



AN ANALYSIS OF THE COMPUTER SECURITY RAMIFICATIONS OF
WEAKENED ASYMMETRIC CRYPTOGRAPHIC ALGORITHMS

GRADUATE RESEARCH PROJECT

Eric R. Bixby, Major, USAF

AFIT/ICW/ENG/12-02

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DISTRIBUTION STATEMENT A: APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION UNLIMITED.

The views expressed in this report are those of the author and do not reflect the official policy or position of the United States Air Force, the Department of Defense, or the United States Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

AFIT/ICW/ENG/12-02

AN ANALYSIS OF THE COMPUTER SECURITY RAMIFICATIONS OF
WEAKENED ASYMMETRIC CRYPTOGRAPHIC ALGORITHMS

GRADUATE RESEARCH PROJECT

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Cyber Warfare

Eric R. Bixby

Major, USAF

June 2012

DISTRIBUTION STATEMENT A: APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION UNLIMITED.

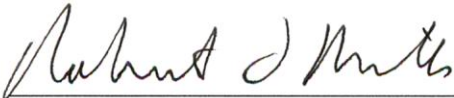
AFIT/ICW/ENG/12-02

AN ANALYSIS OF THE COMPUTER SECURITY RAMIFICATIONS OF
WEAKENED ASYMMETRIC CRYPTOGRAPHIC ALGORITHMS

Eric R. Bixby

Major, USAF

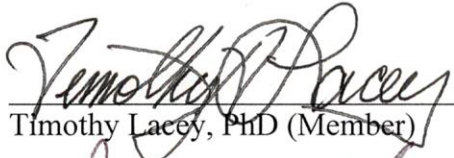
Approved:



Robert F. Mills, PhD (Chairman)

4 JUN 2012

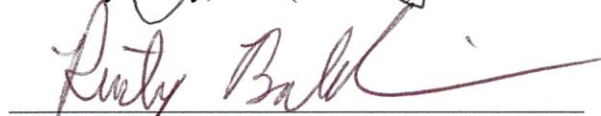
Date



Timothy Lacey, PhD (Member)

4 June 2012

Date



Rusty Baldwin, PhD (Member)

5 Jun 12

Date

Abstract

This paper explores the ramifications of what a proof that non-deterministic polynomial (NP) time algorithms could be solved in polynomial (P) time would mean for computer networking and the Internet as a whole. The $P = NP$ problem is a famous and unresolved mathematical question. If the P and NP classes of problems are really one and the same, there could be significant ramifications across numerous fields, including and especially asymmetric cryptography. Therefore, a great deal of effort in the computer science and mathematics fields has been devoted to this problem over four decades.

A significant subset of modern cryptographic systems relies on mathematical principles that make the assumption that $P \neq NP$. If $P = NP$, these cryptographic systems would be in imminent danger of being weakened or completely obviated. As a result, there are many who speculate that the consequences of a $P = NP$ proof would be the ultimate demise of the Internet. However, rarely are such claims substantiated with an analysis demonstrating how such effects would be caused. Therefore, this research attempts to determine the veracity of those claims through analysis of critical Internet protocols.

The paper includes an explanation of the $P = NP$ debate by describing what a P problem is and the contrasting it with an NP problem and then showing how they are related. It will then show how certain commonly-used cryptographic systems rely upon problems that fall within NP and describe how a $P = NP$ proof could affect the security of

those systems. Next it will examine critical components of computer networking and the Internet and determine how they rely upon potentially weakened cryptologic systems. That examination will include an analysis of how those dependencies impact network security (including data confidentiality, integrity and availability).

Table of Contents

	Page
Abstract	i
Table of Contents	1
List of Figures	3
List of Tables	4
I. Introduction	5
II. Background	13
2.1 P = NP Defined	13
2.2. Asymmetric Versus Symmetric Cryptography	20
III. Methodology	27
3.1 Protocol Selection	29
3.1.1 Bulk Encryption	29
3.1.2 Ethernet (IEEE 802.3)	30
3.1.3 Wi-Fi Protected Access 2 (WPA2)	31
3.1.4 Routing Protocols	34
3.1.5 Internet Protocol	34
3.1.6 Internet Protocol Security	34
3.1.7 Transmission Control Protocol and User Datagram Protocol	35
3.1.8 Secure Sockets Layer and Transport Layer Security	36
3.1.9 Secure Shell	38
3.1.10 Domain Name System	39
3.1.11 Electronic mail	40
3.1.12 Web and HyperText Transfer Protocol	41
IV. Analysis	43
4.1 Physical Layer Protocols	43
4.2 Data Link Layer Protocols	44
4.2.1 Bulk Encryption	45
4.2.3 Ethernet	46
4.2.4 Wi-Fi Protected Access 2	47
4.3 Network Layer Protocols	52
4.3.1 Routing Protocols	52
4.3.2 Internet Protocol	55
4.3.3 Internet Protocol Security	57
4.4 Transport Layer Protocols (TCP and UDP).....	60
4.5 Presentation Layer Protocols (SSL and TLS)	60
4.6 Application Layer Protocols	65
4.6.1 Secure Shell	66
4.6.2 Domain Name System	67
4.6.3 Electronic mail	69
4.6.4 Web and HyperText Transfer Protocol	71
4.7 Protocol Analysis Summary	74
4.8 Timeline Analysis	77

V. Conclusions and Recommendations.....	81
5.1 Conclusions.....	81
5.2 Recommendations for Future Research	84
Bibliography	86
Report Documentation Page.....	92

List of Figures

Figure	Page
1. Complexity class P and E relation	16
2. Complexity class P, NP, and E relation	18
3. Transport Layer Security (TLS) Message Sequence	62
4. TLS Authentication Using Certificate Authority.....	63

List of Tables

Table	Page
1. Growth in time to solve complexity functions as size of inputs (N) increases.....	15
2. Growth in number of symmetric keys required per user.....	22
3. List of Key Internet Protocols.....	29
4. WPA2 / 802.1X EAP Types	49
5. Summary of Key Internet Protocols and Their Dependency on Asymmetric Cryptography	76

AN ANALYSIS OF THE COMPUTER SECURITY RAMIFICATIONS OF WEAKENED ASYMMETRIC CRYPTOGRAPHIC ALGORITHMS

I. Introduction

This research does not seek to resolve the $P = NP$ question. Rather, it provides an analysis of the ramifications on modern telecommunications networking should a proof be developed showing that $P = NP$ (P stands for polynomial and NP for non-deterministic polynomial, defined later). It seeks to determine what core Internet infrastructures would be affected, to what degree, and on what kind of timeline should such an event come to pass. Further, it is intended for readers who wish to gain an understanding of the P vs. NP problem and exactly what the ramifications of a $P = NP$ proof would mean for the confidentiality, integrity, and availability of information and services on the Internet.

