prominent contextual parameter [4, 5, 8]. Contextual data are usually collected via sensing technologies, e.g., GPS and the Active Badge [5]. However, to our knowledge, very little has been published on authenticating contextual data. Many sensing-based approaches are intrinsically difficult for use in authentication. For example, in the Active Badge system the badges are easily separable from the owner. The outputs of conventional (code correlating and differential) GPS receivers can be easily forged since there is no way to tell whether they were actually calculated by a GPS receiver. To make forgery difficult, Denning, et al. [6] introduced location signature sensors (LSS) to compute a location signature from the microwave signals transmitted by the GPS satellites. Location signatures are hard to forge since GPS observations are unpredictable. However, this system is vulnerable if an attacker is able to record the GPS satellite signals and re-assemble the aggregate signal with the appropriate delays for the location he is trying to spoof. Moreover, there is commercially available test equipment to simulate GPS satellites, which transmit a signal appropriate for any location entered into them.

Location or distance information has been found useful in designing cryptographic protocols. For example, the so-called "mafia frauds" [9] against identification protocols work when a fraudulent prover is able to use an honest verifier as an oracle without them noticing it. Knowing the physical distance between the prover and the verifier can prevent a fraudulent prover from using a distant honest prover. Brands and Chaum [10] introduced a distance-bounding technique to determine an upper-bound on the distance between two communicating parties by timing the delay between sending out a challenge bit and receiving back the corresponding response bit. They showed how to integrate their technique into common identification protocols and also some three-party protocols.

We are continuing from the work of Caswell and Debaty, also within the CoolTown project [4]. Caswell and Debaty introduced the idea of "establishing a user's presence by proving proximity to a known reference point within a place". They went on to present a timestamp-based protocol for location authentication. A short-range wireless beacon is used to emit a time-varying token for location authentication. A shortcoming of this approach is that the clocks of the beacon and the authenticator have to be synchronised to avoid replay attacks.

In this work, we are interested in a general model of context authentication without assuming a particular technology. As an instantiation of our model, we will present some new location authentication protocols that do not use time.

3 The model

We are interested in channels that implement a contextual constraint: ones that allow us to make inferences about the context of sending or receiving principals. In this section, we first define such channels and then show how they can be realised.

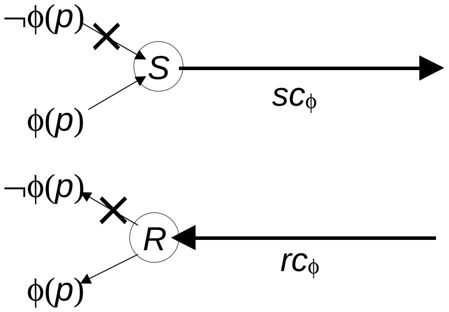


Figure 1. Send-constrained and receive-constrained channels

Any one-way channel *c* has message *send* and *receive* operations as follows:

$m = c.receive()$
$c.send(m)$.

We denote the principals that perform those operations for a uniquely identified message *m* as *receiver*(*m*) and *sender*(*m*), respectively.

A *constrained channel* is a one-way communication channel that is either *send-constrained* or *receive-constrained* or both (Figure 1):

send-constrained channel sc_ϕ on the predicate ϕ: if $m = sc_\phi.receive()$ then ϕ (*sender*(*m*)).

receive-constrained channel rc_ϕ on the predicate ϕ: ϕ(*receiver*(*m*)) for any message *m* appearing in an operation $rc_\phi.send(m)$.

These definitions capture some properties established by conventional security protocols. For example, consider two principals connected by a Transport Layer Security (TLS) connection [7], who send and receive clear-text messages that are encrypted and decrypted by the connection. Then that channel is both send-constrained and receive-constrained on the predicate 'possesses secret key *K*' for some *K* negotiated by the TLS protocol.

But constrained channels are designed to capture a much wider class of contextual predicates: any that can be established by construction of suitable hardware and software systems. Among the possibilities, an important example is a channel that imposes constraints upon the location of the communicating parties at one or other end of the channel.

The telephone system

The telephone system provides a simple example of constrained channels—at least, assuming that we were to trust certain aspects of its implementation. First, it provides receive-constrained channels on predicates of the form 'is in the location *L*'. If a principal wishes to impart some information to any principal who is in a particular place, they can do so by knowing the

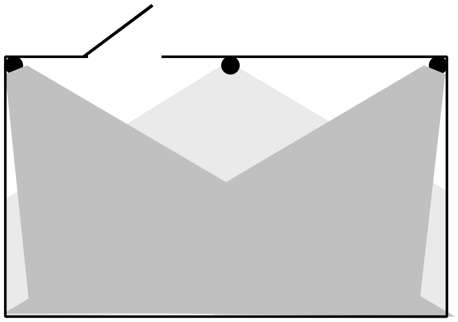


Figure 2. A room covered (mostly) by three IR beacons, which do not penetrate walls.

appropriate number to dial for a telephone fixed in the place. Assuming that we trust the integrity of the system, anyone who answers and receives that message is near to the phone. (In films, this is a technique favoured by kidnappers arranging to pick up a ransom payment!)

Caller-id provides us with send-constrained channels on predicates of the form 'is in the location *L*'. The user of a telephone may choose to answer only calls that originate from a specified telephone number—in particular, one that is wired in a fixed location. The user plus the phone system together implement a send-constrained channel.

3.1 Implementing constrained channels

The telephone system shows how we can exploit a channel with appropriate mechanical characteristics. Our telephone examples assume that the series of links and logic circuits connecting one telephone to another is proof against tampering, and that telephone circuits are not physically moved. That type of assumption is more likely to hold reliably in the case of short-range wired links in physically secured environments.

Other physical characteristics of communication channels also enable us to construct constrained channels: the speed of signal propagation and the decrease in signal strength as it propagates.

Constrained propagation

One way of establishing location information is to use network time-of-flight. In principle, with sufficiently accurate instrumentation and knowledge of real-time system parameters, we can use round-trip times to bound the location of a network node. To gauge a bound on the distance to node *M*, node *N* sends a 1-bit message to it, which *M* is to return to *N* immediately. If the speed of signal propagation is *c* then, if *M* can return the message to node *N* in time $t < (l + 2d/c)$, it is within a distance *d* of *N*, where *l* is the total communication latency imposed by software and hardware.

We can also employ wireless network segments whose range (the distance over which messages may effectively propagate) is bounded. This includes radio (e.g. bluetooth or 802.11), with a range of 10cm-1km; infrared (IR), with a range of 10cm-10m; and ultrasound, up to 10m in range.

Radio permeates walls, in general, so its reach often does not match the physical territory that we think of as a distinct location. IR [3, 5] and ultrasound [8], on the other hand, have the useful property that walls tend to attenuate their signals to a negligible level.

We may combine several short-range transmitters, if necessary (see Figure 2). Transmitters can be placed so as to cover (most of) a place such as a room, without it being practically possible to receive their signals from anywhere outside that place.

In principle, an attacker that has an arbitrarily powerful transmitter or an arbitrarily sensitive receiver may be able to flout what are normally considered to be the limits of wireless technologies. However, we can make it reasonably difficult for an attacker to do so. For example, we can control line-of-sight access to thwart directional antennae, and use highly attenuating materials.

3.2 Constrained channels as building blocks

We can use constrained channels as building blocks for making further constrained channels. We do so by inserting a proxy between two constrained channels; and by running a protocol that turns a send-constrained channel into a receive-constrained channel, or *vice versa*.

A *channel proxy P* is a process that connects exactly two one-way channels c_1 and c_2. We denote the complex of the proxy and two channels as $c_1.P.c_2$. The channel proxy receives messages from channel c_1 and selectively forwards the messages on to channel c_2. For each message it receives, it may either discard or forward it, possibly after a delay but without modifying it. The proxy may only send messages that it has received.

The complex $C = c_1.P.c_2$ behaves as a (possibly lossy) channel, if we identify $C.send \equiv c_1.send$, and $C.receive \equiv c_2.\ receive$.

One particular configuration in which we are interested is where a channel proxy *P* is connected to a channel that is send-constrained on the predicate 'sender is *P*'. We shall denote that channel as *sc*(*P*). Similarly, we can consider a proxy whose input side is a channel that is receive-constrained on the predicate 'receiver is *P*'. We shall denote that channel as *rc*(*P*).

Appending channels

We can construct new send-constrained or receive-constrained channels from others, as follows.

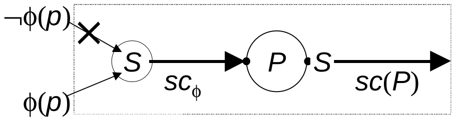


Figure 3. A channel proxy interposed to construct a new send-constrained channel.

If sc_ϕ is a send-constrained channel on the predicate ϕ, then so is the channel $C = sc_\phi.P.sc(P)$ (see Figure 3). To verify this, we must show that all senders of a message received from C satisfy ϕ. $C.receive(m)$ means $sc(P).receive(m)$. That implies, by the definition of $sc(P)$, that P sent m. But P can only send what it receives, so m arrives through an operation $sc_\phi.receive()$. Therefore, $\phi(sender(m))$.

Similarly, we can create a new receive-constrained channel from a receive-constrained channel and a channel proxy. If rc_ϕ is a receive-constrained channel on the predicate ϕ, then so is the channel $C = rc(P).P.rc_\phi$. We must show that all receivers of the message outside C satisfy ϕ. If m is sent on C then it is sent on $rc(P)$. Therefore, by definition, only P receives it. Since P may only forward m along rc_ϕ, we have that ϕ $(receiver(m))$—if P forwards m.

Channel proxies and contextual parameters

The simplest type of channel proxy forwards all the messages it receives. But we are free to use channel proxies that delay or discard messages.

By delaying messages, proxies can implement temporal constraints: a proxy could delay all messages until midnight 1/1/2002.

Another type of proxy decides whether to discard or forward messages based upon some property of a context. Suppose that, for Tim to access the Kan filing cabinet service, (a) he must be present in Kan's office and (b) Kan must also be present (so that he can observe Tim's activities). Kan installs a proxy in his office that uses a wireless network to detect the presence of users in the office. When Tim sends a message to the filing cabinet service on that channel, the proxy knows that Tim is present. But it holds onto the message until it can establish, using the same wireless channel, that Kan is also present. If so, it forwards the request. Otherwise, it discards it.

In general, we can construct channel proxies that can independently evaluate a contextual predicate ψ that applies to any principal p such that $\phi(p)$. For example, it could be a proxy that measures the set of people in a room or the temperature in the room. Employing an input channel that is send-constrained on ϕ, we can use the proxy to implement a channel constrained on $\phi\wedge\psi$.

Reversing channels

Let rc_ϕ be a receive-constrained channel on the predicate ϕ. We shall show how to construct a channel $s(rc_\phi)$, which is send-constrained on ϕ.

We use a trusted node N. When it receives a message m, it uses the receive-constrained channel to return to the sender a signed hash sig$\{h(m)\}$.

Let c be any (possibly unconstrained) channel connecting the parties that we wish to be able to communicate. We construct $s(rc_\phi)$ from c and N. The rules for sending and receiving on $s(rc_\phi)$ are as follows:

To send m on $s(rc_\phi)$:

```
    send $m$ to $N$
    receive sig$\{h(m)\}$ from $N$ over $rc_\phi$
    send $<m$, sig$\{h(m)\}>$ on $c$.
```

To receive m on $s(rc_\phi)$:

```
    receive $<m, h>$ on $c$
    verify $h$ = sig$\{h(m)\}$
    Discard $m$ if verification fails, else receive $m$.
```

In a similar fashion, we can implement a receive-constrained channel from a send-constrained channel. All messages are sent (over any channel) to a trusted node, which stores them. Receivers must use a particular send-constrained channel to reach that node, which responds with the next message for them.

4 Location authentication protocols

In the preceding, we developed a model of constrained channels and outlined how they can be constructed—from components with appropriate physical characteristics, and from other constrained channels. We now look more closely at protocols for the particular case of location authentication, filling in the more important details that are necessary for practical purposes.

A location authentication protocol enables an authenticator to verify their own location or the location of another principal. The statement 'principal p is at location L' can be seen as a contextual predicate, which we denote $\lambda_L(p)$. As we have discussed earlier, a send- or receive-constrained channel can be used to authenticate contextual properties of the principal who uses the channel. If we can find a receive-constrained channel or a send-constrained channel on the predicate $\lambda_L(p)$, we can design location authentication protocols by requiring the principal in question to communicate over the constrained channel.

Note that location authentication is different from location identification, where the aim is to determine a principal's location, e.g., using a GPS system. In a location authentication problem, the location of the principal in question is asserted; the task is to find out whether the assertion is true.

In this section, we shall introduce several location authentication protocols based on constrained communication channels. As we stated above, there are many short-range communication technologies that can be used to implement constrained communication channels for location authentication, with various degrees of precision. For example, a Bluetooth radio or an IR transceiver can be used as either a receive- or a send-constrained channel.

The basic idea of our approach to authenticating a principal's location is to employ a challenge-response protocol. The authenticator can choose a nonce and either ask the principal in question to send it back to the authenticator over a send-constrained channel; or send the nonce to the principal in question over a receive-constrained channel and check if the principal has received it.

If the authenticator has direct access to a physically constrained (e.g. range-bounded) channel, that is, if the authenticator is located inside location *L*, it is trivial to implement location authentication. For example, the authenticator can send a nonce using a Bluetooth transceiver located at *L* and check if the principal receives it. If the principal is actually within the range of the Bluetooth transceiver, he/she should be able to receive the data from the Bluetooth transceiver. Here, the Bluetooth radio link is used as a receive-constrained channel.

What we are interested in is the more general case where the authenticator does not have direct access to a physically constrained communication channel at location *L*, i.e. the authenticator is remote from location *L*. In such a case, we need to use a trusted channel proxy to connect the authenticator with the constrained channel or to turn a local constrained channel into a remote constrained channel. We outlined such protocols in Section 3.2. In the following, we give more detailed protocols for this general case.

Before we describe the protocols, we should clarify one assumption we make about our constrained channels. That is 'whoever uses a physically constrained channel must be physically located within the transmission range of that channel'. It is conceivable that our protocols can be defeated by an attacker who sends an agent to the place of the constrained channel and uses the agent to relay the communication between the attacker and the constrained channel. Such an attack works unless the underlying communication system allows sufficiently accurate measurement of the response time to find out if a message has traveled 'extra' miles. This is not generally feasible on the Internet.