The initial idea for this paper was only loosely related to what was ultimately written here, and the research became something of a journey of discovery that led to this point. In reality, this paper began nearly seven years ago when I was first exposed to cryptography. In a lesson on asymmetric cryptography, an instructor off-handedly remarked that “Of course, all of this is based on unproven math and could be shown someday to be simple to reverse. This would mean the end of the Internet.”

I questioned that assertion at the time, believing that surely there was some mistake, because if this “fact” was correct, how could computer scientists, industry, and government accept such a risk? Or so my thinking went at the time. Needless to say, we didn’t solve the problem that day in class. But the niggling concern that modern technology was all built on the shaky foundation of an open question stuck with me. In

later academic pursuits, I was able to explore the problem a bit deeper and ultimately chose it as the topic for this research.

Modern asymmetric cryptography (cryptography that relies on public / private key pairs) is based on the premise that there are mathematical functions that are easy to perform in one direction and difficult to perform in the reverse. One classic example of these functions is the act of multiplying two large prime numbers to get a product (which is computationally easy) and the reverse act of factoring that product into the original two primes (which is computationally difficult). Other examples include modulus arithmetic, plotting points on an elliptic curve, and computing discrete logarithms. Using mathematical trickery, cryptographers are able to exchange keys over a non-secure channel and then initiate secure communications. This practice of secure key exchange over a non-secure channel enables important aspects of e-commerce and Internet security in general.

Surprisingly (at least to me), the assumed “one-way-ness” of the mathematical functions that underpin the secure exchange of information has not been proven. This means there is an unquantified, but non-zero possibility that someone could devise a method by which these functions could be efficiently (I will discuss what I mean by “efficiently” later) reversed by a computer. Admittedly, a lot of smart people have tried for nearly 50 years to develop a proof which lays to rest the P vs NP debate, so there is some merit in continuing to use these functions for encryption purposes. However, the creation of a method to reverse one-way functions could have immense ramifications on network security and e-commerce. So putting some thought behind preparing for the potential outcomes seemed prudent.

However, after conducting research, I do not believe a thorough analysis has ever been performed to determine the actual ramifications of the development of a method to efficiently reverse these functions. Indeed, apocalyptic predictions of the demise of the Internet were what attracted me to research the subject in the first place. Yet the predictions I found in books dealing with cryptanalysis, theoretical computer science, and simply in popular science were vague and unsubstantiated. For example:

“A proof that P equals NP would imply at once that the code-breaking problem for RSA could be solved in polynomial time, and thus would throw the entire Internet security system into doubt. Since we do not, at present, know of any way of ensuring the security of open Internet communications that does not depend on the effective impossibility of solving an NP problem, the current dependence of the Western economies on secure electronic communication over the Internet demonstrate just how high are the $P = NP$ stakes.” (Devlin 127)

“Many focus on the negative, that if $P = NP$ then public-key cryptography becomes impossible. True, but what we will gain from $P = NP$ will make the whole Internet look like a footnote in history.” (Fortnow, “Proceedings”)

“If these problems were shown to be solvable, that could undermine modern cryptography, which could paralyze electronic commerce and digital privacy because transactions would no longer be secure.” (Markhoff)

When I began this research, I assumed those types of predictions were true. My initial goal was to build a checklist to prepare the Department of Defense (DoD) to operate in a post- $P = NP$ world in which modern networking failed. Initially, I sought documents that discussed the ramifications of someone resolving the $P = NP$ problem because this is the open question that, if answered the wrong way, would spell disaster for asymmetric cryptography. I believed that I would find papers that explored key infrastructures and outlined the effects on those infrastructures should asymmetric encryption become defunct. I planned to then use those analyses to determine where the DoD relied upon those same infrastructures and plan alternatives and workarounds for

operating in a crypto-degraded state. Surely such analyses existed to justify the common proclamations that $P = NP$ would mean the end of the Internet. Yet I was unable to locate any such analysis documents or even references.

As a result, this research evolved into an analysis to prove or disprove unsubstantiated statements about how a $P = NP$ proof might affect Internet operations and security. I changed my approach and began analyzing critical infrastructures to gauge their reliance on asymmetric cryptography. I assumed that I would find dependencies that would indicate catastrophe should asymmetric cryptography become defunct. I was surprised when I found that many of the core Internet and DoD infrastructures rely on symmetric algorithms and are therefore relatively immune to this potential problem.

In the end, I did find some critical areas that would be profoundly affected by a $P = NP$ proof and a resulting efficient method for reversing one-way functions. These areas could directly and indirectly impact the DoD, but they would disrupt e-commerce and the financial industry far more. Encouragingly, my research indicated that even for these highly at-risk areas, there would likely be a warning followed by a preparation period during which corrections could be made before actual methods become available to exploit weakened asymmetric cryptography. Additionally, this analysis has identified potential methods that could be implemented to replace vulnerable asymmetric techniques.

I should take a brief moment to note that the premise this research is based on, namely, the $P = NP$ conjecture, is highly contested within the mathematics and computer science fields. Complexity theorists “generally believe $P \neq NP$ ” (Fortnow, “Status”). Simply stating, “Assume $P = NP$ for argument’s sake” is enough to have this

research dismissed by many theoretical computer scientists. Scott Aaronson, Associate Professor of Electrical Engineering and Computer Science at MIT, does an excellent job of summarizing the reasoning of those that believe that $P \neq NP$ in his article *Reasons to Believe* (3).

However, when you pick apart Dr. Aaronson's ten reasons, you see that there are really only four main arguments, and none of them conclude with certainty that P cannot equal NP . They merely give good reasons why it is probably the case that $P \neq NP$. The primary arguments against $P = NP$ put forth by Dr. Aaronson and the most stringent objectionists are 1) their gut feeling tells them that $P = NP$ is not how the universe works and 2) several decades of fruitlessly chewing on the problem with no definitive proof demonstrates that there can be no proof, 3) the reason $P \neq NP$ hasn't been proven is that it should be much harder to prove than $P = NP$, and 4) the separation between P and NP seems to apply in simplified models, therefore it seem reasonable that it would apply in more complex models.