But there are many cases where it is either difficult or not worthwhile to send an agent or put a relaying device at the place of interest. For example, to use a relaying device means that the attacker has to put it there beforehand but the whereabouts of 'there' is often unforeseeable. In these cases, it is safe to make the assumption that whoever communicates with the constrained channel at a certain location is physically there. In this sense, we are providing reasonable solutions for some practical problems. Finally, we want to point out that it does not mean that the model we presented earlier is flawed. It only means we need some assumptions about the physical world to get the desired constrained channels.

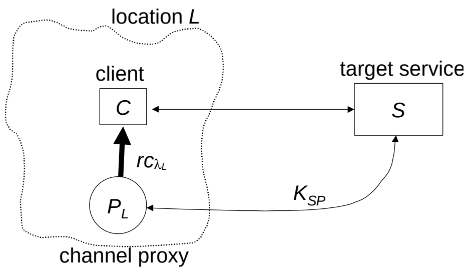


Figure 4. System model for telephone protocol

4.1 The telephone protocol

This protocol uses a receive-constrained channel for location authentication. The principals in our telephone protocol are the following (see Figure 4):

- A server *S*
- A channel proxy P_L
- A client *C*

Server *S* wants to verify that client *C* is at a certain location *L*—that is, that $\lambda_L(C)$—before providing services. Channel proxy P_L is a process (device) that connects securely to the send-end of a channel rc_{λ_L} at location *L* that is receive-constrained on λ_L. Channel proxy P_L shares a secret key K_{SP} with *S* and is trusted by *S* to behave correctly. Channel rc_{λ_L} could be a network whose physical reach is limited to *L*, such as one or more wireless LANs or infrared beacons. The constrained channel is such that data may be received from the channel only if the receiver is physically inside location *L*.

The goal of the telephone protocol is for *S* to verify that $\lambda_L(sender(R))$, where *R* is a service request and *L* is the asserted location of $C = sender(R)$. The protocol proceeds as follows (we use the standard notation $\{M\}K$ for the encryption of *M* with key *K*):

(1) $C \rightarrow S$: C, R, L

(2) $S \rightarrow P_L$: $\{C, N\}K, \{K\}K_{SP}$
/* *N* is a nonce and *K* is a randomly chosen session key */

(3) $P_L \rightarrow C$: C, N
/* broadcast over receive-constrained channel rc_{λ_L} */

(4) $C \rightarrow S$: C, R, N
/* S checks whether the received N is equal to the N sent to P_L */

In Step (2), the message sent is an encryption of $\{C, N\}$ using key K_{SP}. Session key K is used to defend against known-plaintext attacks on K_{SP}. Since only P_L knows K_{SP}, only $\mathrm{P_L}$ can receive $\{C, N\}$. Hence, the protocol effectively set up a receive-constrained channel $rc(P_L)$ on the predicate 'receiver is P_L' between S and P_L. (see Section 3.2). Moreover, Step (2) and (3) together can be viewed as a single step in which S sends $\{C, N\}$ to C over the aggregated receive-constrained channel $rc(P_L).P_L.\ rc_{\lambda_L}$.

We call this the 'telephone protocol' because a telephone could, in principle, serve to implement rc_{λ_L} (it rings and C must answer it to prove his presence in L).

Our protocol does not address issues of identity authentication or anonymity. Those are important for nomadic computing systems but they are orthogonal to our current purposes.

4.2 The private telephone protocol

The above telephone protocol does not protect client C's privacy since $\{C, N\}$ sent in Step (3) may be broadcast to all receivers located in L, e.g., over a Bluetooth link. An eavesdropper could pick up $\{C, N\}$ and learn about C's identity. Suppose we have the same principals and the same set-up as in the telephone protocol, except that, for privacy reasons, C and S share an encrypted channel. We can protect client C's identity by using the following private telephone protocol:

(1) $C \rightarrow S$: C, R, L, N_1
/* N_1 is a nonce generated by C */

(2) $S \rightarrow P_L$: $\{N_1, N_2\}K, \{K\}K_{SP}$
/* K is a randomly chosen session key and N_2 is a nonce generated by S */

(3) $P_L \rightarrow C$: N_1, N_2
/* broadcast over receive-constrained channel rc_{λ_L} */

(4) $C \rightarrow S$: C, R, N_2
/* S checks if the received N_2 is equal to the N_2 sent to $\mathrm{P_L}$ */

This protocol is similar to the telephone protocol. The difference is that a random identifier (N_1) is used to protect the identity of client C.

4.3 The offline protocol

The above two protocols assume that S and P_L can communicate in real-time. In cases where this is not true, we can use an 'offline' protocol to achieve our goal.

We assume the same principals and the same set-up as in the telephone protocol except that S and P_L cannot communicate directly (although they do share a secret key K_{SP}). The offline protocol follows:

(1) $C \rightarrow S$: C, R, L

(2) $S \rightarrow C$: $\{N_1\}K_{SP}, \{N_1 \otimes N_2\}K_{SP}$
/* N_1, N_2 are nonces; '⊗' denotes the exclusive-or operation */

(3) $C \rightarrow P_L$: $\{N_1\}K_{SP}, \{N_1 \otimes \mathrm{N}_2\}K_{SP}$

(4) $P_L \rightarrow C$: $\{N_2\}K_{SP}$
/* sent over receive-constrained channel rc_{λ_L} */

(5) $C \rightarrow S$: $C, R, \{N_2\}K_{SP}$
/* S checks whether the received N_2 is equal to the N_2 sent in Step (2) */

Since P_L is the only party (except S) that can compute $\{N_2\}K_{SP}$ from $\{\{N_1\}K_{SP}, \{N_1 \otimes N_2\}K_{SP}\}$, the fact that C can show the correct $\{N_2\}K_{SP}$ means that C is able to receive it from the receive-constrained channel rc_{λ_L}. Therefore, it verifies that C is at location L. Note that no nonce is sent in the clear, thus we avoid known-plaintext attacks.

In Step (4), C receives $\{N_2\}K_{SP}$ over a receive-constrained channel rc_{λ_L}. However, if we view Steps (3) and (4) combined as the precondition for Step (5), the protocol is effectively turning a receive-constrained channel from P_L to C into an aggregated send-constrained channel from C to S—as we outlined in Section 3.2.

This protocol can be implemented transparently for a standard web browser. Suppose a client's browser C contacts a web server S for services in Step (1). The response coming back from S in Step (2) includes an HTTP redirection pointing to a local channel proxy P_L. Similarly, the response from P_L in Step (4) includes another HTTP redirection pointing back to S. Hence, the browser can be transparently directed to contact P_L for authenticating its location.

4.4 The self-verification protocol

Interestingly, a constrained channel can be used to verify a principal's own location. In the case of a receive-constrained channel, a principal can send a message to a receive-constrained channel rc_{λ_L} for location L and check if he can receive the same message from this channel. The principal has to know how to send a message to rc_{λ_L} but the self-verification protocol itself is trivial once the receive-constrained channel is available. The difficulty lies in how to construct rc_{λ_L} so that its guarantees are securely implemented.

Alternatively, a principal can send a message to a local send-constrained channel sc_{λ_L} at location L and check if the same message can be received from sc_{λ_L}. Again the self-verification protocol itself is trivial once the send-constrained channel is available.

A real-life example arises from using the fixed-line telephone system (again, assuming that that system were to be sufficiently secure). If a user knows the telephone number of the fixed phone located at a certain place, the user can check if they are in that place by calling that number using, for example, a mobile phone and seeing whether the phone at their current location rings. In this way, they are using the fixed-line telephone system as a receive-constrained channel.

Alternatively, the user can employ the fixed-line telephone system as a send-constrained channel by calling their mobile phone from the fixed phone at their current location. The user checks whether the caller-id shown on the mobile phone matches the phone number of the place whose verification is in question.

5 Conclusion

We have described a model of send- and receive-constrained channels for context authentication. We have shown how to construct simple examples of constrained channels on location predicates. For this purpose, we use physical communication channels that are subject to mechanical or signal propagation constraints.

We further showed how, by inserting channel proxies, we can 'lengthen' channels constrained on location predicates. Moreover, proxies enable us to broaden channel constraints to temporal and other contextual predicates. We showed how to construct send-constrained channels from receive- constrained channels and *vice versa*.

Finally, we gave practical protocols for location authentication, including one that protects clients' privacy. We outlined how components can use similar protocols to authenticate their own location.

Applicability

In Section 1, we listed six problems that exemplify the types of context authentication in which we are interested. The first four are examples of location authentication; the third and fourth involve self-verification of location (a computer inside a building, a kiosk near to a human carrying a short-range radio transceiver). We have shown how to achieve those types of authentication in Section 4.

The fifth asks us to authenticate that a principal is in a certain 'virtual location'. That problem falls under the techniques we have given for physical location. A server can place a nonce on the web page, of which the client (who must 'visit the URL') must demonstrate knowledge.

The final example illustrates a temporal predicate. We have shown how to use a channel proxy to achieve that.

Status

We are engaged in an implementation of a location authentication protocol for CoolTown places. Users with handheld devices running standard web browsers will be able to prove their presence in, for example, a coffee shop or bookshop, and thus obtain privileged services such as printing-on-the-go, or a discount.

In this paper, we have contributed a new abstraction—the send- or receive-constrained channel—to help us formulate a notion of context authentication that can be realised. We have given protocols for some simple cases involving location, based on assumptions about signal propagation. It remains to show the true practicality of implementing a location authentication system in a real situation, and extending the work to a broader notion of context beyond toy examples such as time, to complex notions of, for example, user-presence.

References

[1] John Barton & Tim Kindberg (2001). “The challenges and opportunities of integrating the physical world and networked systems”. HPL Technical report HPL-2001-18, available as http://www.champignon.net/TimKindberg/MobicomChallengeAsTR.pdf.

[2] Tim Kindberg & John Barton (2001). “A Web-Based Nomadic Computing System”. To appear, Computer Networks, Elsevier. Available as http://www.cooltown.hp.com/papers/nomadic/nomadic.htm.

[3] Tim Kindberg et al. (2000). “People, Places, Things: Web Presence for the Real World”. In proceedings WMCSA2000. Available as http://www.cooltown.hp.com/papers/webpres/-webpresence.htm.

[4] Debbie Caswell & Philippe Debaty (2000). “Creating web representations for places”. Proceedings Handheld and Ubiquitous Computing 2000, Springer, pp. 114-126. Available as http://www.cooltown.hp.com/papers/placeman/placesHUC2000.pdf.

[5] R. Want, A. Hopper, V. Falcao, and J. Gibbons (1992). “The active badge location system”. ACM Transactions on Information Systems, vol. 10, pp. 91-102.

[6] Dorothy E. Denning and Peter F. MacDoran (1996). “Location-Based Authentication: Grounding Cyberspace for Better Security”. In Computer Fraud & Security, February. Available as http://www.cosc.georgetown.edu/~denning/infosec/Grounding.txt.

[7] T. Dierks and C. Allen (1999). “Transport Layer Security”. RFC 2246. www.ietf.org.

[8] A. Harter, A. Hopper, P. Steggles, A. Ward, P. Webster. “The Anatomy of a Context-Aware Application”. *Proc. 5th Annual ACM/IEEE Int’l Conf. on Mobile Computing and Networking*, Seattle, Washington, USA, August 1999, pp. 59-68.

[9] Y. Desmedt. “Major security problems with the ‘unforgeable’ (Feige)-Fiat-Shamir proofs of identity and how to overcome them,” SecuriCom’88, SEDEP Paris, 1988, pp 15-17.

[10] S. Brands and D. Chaum, “Distance-Bounding Protocols,” Proc. EUROCRYPT '93, Lecture Notes in Computer Science, no. 765, Springer-Verlag, pp. 344-359.

A Method for Obtaining Digital Signatures and Public-Key Cryptosystems

R.L. Rivest, A. Shamir, and L. Adleman*

Abstract

An encryption method is presented with the novel property that publicly revealing an encryption key does not thereby reveal the corresponding decryption key. This has two important consequences:

1. Couriers or other secure means are not needed to transmit keys, since a message can be enciphered using an encryption key publicly revealed by the intended recipient. Only he can decipher the message, since only he knows the corresponding decryption key.
2. A message can be "signed" using a privately held decryption key. Anyone can verify this signature using the corresponding publicly revealed encryption key. Signatures cannot be forged, and a signer cannot later deny the validity of his signature. This has obvious applications in "electronic mail" and "electronic funds transfer" systems.

A message is encrypted by representing it as a number M, raising M to a publicly specified power e, and then taking the remainder when the result is divided by the publicly specified product, n, of two large secret prime numbers p and q. Decryption is similar; only a different, secret, power d is used, where $e \cdot d \equiv 1 \pmod{(p-1) \cdot (q-1)}$. The security of the system rests in part on the difficulty of factoring the published divisor, n.

Key Words and Phrases: digital signatures, public-key cryptosystems, privacy, authentication, security, factorization, prime number, electronic mail, message-passing, electronic funds transfer, cryptography.

CR Categories: 2.12, 3.15, 3.50, 3.81, 5.25

This research was supported by National Science Foundation grant MCS76-14294, and the Office of Naval Research grant number N00014-67-A-0204-0063.

Author's Address: Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 E-mail addresses: `rivest@theory.lcs.mit.edu`

I Introduction

The era of "electronic mail" [10] may soon be upon us; we must ensure that two important properties of the current "paper mail" system are preserved: (a) messages are *private*, and (b) messages can be *signed*. We demonstrate in this paper how to build these capabilities into an electronic mail system.

At the heart of our proposal is a new encryption method. This method provides an implementation of a "public-key cryptosystem," an elegant concept invented by Diffie and Hellman [1]. Their article motivated our research, since they presented the concept but not any practical implementation of such a system. Readers familiar with [1] may wish to skip directly to Section V for a description of our method.

II Public-Key Cryptosystems

In a "public key cryptosystem" each user places in a public file an encryption procedure E. That is, the public file is a directory giving the encryption procedure of each user. The user keeps secret the details of his corresponding decryption procedure D. These procedures have the following four properties:

(a) Deciphering the enciphered form of a message M yields M. Formally,

$$D(E(M) = M. \tag{1}$$

(b) Both E and D are easy to compute.

(c) By publicly revealing E the user does not reveal an easy way to compute D. This means that in practice only he can decrypt messages encrypted with E, or compute D efficiently.