In the first case, there have been numerous instances in which the universe has turned out to be much more complex and counter-intuitive than we thought. Discoveries such as Newtonian mechanics, Copernican astronomy, relativity, quantum theory, black holes, zero-point energy, dark matter and energy have all profoundly shaken humanity's views on how the universe works. Human beings are prone to "naïve empiricism" or the belief that things that agree with our experience constitute evidence that the world always works the way we've come to expect it to (Taleb XXVII). Quite often, the universe turns out to be different than what we expect.

Second, just because people have tried to resolve the P vs NP debate and failed doesn't really constitute any rational form of proof. Regarding P vs. NP, Dr. Keith Devlin of Stanford University said, "It's not enough to say that a lot of bright people have tried hard for a number of years and failed" (Devlin 117). There are precedents for popular problems going long periods without being solved. For instance, Fermat's Last Theorem persisted as an open question for over 360 years. (Admittedly, that problem turned out to be resolved just as predicted.) There are also instances of problems being solved unexpectedly by relative amateurs like George Bernard Dantzig and Enrico Fermi who went on to redefine their sub-fields. Moreover, Dr. Aaronson himself said, "We have very strong reasons to believe that these problems cannot be solved without major — enormous — advances in human knowledge" (Aaronson, "Three Questions" 2). This implies that our understanding of the whole problem is significantly lacking. Working under that implication it is simply not reasonable to definitively select one outcome over the other.

The third objection, that $P = NP$ should be easier to prove than the converse is also not a definitive argument. Again, the entire P vs. NP debate shows that our understanding of mathematics is inadequate to resolve the question. Therefore, it is reasonable to assume that advances in mathematics may yield novel approaches not previously considered. Therefore, there may be some approach that in hindsight seems simple yet still shows that $P = NP$ and we simply haven't tried it yet. We don't know what we don't know.

Finally, arguments that attempt to extend solutions found in simplified representations of reality to the real world are at best approximations. Sometimes they

are woefully wrong. Any sophomore physics student can successfully argue that the simplification “ignore friction” is going to cause problems if you try to interpolate your results into the real world.

I contend that the arguments that P cannot equal NP overreach. I will concede that it is more likely that $P \neq NP$, but it is *not* certain. Therefore, due to the high potential impacts should $P = NP$, the scenario is worth thinking about. And there are those that tend to agree with that contention enough to continue trying to prove the P vs. NP problem one way or the other. In 2010 Cambridge’s Clay Math Institute named it one of the Millennium Problems and offered a \$1 million reward for solving it (claymath.org). The P versus NP web page has links to no less than 85 papers published in the last 15 years trying to prove either $P = NP$ or its converse (Woeginger).

Moreover, even if the P vs. NP debate is solved with a proof that $P \neq NP$, that does not mean that the encryption systems we rely upon are safe. There are potential advances in computing, algorithms, and mathematics that could still bring about $P = NP$ ramifications in a $P \neq NP$ world:

“ $P \neq NP$ implies that the encryption scheme is hard to break in the worst case. It does not rule-out the possibility that the encryption scheme is easy to break almost always. Indeed, one can construct encryption schemes for which the breaking problem is NP -complete, and yet there exist an efficient breaking algorithm that succeeds 99% of the time.” (Goldreich 23)

Even if $P \neq NP$ it may still happen that every NP problem is susceptible to a polynomial-time algorithm which works on “most” inputs. This could render cryptography impossible and bring about most of the benefits of a world in which $P = NP$. (Cook, “ P vs. NP ” 9)

In the end, I believe there is insight to be gained from asking the simple question, “What if $P = NP$?” despite those that believe strongly that that is a faulty assumption. Robert Heinlein said it well: "Always listen to experts. They'll tell you what can't be done

and why. Then do it." It is possible that some bright mathematician may surprise us all. And if he or she does, I think it would be a good idea to have considered what $P = NP$ might mean from a practical standpoint because certainly things will change rapidly.

This research is intended for readers with a moderate level of understanding of networking and a basic understanding of cryptography. It will provide such readers with an in-depth understanding of the $P = NP$ problem and illuminate the rationale for claims that a $P = NP$ proof will degrade information security. To begin, this research will provide background on the P vs. NP debate. It will give an origin of the discussion, followed by a characterization of the relevant complexity classes, and finally it will discuss some of the ramifications of different P vs. NP proofs. Next, it will describe the two main types of modern encryption and relate them to the P vs. NP debate. Following, the background chapter, it will describe my research methodology, including a list of Internet protocols to be analyzed as well as rationale for including each. Next, it will perform an analysis of several Internet protocols to determine how they use encryption and how they would be affected should P be proven to equal NP . Finally, I will use that analysis to draw conclusions about the actual impacts of a $P = NP$ proof on the Internet as a whole. I will conclude with recommendations for future research, including suggestions for research on how to correct problems that could be introduced by efficient NP algorithms.

II. Background

2.1 $P = NP$ Defined

The P vs NP problem, at its most basic level, amounts to a question of taxonomy: is NP a distinct complexity class, or can it be reduced to P? Biologists classify living things according to speciation rules. Similarly, complexity theorists classify decision problems according to their complexity, or the theoretic upper and lower bounds of time required to solve those problems. Decision problems are questions or algorithms which, given some input parameters, return either a “yes” or a “no” answer. Complexity theorists have defined numerous complexity classes, including Polynomial and Non-Deterministic Polynomial as well as Logarithmic, Non-Deterministic Logarithmic, Linear Exponent, and a host of others. Mathematicians like Scott Aaronson ironically refer to the host of complexity classes as the “petting zoo...” (Aaronson, “Lecture”).

Complexity classes are categorized based on the amount of time it takes to solve a decision problem within that class. More accurately, it is the number of steps, based on the number of inputs, required to generate the answer that truly separates one class from another. Generally speaking, the more inputs a problem has, the more steps it will take to produce an answer. But the important question when comparing two problems with N number of inputs is, “How many more steps?”

Peter Devlin, in his book *The Millennium Problems*, does an excellent job illustrating the different complexity classes by comparing the basic arithmetic functions. Addition of two numbers with N -digits involves $3N^1$ basic steps. Thus, adding two 4-digit numbers requires $3 \times 4^1 = 12$ steps. Addition is an example of a “linear time” process meaning that the number of steps increases linearly as N increases. You can also

think of this in terms of graphing the output of the number of steps; as N increases the points on the graph will produce a straight line. Subtraction produces the same results. Multiplication (and division) on the other hand, requires $2 \times N^2$ basic steps to solve (Devlin 119). Multiplying two 4-digit numbers takes $2 \times 4^2 = 32$ basic steps. Multiplication is therefore a “quadratic time” process and a graph of the number of steps for a given N will result in a parabolic curve.

We can generalize decision problems with functions of the form $C \times N^k$ as “polynomial time” problems, which define the complexity class “P.” Decision problems in P are solvable by some algorithm within a number of steps bounded by some fixed polynomial in the length of the input (Cook, “Complexity” 1). The practical importance of polynomial time problems is that they are generally problems that can be efficiently computed (Devlin 120). In fact, mathematicians and computer scientists consider the P the boundary between feasible and infeasible problems (Allender 4).

The types of decision problems that lie across the feasibility boundary are called “exponential time” problems or complexity class E. E problems are those that require at least C^N number of steps to complete (Devlin 121). As N grows, the time required to compute a solution to an E problem becomes infeasible rather quickly. Table 1 below (borrowed from page 122 of *The Millennium Problems*) dramatically illustrates this point by showing how much time it would take a computer that can run 1 million steps per second to solve functions at these different complexity levels:

Table 1: Growth in time to solve complexity functions as size of inputs (N) increases (Devlin 122)

Time Complexity Function	Size of N				
	10	20	30	40	50
N	.00001 sec	.00002 sec	.00003 sec	.00004 sec	.00005 sec
N^2	.0001 sec	.0004 sec	.0009 sec	.0016 sec	.0036 sec
N^3	.001 sec	.008 sec	.027 sec	.064 sec	.125 sec
2^N	.001 sec	1.0 sec	17.19 min	12.7 days	35.7 years
3^N	.059 sec	58 min	6.5 years	3,855 centuries	200 million centuries

Generally speaking, it takes less time, as a function of N , to compute a P problem than it does an E problem. Intuitively, this does not hold for all examples of P and E functions. For example, if the polynomial had an enormous exponent (say N^{99999}) or if the exponential function was 1^N , the relation would not hold. In practice, however, values of k tend to be fairly small for P problems and values of C tend to be large enough that E problems are, well, problematic. Of the types of outlying examples mentioned above, Scott Aaronson said the following during his lecture on physics:

“My answer is pragmatic: if cases like that regularly arose in practice, then it would've turned out that we were using the wrong abstraction. But so far, it seems like we're using the right abstraction. Of the big problems solvable in polynomial time -- matching, linear programming, primality testing, etc. -- most of them really do have practical algorithms. And of the big problems that we think take exponential time -- theorem-proving, circuit minimization, etc. -- most of them really don't have practical algorithms.”

At this point, we can draw an illustration of the complexity classes discussed in terms of increasing computational complexity. As shown above, the P class contains linear and quadratic types of polynomials as well as several other types like constant and logarithmic. Similarly, P problems are contained within the set E because P problems can be expressed as exponential functions. So we see our relevant complexity classes below in Figure 1.

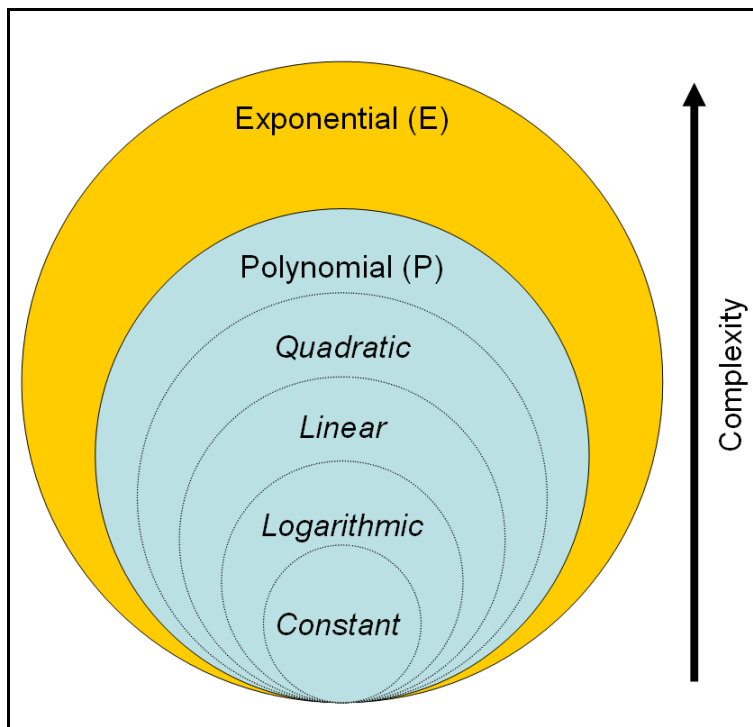


Figure 1: Complexity class P and E relation

Table 1 alludes to the fact that there is an enormous gulf between the complexity of P and E problems. Mathematicians also noted this gulf and sought intermediate measures to define process complexity (Devlin 123). Specifically, in the 1930s and 40s, mathematicians like Kurt Gödel, Stephen Kleene, John von Neumann, and Alan Turing (among others) noted a type of problem for which the computation of the solution was simple. However, this type of problem became as difficult to solve as an E problem because the only way to find the solution was to iterate over all of the possible solutions and compare them. Or put another way, it requires completing the same simple computation an impossible number of times to find a solution.

A classic example of this type of problem is the “bin packing problem.” In this problem, a subject has N number of items of different volumes that must be packed into a set of bins of a finite capacity in a way that minimizes the number of bins. The reason

this is difficult is that you must determine every possible combination of ways to stack the items and then check each solution against all the previous solutions. The next solution you try may take up less bins or volume in the bins than the previous ones. With just five items and only one bin, there are over 100 possible ways to stack the items (Stern). As the number of items to pack increases, the number of possible stacking combinations increases exponentially. However, once someone has calculated all the possible stacking combinations, checking through the solutions to see which one best satisfies the condition is a P problem.

Mathematicians dubbed these types of problems Non-deterministic Polynomial or NP. NP equals the class of problems whose solutions can be verified quickly, by deterministic machines in polynomial time (Allender 5). Finding the solution is still as difficult as a problem in E, but verifying it is as easy as a problem in P. NP problems are thus an intermediate between P and E problems (Devlin 124). Our complexity diagram in Figure 1 can be expanded to include NP problems as shown in Figure 2.

The fact that NP problems seem to blend characteristics of E and P problems prompted some mathematicians to question if it was possible to find polynomial solutions to NP problems. Perhaps, the reasoning goes, NP problems can be solved with polynomial-bounded algorithms that we haven't discovered yet. When the concept of NP problems was introduced in the 1960s, several important problems in industry were identified as being in NP. These problems included applications in efficiency, scheduling, logistics, and pattern matching. Solutions in any of these areas could mean billions of dollars in increased profits in business (Devlin 126). As a result, the $P = NP$ problem generated a good deal of attention in the 1960s.