(d) If a message M is first deciphered and then enciphered, M is the result. Formally,

$$E(D(M) = M. \tag{2}$$

An encryption (or decryption) procedure typically consists of a *general method* and an *encryption key*. The general method, under control of the key, enciphers a message M to obtain the enciphered form of the message, called the *ciphertext C*. Everyone can use the same general method; the security of a given procedure will rest on the security of the key. Revealing an encryption algorithm then means revealing the key.

When the user reveals E he reveals a very *inefficient* method of computing $D(C)$: testing all possible messages M until one such that $E(M) = C$ is found. If property (c) is satisfied the number of such messages to test will be so large that this approach is impractical.

A function E satisfying (a)-(c) is a "trap-door one-way function;" if it also satisfies (d) it is a "trap-door one-way permutation." Diffie and Hellman [1] introduced the concept of trap-door one-way functions but did not present any examples. These functions are called "one-way" because they are easy to compute in one direction but (apparently) very difficult to compute in the other direction. They are called "trap-door" functions since the inverse functions are in fact easy to compute once certain private "trap-door" information is known. A trap-door one-way function which also satisfies (d) must be a permutation: every message is the cipertext for some other message and every ciphertext is itself a permissible message. (The mapping is "one-to-one" and "onto"). Property (d) is needed only to implement "signatures."

The reader is encouraged to read Diffie and Hellman's excellent article [1] for further background, for elaboration of the concept of a public-key cryptosystem, and for a discussion of other problems in the area of cryptography. The ways in which a public-key cryptosystem can ensure privacy and enable "signatures" (described in Sections III and IV below) are also due to Diffie and Hellman.

For our scenarios we suppose that A and B (also known as Alice and Bob) are two users of a public-key cryptosystem. We will distinguish their encryption and decryption procedures with subscripts: E_A, D_A, E_B, D_B.

III Privacy

Encryption is the standard means of rendering a communication private. The sender enciphers each message before transmitting it to the receiver. The receiver (but no unauthorized person) knows the appropriate deciphering function to apply to the received message to obtain the original message. An eavesdropper who hears the transmitted message hears only "garbage" (the ciphertext) which makes no sense to him since he does not know how to decrypt it.

The large volume of personal and sensitive information currently held in computerized data banks and transmitted over telephone lines makes encryption increasingly important. In recognition of the fact that efficient, high-quality encryption techniques are very much needed but are in short supply, the National Bureau of Standards has recently adopted a "Data Encryption Standard" [13, 14], developed at IBM. The new standard does not have property (c), needed to implement a public-key cryptosystem.

All classical encryption methods (including the NBS standard) suffer from the "key distribution problem." The problem is that before a private communication can begin, *another* private transaction is necessary to distribute corresponding encryption and decryption keys to the sender and receiver, respectively. Typically a private courier is used to carry a key from the sender to the receiver. Such a practice is not feasible if an electronic mail system is to be rapid and inexpensive. A public-key cryptosystem needs no private couriers; the keys can be distributed over the insecure communications channel.

How can Bob send a private message M to Alice in a public-key cryptosystem? First, he retrieves E_A from the public file. Then he sends her the enciphered message $E_A(M)$. Alice deciphers the message by computing $D_A(E_A(M)) = M$. By property (c) of the public-key cryptosystem only she can decipher $E_A(M)$. She can encipher a private response with E_B, also available in the public file.

Observe that no private transactions between Alice and Bob are needed to establish private communication. The only "setup" required is that each user who wishes to receive private communications must place his enciphering algorithm in the public file.

Two users can also establish private communication over an insecure communications channel without consulting a public file. Each user sends his encryption key to the other. Afterwards all messages are enciphered with the encryption key of the recipient, as in the public-key system. An intruder listening in on the channel cannot decipher any messages, since it is not possible to derive the decryption keys from the encryption keys. (We assume that the intruder cannot modify or insert messages into the channel.) Ralph Merkle has developed another solution [5] to this problem.

A public-key cryptosystem can be used to "bootstrap" into a standard encryption scheme such as the NBS method. Once secure communications have been established, the first message transmitted can be a key to use in the NBS scheme to encode all following messages. This may be desirable if encryption with our method is slower than with the standard scheme. (The NBS scheme is probably somewhat faster if special-purpose hardware encryption devices are used; our scheme may be faster on a general-purpose computer since multiprecision arithmetic operations are simpler to implement than complicated bit manipulations.)

IV Signatures

If electronic mail systems are to replace the existing paper mail system for business transactions, "signing" an electronic message must be possible. The recipient of a signed message has proof that the message originated from the sender. This quality is stronger than mere authentication (where the recipient can verify that the message came from the sender); the recipient can convince a "judge" that the signer sent the message. To do so, he must convince the judge that he did not forge the signed message himself! In an authentication problem the recipient does not worry about this possibility, since he only wants to satisfy *himself* that the message came from the sender.

An electronic signature must be *message*-dependent, as well as *signer*-dependent. Otherwise the recipient could modify the message before showing the message-signature pair to a judge. Or he could attach the signature to any message whatsoever, since it is impossible to detect electronic "cutting and pasting."

To implement signatures the public-key cryptosystem must be implemented with trap-door one-way permutations (i.e. have property (d)), since the decryption algorithm will be applied to unenciphered messages.

How can user Bob send Alice a "signed" message M in a public-key cryptosystem? He first computes his "signature" S for the message M using D_B:

$$S = D_B(M) \ .$$

(Deciphering an unenciphered message "makes sense" by property (d) of a public-key cryptosystem: each message is the ciphertext for some other message.) He then encrypts S using E_A (for privacy), and sends the result $E_A(S)$ to Alice. He need not send M as well; it can be computed from S.

Alice first decrypts the ciphertext with D_A to obtain S. She knows who is the presumed sender of the signature (in this case, Bob); this can be given if necessary in plain text attached to S. She then extracts the message with the encryption procedure of the sender, in this case E_B (available on the public file):

$$M = E_B(S) \ .$$

She now possesses a message-signature pair (M, S) with properties similar to those of a signed paper document.

Bob cannot later deny having sent Alice this message, since no one else could have created $S = D_B(M)$. Alice can convince a "judge" that $E_B(S) = M$, so she has proof that Bob signed the document.

Clearly Alice cannot modify M to a different version M', since then she would have to create the corresponding signature $S' = D_B(M')$ as well.

Therefore Alice has received a message "signed" by Bob, which she can "prove" that he sent, but which she cannot modify. (Nor can she forge his signature for any other message.)

An electronic checking system could be based on a signature system such as the above. It is easy to imagine an encryption device in your home terminal allowing you to sign checks that get sent by electronic mail to the payee. It would only be necessary to include a unique check number in each check so that even if the payee copies the check the bank will only honor the first version it sees.

Another possibility arises if encryption devices can be made fast enough: it will be possible to have a telephone conversation in which every word spoken is signed by the encryption device before transmission.

When encryption is used for signatures as above, it is important that the encryption device not be "wired in" between the terminal (or computer) and the communications channel, since a message may have to be successively enciphered with several keys. It is perhaps more natural to view the encryption device as a "hardware subroutine" that can be executed as needed.

We have assumed above that each user can always access the public file reliably. In a "computer network" this might be difficult; an "intruder" might forge messages purporting to be from the public file. The user would like to be sure that he actually obtains the encryption procedure of his desired correspondent and not, say, the encryption procedure of the intruder. This danger disappears if the public file "signs" each message it sends to a user. The user can check the signature with the public file's encryption algorithm E_{PF}. The problem of "looking up" E_{PF} itself in the public file is avoided by giving each user a description of E_{PF} when he first shows up (in person) to join the public-key cryptosystem and to deposit his public encryption procedure. He then stores this description rather than ever looking it up again. The need for a courier between every pair of users has thus been replaced by the requirement for a single secure meeting between each user and the public file manager when the user joins the system. Another solution is to give each user, when he signs up, a book (like a telephone directory) containing all the encryption keys of users in the system.

V Our Encryption and Decryption Methods

To encrypt a message M with our method, using a public encryption key (e, n), proceed as follows. (Here e and n are a pair of positive integers.)

First, represent the message as an integer between 0 and $n - 1$. (Break a long message into a series of blocks, and represent each block as such an integer.) Use any standard representation. The purpose here is not to encrypt the message but only to get it into the numeric form necessary for encryption.

Then, encrypt the message by raising it to the eth power modulo n. That is, the result (the ciphertext C) is the remainder when M^e is divided by n.

To decrypt the ciphertext, raise it to another power d, again modulo n. The encryption and decryption algorithms E and D are thus:

$$\begin{aligned} C &\equiv E(M) \equiv M^e \pmod{n}, \text{ for a message } M \ . \\ & D(C) \equiv C^d \pmod{n}, \text{ for a ciphertext } C \ . \end{aligned}$$

Note that encryption does not increase the size of a message; both the message and the ciphertext are integers in the range 0 to $n - 1$.

The *encryption key* is thus the pair of positive integers (e, n). Similarly, the *decryption* key is the pair of positive integers (d, n). Each user makes his encryption key public, and keeps the corresponding decryption key private. (These integers should properly be subscripted as in n_A, e_A, and d_A, since each user has his own set. However, we will only consider a typical set, and will omit the subscripts.)

How should you choose your encryption and decryption keys, if you want to use our method?

You first compute n as the product of two primes p and q:

$$n = p \cdot q \ .$$

These primes are very large, "random" primes. Although you will make n public, the factors p and q will be effectively hidden from everyone else due to the enormous difficulty of factoring n. This also hides the way d can be derived from e.

You then pick the integer d to be a large, random integer which is relatively prime to $(p - 1) \cdot (q - 1)$. That is, check that d satisfies:

$$\gcd(d, (p - 1) \cdot (q - 1)) = 1$$

("gcd" means "greatest common divisor").

The integer e is finally computed from p, q, and d to be the "multiplicative inverse" of d, modulo $(p-1)\cdot(q-1)$. Thus we have

$$e \cdot d \equiv 1 \pmod{(p-1)\cdot(q-1)}.$$

We prove in the next section that this guarantees that (1) and (2) hold, i.e. that E and D are inverse permutations. Section VII shows how each of the above operations can be done efficiently.

The aforementioned method should not be confused with the "exponentiation" technique presented by Diffie and Hellman [1] to solve the key distribution problem. Their technique permits two users to determine a key in common to be used in a normal cryptographic system. It is not based on a trap-door one-way permutation. Pohlig and Hellman [8] study a scheme related to ours, where exponentiation is done modulo a prime number.

VI The Underlying Mathematics

We demonstrate the correctness of the deciphering algorithm using an identity due to Euler and Fermat [7]: for any integer (message) M which is relatively prime to n,

$$M^{\phi(n)} \equiv 1 \pmod{n} \ . \tag{3}$$

Here $\phi(n)$ is the Euler totient function giving number of positive integers less than n which are relatively prime to n. For prime numbers p,

$$\phi(p) = p - 1 \ .$$

In our case, we have by elementary properties of the totient function [7]:

$$\begin{aligned} \phi(n) &= \phi(p)\cdot\phi(q) \\ &= (p-1)\cdot(q-1) \\ &= n-(p+q)+1 \ . \end{aligned} \tag{4}$$

Since d is relatively prime to $\phi(n)$, it has a multiplicative inverse e in the ring of integers modulo $\phi(n)$:

$$e \cdot d \equiv 1 \pmod{\phi(n)}. \tag{5}$$

We now prove that equations (1) and (2) hold (that is, that deciphering works correctly if e and d are chosen as above). Now

$$\begin{aligned} D(E(M)) &\equiv (E(M))^d \equiv (M^e)^d \pmod{n} = M^{e\cdot d} \pmod{n} \\ E(D(M)) &\equiv (D(M))^e \equiv (M^d)^e \pmod{n} = M^{e\cdot d} \pmod{n} \end{aligned}$$

and

$$M^{e\cdot d} \equiv M^{k\cdot\phi(n)+1} \pmod{n} \text{ (for some integer } k).$$

From (3) we see that for all M such that p does not divide M

$$M^{p-1} \equiv 1 \pmod{p}$$

and since $(p-1)$ divides $\phi(n)$

$$M^{k\cdot\phi(n)+1} \equiv M \pmod{p}.$$

This is trivially true when $M \equiv 0 \pmod{p}$, so that this equality actually holds for *all* M. Arguing similarly for q yields

$$M^{k\cdot\phi(n)+1} \equiv M \pmod{q}\ .$$

Together these last two equations imply that for all M,

$$M^{e\cdot d} \equiv M^{k\cdot\phi(n)+1} \equiv M \pmod{n}.$$

This implies (1) and (2) for all $M, 0 \leq M < n$. Therefore E and D are inverse permutations. (We thank Rich Schroeppel for suggesting the above improved version of the authors' previous proof.)

VII Algorithms

To show that our method is practical, we describe an efficient algorithm for each required operation.

A How to Encrypt and Decrypt Efficiently

Computing $M^e \pmod{n}$ requires at most $2\cdot\log_2(e)$ multiplications and $2\cdot\log_2(e)$ divisions using the following procedure (decryption can be performed similarly using d instead of e):

```
Step 1. Let $e_k e_{k-1}...e_1 e_0$ be the binary representation of $e$.
Step 2. Set the variable $C$ to 1.
Step 3. Repeat steps 3a and 3b for $i = k, k-1, \ldots, 0$:
    Step 3a. Set $C$ to the remainder of $C^2$ when divided by $n$.
    Step 3b. If $e_i = 1$, then set $C$ to the remainder of $C \cdot M$ when divided by $n$.
Step 4. Halt. Now $C$ is the encrypted form of $M$.
```

This procedure is called "exponentiation by repeated squaring and multiplication." This procedure is half as good as the best; more efficient procedures are known. Knuth [3] studies this problem in detail.

The fact that the enciphering and deciphering are identical leads to a simple implementation. (The whole operation can be implemented on a few special-purpose integrated circuit chips.)

A high-speed computer can encrypt a 200-digit message M in a few seconds; special-purpose hardware would be much faster. The encryption time per block increases no faster than the cube of the number of digits in n.

B How to Find Large Prime Numbers

Each user must (privately) choose two large random numbers p and q to create his own encryption and decryption keys. These numbers must be large so that it is not computationally feasible for anyone to factor $n = p \cdot q$. (Remember that n, but not p or q, will be in the public file.) We recommend using 100-digit (decimal) prime numbers p and q, so that n has 200 digits.

To find a 100-digit "random" prime number, generate (odd) 100-digit random numbers until a prime number is found. By the prime number theorem [7], about $(\ln 10^{100})/2 = 115$ numbers will be tested before a prime is found.