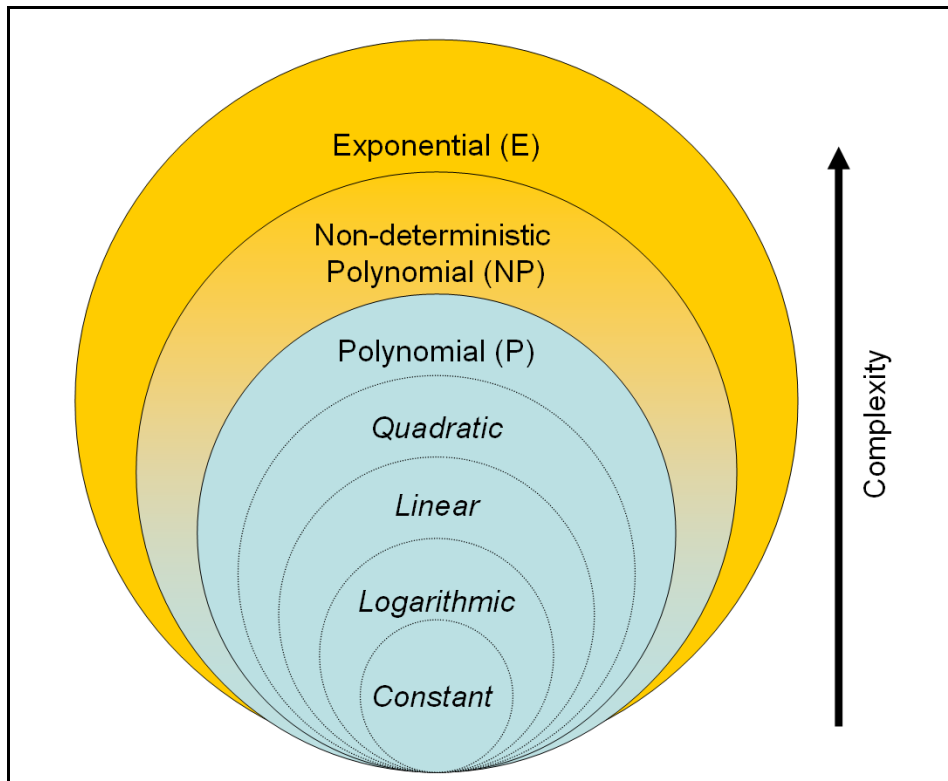


Figure 2: Complexity class P, NP, and E relation

Then in 1971, Stephen Cook published a paper called “The Complexity of Theorem Proving Procedures.” In this paper, Cook demonstrated that any problem in class NP can effectively be reduced to one specific problem he outlined (1). Moreover, this research showed that any NP problem can be translated into any other NP problem. “Although these problems might look unrelated, they’re actually the same problem in different costumes” (Aaronson, “Lecture”). From that conclusion, one can deduce that if $P = NP$ then the associated search problem for every NP problem has a polynomial-time algorithm (Cook, “Complexity” 8). Or to put it another way, if one NP problem has an efficient solution in P, then they all do. So to prove that $P = NP$ it is enough to show that any one NP problem is also in P (the converse also holds) (Aaronson, “Lecture”).

The property of reducibility or translatability of NP problems was eventually dubbed NP-completeness by Cook. Their revelation raised the stakes of the $P = NP$

debate because now mathematicians weren't just working on specific industrial computing problems, they were effectively working on all of the NP problems, the economic value of which was (and still is) immense. Beginning shortly after Cook published his famous paper, the P vs. NP problem became one of the most popular problems for researchers to try to tackle in mathematics and computer science. For more than 40 years, researchers have tried to find efficient ways to solve NP problems, or to prove that there is no efficient way to do so to no avail. This has gone on so long that most experts believe that if a problem can be proved to be NP complete then there is little reason to waste time looking for an efficient solution to the problem. Instead they attempt to side-step the problem and find approximate solutions that are "good enough" (Devlin 126).

NP completeness would also turn out to have applications in cryptography. As we will see, NP problems with their unique characteristics would come to be incorporated into a new type of cryptographic algorithm. The security of these algorithms depends on complexity-theoretic assumptions involving P vs. NP. In fact, they are predicated on the idea that $P \neq NP$. The lurking, unresolved question behind these algorithms (and the networking implementations based on them) is the open $P = NP$ problem. If $P = NP$, these assumptions are all false (Cook, "P vs. NP" 9), which is what leads many to make sweeping statements like, "The security of the internet, including most financial transactions, depend on complexity-theoretic assumptions" (Cook, "P vs. NP" 12). To check the veracity of these statements, we must first look at the different types of cryptosystems, determine if and how they rely upon NP problems.

2.2. Asymmetric Versus Symmetric Cryptography

There are two fundamental methods for encryption and decryption of messages: symmetric (shared key) and asymmetric (also known as public-key) encryption. Should it be proven that $P = NP$, the difference in how the two methods use computation to encrypt data would render asymmetric encryption nearly useless while leaving the symmetric encryption unaffected. Consequently, determining how networking infrastructures will be affected by $P = NP$ is largely a matter of analyzing whether they use encryption, and if so, what type they use, and how critical the encryption is to the operation of the infrastructure.

For all practical purposes, both methods rely upon computers to perform operations on data to transform a plaintext message into an encrypted message. Moreover, the type of computational operations each method performs gives each method strengths and weaknesses from an application perspective (not to be confused with the strength of the actual underlying cryptography). These strengths and weaknesses determine which method is practical to use in different applications, and where and how they are used in modern computer networking.

Symmetric algorithms have been used for hundreds of years to encrypt information. In fact, all classical pre-1970 cryptography used symmetric methods (Trappe 4). Systems that use symmetric algorithms use a pair of secret keys to encrypt and decrypt information. The pair of secret keys is either two copies of the same key, or the second key is derived from the first (and the first can be derived from the second) (Trappe 4).

Symmetric algorithms carry out simple mathematical functions to transpose and substitute bits, typically performing these operations over several iterations (Harris 683). The strength of symmetric algorithms comes from the fact that they use numerous rounds of computation (transposition and substitution), “seeded” with a key comprised of random bits, to scramble and garble the plaintext. The secret keys are what introduce randomness to the encryption process and the process difficult to reverse without the key. Working backwards from the ciphertext to the plaintext may be possible, but this problem lies within the exponential complexity class. Therefore, a brute force attack on ciphertext derived from a symmetric algorithm requires an iterative search over all of the possible plaintexts, and the time it takes to perform such a search can be estimated, averaged, and planned for. With modern algorithms, such a search might take decades or centuries.