To test a large number b for primality we recommend the elegant "probabilistic" algorithm due to Solovay and Strassen [12]. It picks a random number a from a uniform distribution on $\{1, \ldots, b-1\}$, and tests whether

$$\gcd(a, b) = 1 \text{ and } J(a, b) = a^{(b-1)/2} \pmod{b}, \tag{6}$$

where $J(a, b)$ is the Jacobi symbol [7]. If b is prime (6) is always true. If b is composite (6) will be false with probability at least 1/2. If (6) holds for 100 randomly chosen values of a then b is almost certainly prime; there is a (negligible) chance of one in 2^{100} that b is composite. Even if a composite were accidentally used in our system, the receiver would probably detect this by noticing that decryption didn't work correctly. When b is odd, $a \leq b$, and $\gcd(a, b) = 1$, the Jacobi symbol $J(a, b)$ has a value in $\{-1, 1\}$ and can be efficiently computed by the program:

```
$J(a,b) =$ if $a = 1$ then 1 else
           if $a$ is even then $J(a/2, b) \cdot (-1)^{(b^2-1)/8}$
           else $J(b \pmod{a}, a) \cdot (-1)^{(a-1)\cdot(b-1)/4}$
```

(The computations of $J(a, b)$ and $\gcd(a, b)$ can be nicely combined, too.) Note that this algorithm does *not* test a number for primality by trying to factor it. Other efficient procedures for testing a large number for primality are given in [6,9,11].

To gain additional protection against sophisticated factoring algorithms, p and q should differ in length by a few digits, both $(p-1)$ and $(q-1)$ should contain large prime factors, and $\gcd(p-1, q-1)$ should be small. The latter condition is easily checked.

To find a prime number p such that $(p-1)$ has a large prime factor, generate a large random prime number u, then let p be the first prime in the sequence $i \cdot u + 1$, for $i = 2, 4, 6, \ldots$. (This shouldn't take too long.) Additional security is provided by ensuring that $(u-1)$ also has a large prime factor.

A high-speed computer can determine in several seconds whether a 100-digit number is prime, and can find the first prime after a given point in a minute or two.

Another approach to finding large prime numbers is to take a number of known factorization, add one to it, and test the result for primality. If a prime p is found it is possible to *prove* that it really is prime by using the factorization of $p-1$. We omit a discussion of this since the probabilistic method is adequate.

C How to Choose d

It is very easy to choose a number d which is relatively prime to $\phi(n)$. For example, any prime number greater than $\max(p,q)$ will do. It is important that d should be chosen from a large enough set so that a cryptanalyst cannot find it by direct search.

D How to Compute e from d and $\phi(n)$

To compute e, use the following variation of Euclid's algorithm for computing the greatest common divisor of $\phi(n)$ and d. (See exercise 4.5.2.15 in [3].) Calculate $\gcd(\phi(n), d)$ by computing a series $x_0, x_1, x_2, \ldots$, where $x_0 \equiv \phi(n), x_1 = d$, and $x_{i+1} \equiv x_{i-1} \pmod{x_i}$, until an x_k equal to 0 is found. Then $\gcd(x_0, x_1) = x_{k-1}$. Compute for each x_i numbers a_i and b_i such that $x_i = a_i \cdot x_0 + b_i \cdot x_1$. If $x_{k-1} = 1$ then b_{k-1} is the multiplicative inverse of $x_1 \pmod{x_0}$. Since k will be less than $2\log_2(n)$, this computation is very rapid.

If e turns out to be less than $\log_2(n)$, start over by choosing another value of d. This guarantees that every encrypted message (except $M = 0$ or $M = 1$) undergoes some "wrap-around" (reduction modulo n) .

VIII A Small Example

Consider the case $p = 47, q = 59, n = p \cdot q = 47 \cdot 59 = 2773$, and $d = 157$. Then $\phi(2773) = 46 \cdot 58 = 2668$, and e can be computed as follows:

$x_0 = 2668,\quad a_0 = 1,\quad b_0 = 0,$
$x_1 = 157,\quad a_1 = 0,\quad b_1 = 1,$
$x_2 = 156,\quad a_2 = 1,\quad b_2 = -16$ (since $2668 = 157 \cdot 16 + 156$) ,
$x_3 = 1,\quad a_3 = -1,\quad b_3 = 17$ (since $157 = 1 \cdot 156 + 1$) .

Therefore $e = 17$, the multiplicative inverse $\pmod{2668}$ of $d = 157$.

With $n = 2773$ we can encode two letters per block, substituting a two-digit number for each letter: blank = 00, A = 01, B = 02, ..., Z = 26. Thus the message

```
ITS ALL GREEK TO ME
```

(Julius Caesar, I, ii, 288, paraphrased) is encoded:

```
0920 1900 0112 1200 0718 0505 1100 2015 0013 0500
```

Since $e = 10001$ in binary, the first block ($M = 920$) is enciphered:

$$M^{17} = (((((1)^2 \cdot M)^2)^2)^2)^2 \cdot M = 948 \pmod{2773} .$$

The whole message is enciphered as:

`0948 2342 1084 1444 2663 2390 0778 0774 0219 1655 .`

The reader can check that deciphering works: $948^{157} \equiv 920 \pmod{2773}$, etc.

IX Security of the Method: Cryptanalytic Approaches

Since no techniques exist to *prove* that an encryption scheme is secure, the only test available is to see whether anyone can think of a way to break it. The NBS standard was "certified" this way; seventeen man-years at IBM were spent fruitlessly trying to break that scheme. Once a method has successfully resisted such a concerted attack it may for practical purposes be considered secure. (Actually there is some controversy concerning the security of the NBS method [2].)

We show in the next sections that all the obvious approaches for breaking our system are at least as difficult as factoring n. While factoring large numbers is not provably difficult, it is a well-known problem that has been worked on for the last three hundred years by many famous mathematicians. Fermat (1601?-1665) and Legendre (1752-1833) developed factoring algorithms; some of today's more efficient algorithms are based on the work of Legendre. As we shall see in the next section, however, no one has yet found an algorithm which can factor a 200-digit number in a reasonable amount of time. We conclude that our system has already been partially "certified" by these previous efforts to find efficient factoring algorithms.

In the following sections we consider ways a cryptanalyst might try to determine the secret decryption key from the publicly revealed encryption key. We do not consider ways of protecting the decryption key from theft; the usual physical security methods should suffice. (For example, the encryption device could be a separate device which could also be used to generate the encryption and decryption keys, such that the decryption key is never printed out (even for its owner) but only used to decrypt messages. The device could erase the decryption key if it was tampered with.)

A Factoring n

Factoring n would enable an enemy cryptanalyst to "break" our method. The factors of n enable him to compute $\phi(n)$ and thus d. Fortunately, factoring a number seems to be much more difficult than determining whether it is prime or composite.

A large number of factoring algorithms exist. Knuth [3, Section 4.5.4] gives an excellent presentation of many of them. Pollard [9] presents an algorithm which factors a number n in time $O(n^{1/4})$.

The fastest factoring algorithm known to the authors is due to Richard Schroeppel (unpublished); it can factor n in approximately

$$\exp\sqrt{ln(n) \cdot ln(ln(n))} \quad = \quad n^{\sqrt{\ln\ln(n)/\ln(n)}}$$

$$= \quad (\ln(n))^{\sqrt{\ln(n)/\ln(\ln(n))}}$$

steps (here ln denotes the natural logarithm function). Table 1 gives the number of operations needed to factor n with Schroeppel's method, and the time required if each operation uses one microsecond, for various lengths of the number n (in decimal digits).