The transposition and substitution operations performed by symmetric algorithms are not computationally complex nor are they processor intensive (Harris 683). As a result, symmetric algorithms are comparatively fast and lend themselves to use in applications where large amounts of data need to be encrypted. Moreover, the fact that symmetric algorithms do not make use of one-way functions or any type of problem within the NP complexity class makes them impervious to efficient NP algorithms that might be developed if $P = NP$.

Of course, symmetric algorithms have their problems. The main difficulties come in the form of key exchange and management. In key exchange, if two parties wish to initiate secure communications using symmetric encryption, they must each obtain an instance of the same key. Without resorting TO asymmetric methods, there is no way for each of them to *independently* generate a key that they both share. They must meet

together and agree on a key or use a side-channel communication method to do so. But if they've never previously met, and are separated by a large distance, they may be unable establish a secure side-channel to begin with. (If they could, then they would probably just use the secure side channel to communicate in the first place!)

Another issue with symmetric encryption lies in key management. As mentioned above, each pair of users that wants to communicate needs a set of keys (two copies of the same key). If you add a third user that needs to communicate with the first pair then you need a total of three sets of keys. A fourth user drives the number of key sets to six, and so forth. The number of symmetric keys needed to establish a mesh of N users that can all communicate with each other over encrypted channels is $N(N-1)/2 = K$ where N is the number of users and K is the number of keys (Harris 679). Table 2 demonstrates how quickly the storing of keys and matching keys to entities becomes unwieldy.

Table 2: Growth in number of symmetric keys required per user.

# of users	# symmetric keys needed
2	1
4	6
10	45
100	4,950
500	124,750
1000	499,500

Asymmetric or public key algorithms were introduced in the 1970s and presented solutions to some of the problems with symmetric algorithms (Diffie). Mathematicians in the field of Number Theory had known about seemingly one-way functions for decades. However it was not until shortly after Cook published his paper on NP-completeness that two other mathematicians used the strange properties of NP

problems to develop a new type of cryptographic algorithm. In 1976, Whitfield Diffie and Martin Hellman published a paper presenting the idea that a “public-key cryptosystem” could be implemented using NP problems (Diffie). A year later, Rivest, Shamir, and Adleman proposed a workable method to implement public-key cryptography called the RSA algorithm based on the factorization of integers (Trappe 164). Ultimately, RSA and similar algorithms would come to underpin a great deal of the security in modern computer networking. These asymmetric encryption systems allowed two parties to encrypt and share information without the need to meet and share keys (Goldwasser 13).

Asymmetric encryption performs this trick by generating a pair of keys for each user, one public and one private key. Only the user’s private key can unlock a message encrypted with the user’s public key and vice-versa. Goldwasser and Bellare outlined a very simple example of how asymmetric key generation is performed:

“Recipient B can choose at random a trapdoor function f and its associated trapdoor information t , and set its public key to be a description of f and its private key to be t . If A wants to send message m to B, A computes $E(f;m) = f(m)$. To decrypt $c = f(m)$, B computes $f^{-1}(c) = f^{-1}(f(m)) = m$.”

A more complete discussion of the use of one-way functions to generate key pairs is outside the scope of this paper, but either Goldwasser and Bellare or Trappe and Washington do an excellent job of explaining the process.

Once the first user has created his key pair, he then publishes the public key and retains the private key, keeping it secret. If another user wishes to send the first user a secure message, they use the recipient’s public key to encrypt the message. Moreover, a third, fourth, or any number of N users can use the same public key to send to the first

user. Compared to symmetric encryption, which was a one-to-one system, asymmetric cryptography provides a many to one relationship (Goldwasser 13).

Further, a user can encrypt messages with his private key, which enables another feature of asymmetric algorithms that symmetric algorithms lack, namely the ability to authenticate or digitally sign messages. Since anyone can use the public key to decrypt the message, but only the holder of the private key can sign the message, the origin of the message can be positively identified.

All of these features and flexibility of asymmetric algorithms come with a computational cost, however. The complex mathematics responsible for the features of asymmetric algorithms requires more computation and are therefore slower than symmetric equivalents (Harris 683). In fact, the amount of computation required to encrypt a similar sized block of data with an asymmetric algorithm is typically several orders of magnitude greater than symmetric algorithms (Trappe 5). This vast difference in speed and workload is why asymmetric algorithms haven't completely replaced symmetric algorithms.

In fact, in many cases where asymmetric algorithms are used, they are only used for the initial part of the communication in which a secure, in-band channel is established. That channel is then used to exchange symmetric keys which are then used to encrypt the bulk of the traffic that is exchanged. This is because symmetric algorithms are more computationally efficient, and asymmetric algorithms by comparison are special-purpose tools.

It may be fortunate that asymmetric algorithms haven't universally supplanted symmetric algorithms. Their foundation in NP problems means that if $P = NP$ (or if

mathematics makes some other type of leap that enables us to efficiently calculate solutions to NP problems), asymmetric algorithms may become much easier to break. If we can solve those one-way functions quickly in reverse, the protections granted from using asymmetric algorithms would evaporate. This would obviously have impacts on computer networking which uses encryption extensively. How much of an impact is what we will explore throughout the later sections of this paper.

Symmetric and asymmetric algorithms are the main cryptographic tools used to encrypt data, but a third type of tool is also important to mention: the hash function. Hash functions do not actually generate ciphers, but they are a widely used cryptographic function and an evaluation of how they are affected by the ramifications of $P = NP$ is important to this research. Hash functions are incorporated into many networking protocols for simplifying lookups, generating “random” values, verifying message integrity, cryptographic key derivation, and storing password values securely. A hash function will take in an input message of arbitrary length and return a hash of fixed length (Trappe 218). While it is possible that two input messages could produce identical hashes, well-designed hash functions make finding such collisions extremely rare (Trappe 219).

Hash functions are somewhat similar to symmetric algorithms in that they perform a series of formulaic steps to derive a hash given an input message (Trappe 218). However, hash functions are not intended to be reversed, meaning that the original text is not recoverable from the hash. This makes hash functions somewhat similar to asymmetric algorithms as well. However, we must make an important distinction here: hash functions do not rely on functions within the NP complexity class to derive their

“one-way-ness.” Instead, they use several types of bit-level operations, often iterative, to scramble data (Trappe 223). They also include a compression function that takes the large message and reduces it to the standard length hash. The other job of the compression function is to make bit changes in each iteration cascade through further rounds of iteration to generate as much randomness as possible. It is all of these iterative calculations rather than true “one-way” functions as described by NP algorithms that make hash functions work. As a result, hash functions would not be affected should $P = NP$ and protocols relying upon them would continue to function as they do today.

To summarize, there are two main types of encryption: symmetric and asymmetric encryption. Symmetric encryption algorithms use a random key and multiple rounds of substitution and transposition operations to generate a cipher. As a result, the process of recovering the key or the plaintext from a ciphertext falls within the E complexity class. Asymmetric algorithms use one-way functions contained within the NP complexity class to generate a cipher. If someone could prove $P = NP$, then asymmetric encryption could be solved in polynomial time, which could potentially greatly decrease the time required to decrypt asymmetric ciphers using brute-force methods. This would render asymmetric encryption easy to break while symmetric encryption would remain unaffected. Finally, hashing algorithms, like symmetric algorithms, rely upon raw computation tasks that are not NP problems and would therefore be unaffected by efficient NP algorithms.

III. Methodology

Having examined the $P = NP$ debate and extrapolated how it affects the two main types of cryptographic systems, the next step is to scrutinize the critical components of the internet to determine how they transmit information and how they depend on cryptography to do so. Since the primary purpose of inter-networked computers is to transmit information, the overarching focus of this section will be to determine if efficient algorithms for solving NP problems would negatively affect the ability to transmit information. Secondly, I will also discuss the consequences should those networking components lose their ability to have secure communications using asymmetric cryptography. Put another way, this investigation will answer two questions: 1) If $P = NP$, will computers still be able to transmit information? And 2) if so, will a loss of asymmetric cryptography make it undesirable to continue transmitting information even if it is possible?

To frame this analysis, I will use two common models used for network security development: the Open Systems Interconnection (OSI) Reference Model and the Confidentiality, Integrity, Availability (CIA) triad. The CIA triad is used to determine how breaches in information security affect the information being manipulated by an information system (Perrin). Confidentiality refers to a system's ability to protect information from unauthorized disclosure. Integrity is a system's ability to prevent unauthorized or accidental modification of information. Finally, availability refers to the ability of a system to present information to authorized users when it is requested. I will use these three aspects to determine the affects of asymmetric cryptography losses on protocols and systems.

I will also progress through the OSI Reference Model's abstraction layers and address protocols at each layer. This approach is intended to provide organization rather than assign any ranking of importance to individual protocols or infrastructures. Briefly, the OSI model was developed in the late 1970s to describe how information from a software application on one computer traverses a network and arrives at a second software application on a second computer (Kurose 52). The OSI model defines seven protocol layers: application, presentation, session, transport, network, data link, and physical.

A detailed explanation of the OSI model is outside the scope of this paper, but Cisco Systems Inc. publishes an excellent primer in its Internetworking Basics DocWiki which is included in the references section of this report. The protocols in each of the OSI layers govern, through formal sets of rules and conventions, how computers exchange information across a medium (Cisco).

Inter-networked computer systems rely upon a multitude of infrastructures and protocols to transmit information. It is not feasible to examine all of the protocols in common use today in a single document. However, there are several protocols that provide a good measure of the overall impact to the Internet as a whole. These protocols are examined in this research either because they are nearly ubiquitously adopted or are highly representative of similar general-use protocols. The list of selected protocols follows in Table 3. This section, describes the protocols or technologies to be examined and provides the rationale for why they merit consideration.

Table 3 – List of Key Internet Protocols

Protocol	OSI Layer
Bulk Encryption Techniques	Data Link
Ethernet	Data Link
WPA2	Data Link
Routing Protocols (OSPF, BGP)	Network
IP	Network
IPsec	Network
TCP	Transport
UDP	Transport
SSL / TLS	Presentation
SSH	Application
DNS	Application
Electronic mail (SMTP, IMAP, POP)	Application
Web and HTTP	Application

3.1 Protocol Selection

This section describes the protocols or technologies to be examined and provides the rationale for why they merit consideration. For various reasons, these protocols represent critical components of the Internet. If new algorithms caused systemic problems with any of these protocols, it could prove predictions of Internet-wide failure to be possible.

3.1.1 Bulk Encryption

Bulk encryption, although not technically a protocol, is an important technology to examine because it is widely adopted to secure large amounts of network traffic. For the purposes of this paper, bulk encryption is defined as physical or data link layer encryption in which all packet data (including header and trailers) is encrypted along a communication path. Typically, bulk encryption is implemented with hardware devices and messages must be decrypted at each hop point so that the router can determine where to send the packet next. Bulk encryption is also called link encryption or traffic-flow

encryption. Bulk encryption is employed by service providers in large-scale network implementations, to connect geographically-separated sites, and to create intranets. Of particular interest in this discussion is the fact that the DoD uses bulk encryption to implement intranets such as Non-secure Internet Protocol Router Network (NIPRNET), Secure Internet Protocol Router Network (SIPRNET) and Joint World-wide Intelligence Communications System (JWICS).

Although bulk encryption is often used to establish intranets, in this paper it is distinguished from Virtual Private Networking (VPN) protocols such as Point-to-Point Tunneling Protocol (PPTP), Internet Protocol Security (IPsec), and Transport Layer Security (TLS). Each of these protocols deserves separate discussion due to their varied characteristics, usages, and the fact that they operate at different layers of the OSI model. Additionally, this paper will forgo an in-depth analysis of PPTP in favor of IPsec, TLS and Secure Sockets Layer (SSL) protocols. This is because, despite still-prevalent use, PPTP is generally considered to be cryptographically unsecure (Schneier 1) (Cameron) and the aforementioned protocols are better representatives of current technologies.

3.1.2 Ethernet (IEEE 802.3)

The Ethernet protocol determines how networked computers connect to and send data across local- and wide-area networks at the data link layer of the OSI model. It has survived as the major LAN technology because it is easy to implement, manage, and maintain. The protocol, and the working group that maintains it, has actively evolved and expanded to meet increasing bandwidth and demands. Ethernet also has topological flexibility for diverse network implementation. As a result, Ethernet hardware is ubiquitous, inexpensive, and guarantees successful interconnection and operation of

standards-compliant products, regardless of manufacturer. Due to these factors, Ethernet has largely replaced all competing LAN and WAN technologies with more than 95-98 percent of subscriber traffic starting on Ethernet (Dell) (Verizon). The nearly-universal adoption of Ethernet makes it an important candidate for inclusion in this analysis—if efficient NP algorithms impact this protocol, it could affect nearly every Internet-connected computer.