Table 1

Digits	*Number of operations*	*Time*
50	1.4×10^{10}	3.9 hours
75	9.0×10^{12}	104 days
100	2.3×10^{15}	74 years
200	1.2×10^{23}	3.8×10^{9} years
300	1.5×10^{29}	4.9×10^{15} years
500	1.3×10^{39}	4.2×10^{25} years

We recommend that n be about 200 digits long. Longer or shorter lengths can be used depending on the relative importance of encryption speed and security in the application at hand. An 80-digit n provides moderate security against an attack using current technology; using 200 digits provides a margin of safety against future developments. This flexibility to choose a key-length (and thus a level of security) to suit a particular application is a feature not found in many of the previous encryption schemes (such as the NBS scheme).

B Computing $\phi(n)$ Without Factoring n

If a cryptanalyst could compute $\phi(n)$ then he could break the system by computing d as the multiplicative inverse of e modulo $\phi(n)$ (using the procedure of Section VII D).

We argue that this approach is no easier than factoring n since it enables the cryptanalyst to easily factor n using $\phi(n)$. This approach to factoring n has not turned out to be practical.

How can n be factored using $\phi(n)$? First, $(p+q)$ is obtained from n and $\phi(n) = n - (p+q) + 1$. Then $(p-q)$ is the square root of $(p+q)^2 - 4n$. Finally, q is half the difference of $(p+q)$ and $(p-q)$.

Therefore breaking our system by computing $\phi(n)$ is no easier than breaking our system by factoring n. (This is why n must be composite; $\phi(n)$ is trivial to compute if n is prime.)

C Determining d Without Factoring n or Computing $\phi(n)$.

Of course, d should be chosen from a large enough set so that a direct search for it is unfeasible.

We argue that computing d is no easier for a cryptanalyst than factoring n, since once d is known n could be factored easily. This approach to factoring has also not turned out to be fruitful.

A knowledge of d enables n to be factored as follows. Once a cryptanalyst knows d he can calculate $e \cdot d - 1$, which is a multiple of $\phi(n)$. Miller [6] has shown that n can be factored using any multiple of $\phi(n)$. Therefore if n is large a cryptanalyst should not be able to determine d any easier than he can factor n.

A cryptanalyst may hope to find a d' which is equivalent to the d secretly held by a user of the public-key cryptosystem. If such values d' were common then a brute-force search could break the system. However, all such d' differ by the least common multiple of $(p-1)$ and $(q-1)$, and finding one enables n to be factored. (In (3) and (5), $\phi(n)$ can be replaced by $\mathrm{lcm}(p-1, q-1)$.) Finding any such d' is therefore as difficult as factoring n.

D Computing D in Some Other Way

Although this problem of "computing e-th roots modulo n without factoring n" is not a well-known difficult problem like factoring, we feel reasonably confident that it is computationally intractable. It may be possible to prove that any general method of breaking our scheme yields an efficient factoring algorithm. This would establish that any way of breaking our scheme must be as difficult as factoring. We have not been able to prove this conjecture, however.

Our method should be certified by having the above conjecture of intractability withstand a concerted attempt to disprove it. The reader is challenged to find a way to "break" our method.

X Avoiding "Reblocking" When Encrypting A Signed Message

A signed message may have to be "reblocked" for encryption since the signature n may be larger than the encryption n (every user has his own n). This can be avoided as follows. A threshold value h is chosen (say $h = 10^{199}$) for the public-key cryptosystem. Every user maintains two public (e, n) pairs, one for enciphering and one for signature-verification, where every signature n is less than h, and every enciphering n is greater than h. Reblocking to encipher a signed message is then unnecessary; the message is blocked according to the transmitter's signature n.

Another solution uses a technique given in [4]. Each user has a single (e, n) pair where n is between h and $2h$, where h is a threshold as above. A message is encoded as a number less than h and enciphered as before, except that if the ciphertext is greater than h, it is repeatedly re-enciphered until it is less than h. Similarly for decryption the ciphertext is repeatedly deciphered to obtain a value less than h. If n is near h re-enciphering will be infrequent. (Infinite looping is not possible, since at worst a message is enciphered as itself.)

XI Conclusions

We have proposed a method for implementing a public-key cryptosystem whose security rests in part on the difficulty of factoring large numbers. If the security of our method proves to be adequate, it permits secure communications to be established without the use of couriers to carry keys, and it also permits one to "sign" digitized documents.

The security of this system needs to be examined in more detail. In particular, the difficulty of factoring large numbers should be examined very closely. The reader is urged to find a way to "break" the system. Once the method has withstood all attacks for a sufficient length of time it may be used with a reasonable amount of confidence.

Our encryption function is the only candidate for a "trap-door one-way permutation" known to the authors. It might be desirable to find other examples, to provide alternative implementations should the security of our system turn out someday to be inadequate. There are surely also many new applications to be discovered for these functions.

Acknowledgments. We thank Martin Hellman, Richard Schroeppel, Abraham Lempel, and Roger Needham for helpful discussions, and Wendy Glasser for her assistance in preparing the initial manuscript. Xerox PARC provided support and some marvelous text-editing facilities for preparing the final manuscript.

Received April 4, 1977; revised September 1, 1977.

References

1. Diffie, W., and Hellman, M. New directions in cryptography. *IEEE Trans. Inform. Theory IT-22*, (Nov. 1976), 644-654.

2. Diffie, W., and Hellman, M. Exhaustive cryptanalysis of the NBS data encryption standard. *Computer* 10 (June 1977), 74-84.

3. Knuth, D. E. *The Art of Computer Programming, Vol 2: Seminumerical Algorithms.* Addison-Wesley, Reading, Mass., 1969.

4. Levine, J., and Brawley, J.V. Some cryptographic applications of permutation polynomials. *Cryptologia* 1 (Jan. 1977), 76-92.

5. Merkle, R. Secure communications over an insecure channel. Submitted to *Comm. ACM.*

6. Miller, G.L. Riemann's hypothesis and tests for primality. Proc. Seventh Annual ACM Symp. on the Theory of Comptng. Albuquerque, New Mex., May 1975, pp. 234-239; extended vers. available as Res. Rep. CS-75-27, Dept. of Comptr. Sci., U. of Waterloo, Waterloo, Ont., Canada, Oct. 1975.

7. Niven, I., and Zuckerman, H.S. *An Introduction to the Theory of Numbers.* Wiley, New York, 1972.

8. Pohlig, S.C., and Hellman, M.E. An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. To appear in IEEE Trans. Inform. Theory, 1978.

9. Pollard, J.M. Theorems on factorization and primality testing. *Proc. Camb. Phil. Soc. 76* (1974), 521-528.

10. Potter, R.J., Electronic mail. *Science 195*, 4283 (March 1977), 1160-1164.

11. Rabin, M.O., Probabilistic algorithms. In *Algorithms and Complexity*, J. F. Traub, Ed., Academic Press, New York, 1976, pp. 21-40.

12. Solovay, R., and Strassen, V. A Fast Monte-Carlo test for primality. *SIAM J. Comptng.* (March 1977), 84-85.

13. *Federal Register, Vol. 40*, No. 52, March 17, 1975.

14. *Federal Register, Vol. 40*, No. 149, August 1, 1975.