3.1.3 Wi-Fi Protected Access 2 (WPA2) (IEEE 802.11i)

Wi-Fi is one of the most widely used technologies for LAN connectivity, next to wired Ethernet, with one of the highest growth rates in mobile and consumer electronic devices (Wi-Fi Alliance 2). One-third of US households with broadband Internet access have a Wi-Fi network, over one billion Wi-Fi chipsets shipped in 2011 and the number of annual shipments is forecast to reach two billion by 2015 (Wi-Fi Alliance 2). While there are still Wi-Fi networks using WPA or Wired Equivalency Protection (WEP), Wi-Fi Protected Access 2 (WPA2) is quickly becoming the de facto standard for wireless connectivity due to superior encryption protocols, certification requirements, and improvements in hardware capabilities.

WEP was the first broadly-adopted wireless standard. It was introduced in 1999 as part of the first 802.11 standard. As a first-generation security solution, WEP was vulnerable due to limitations in key size (initially 40 bits, later extended to 104 bits) and its lack of replay detection (Wi-Fi Alliance 4). Due to these limitations, the Wi-Fi Alliance moved to WPA as an interim solution to WPA2 in 2003. WPA included a subset of WPA2 features, but was backwards compatible with some of the first-generation, processor-limited WEP hardware. WPA was a second-generation interim

solution designed to address WEP vulnerabilities in anticipation of the ratification of the IEEE 802.11i amendment.

Support for WPA2 security has been mandatory for all Wi-Fi certified equipment since early 2006 (Wi-Fi Alliance 6). In addition to the certification requirements, WPA2 adoption by vendors and consumers has become nearly universal for several reasons. First, processor speeds have increased dramatically since WEP and new hardware can easily take advantage of strong encryption. Second, WPA2 is standards-based and strongly interoperable between brands. And third, most standardized implementations are extremely easy to use and activate.

Much like Ethernet, if WPA2 would be severely weakened by the employment of efficient NP complete algorithms, there would be broad security implications. Moreover, due to the broadcast nature of wireless technologies, WPA2 implementations would be even more vulnerable than wired equivalents in such an environment. The broad adoption of WPA2 and the security implications inherent in open wireless make it a strong candidate for inclusion in this analysis.

3.1.4 Routing Protocols

Routing protocols form the foundation of the Internet's ability to transfer information on a global scale. Without routing protocols, the Internet could never have scaled to the size it is today. Moreover, identifying, locating, and determining a path to another computer elsewhere in a global network would be nearly impossible without routing protocols. Kurose and Ross go so far as to state that routing protocols are, "absolutely critical ... for the internet" (Kurose 401). The major routing protocols were even listed in the 2003 US National Strategy to Secure Cyberspace as one of three

infrastructures that should be protected in order to protect US interests (the others being IP and DNS) (“National Strategy to Secure Cyberspace” 30).

Routing protocols are broken into two categories: Interior Gateway Protocols (IGP) and Exterior Gateway Protocols (EGP). IGP determine how traffic is routed within an autonomous network, and EGP determine how traffic bound from one network to another is routed. IGP are more varied than EGP because network administrators have more flexibility within their own enclaves to choose protocols that meet their unique needs. However, the two most extensively used IGP are Routing Information Protocol (RIP) and Open Shortest Path First (OSPF) (Kurose 394). For IGP, this paper will focus on analyzing OSPF exclusively because OSPF was planned to be the replacement protocol for RIP and as such has many of the same characteristics but also includes additional advanced features (Kurose 398).

For EGP, the Border Gateway Protocol (BGP) has become the de facto standard on the Internet (Kurose 401). BGP achieved this status for a variety of reasons, some focused on efficiency, some focused on economic considerations. As the Internet evolved, Internet Service Providers began to have vested interests in controlling the way traffic flowed across the Internet for economic reasons (Caesar 1). During this evolution, BGP, which started as a simple protocol, received incremental modifications which allowed ISPs to set policies to control routing. Specifically, BGP allows ISPs to control route selection and propagation (Caesar 1). Gradually, the ability to enforce policies on routing decisions coupled with an underlying simple and efficient routing algorithm won BGP nearly universal adoption for EGP.

Due to the criticality and ubiquitous use of routing protocols, and because of OSPF and BGP's dominance among routing protocols, an analysis of their susceptibility for disruption is important in this research. If either protocol relies heavily on asymmetric encryption that could be affected by efficient NP algorithms, the stability of the Internet could be in jeopardy.

3.1.5 Internet Protocol

The fact that the Internet and the Internet Protocol (IP) share a name isn't a coincidence. Both are based on the interconnection of computers and networks, and without IP (or something very like it) there wouldn't be an Internet. Kurose and Ross point out that IP addressing "is of central importance to the Internet" (Kurose 348). IP is the protocol responsible for addressing individual nodes on a network, specifying datagram addressing, and the transmission of those datagrams from one node to another (Postel, "RFC 791" 5). Nearly every meaningful bit that gets transferred between more than two computers is encapsulated in an IP datagram. Every protocol discussed from this point forth uses IP as its foundation for moving data across a connection. Essentially, IP is the primary network protocol used on the Internet, and while it may be intuitive that the protocol has very little to do with encryption, an analysis that disregards IP is certainly incomplete. Should IP have some fatal flaw exploitable by efficient NP algorithms, one could certainly make a case that such a situation could put the Internet as a whole in danger.

3.1.6 Internet Protocol Security

There is however a security extension to the IP protocol, namely Internet Protocol Security (IPsec) that does rely on encryption and also deserves analysis. IP and IPsec are

independent protocols, and while IPsec certainly depends on IP to transmit datagrams, IPsec is no more related to IP than any other protocol. IPsec is a widely used, open standard to provide security services for traffic at the IP layer (Kent and Seo 3). IPsec is designed to secure communications by providing authentication and encrypting each datagram in a communication session, thus enabling private communication over public networks. Its main advantage over other encryption protocols is that it operates at the network layer and can provide encryption services for applications that do not incorporate it into application protocols.

IPsec has become the most common network layer security control (Frankel, ES-1) IPsec is incorporated into several important operating systems including Linux derivations from OpenBSD (including Mac OSX), Juniper OS, Cisco IOS, and Microsoft OS suites. Since it is widely adopted, standards based, and interoperable, IPsec is often used to establish VPNs (Frankel ES-1). Moreover, it is recommended by NSA as an alternative to SSH for remote logon and administration of Cisco routers and other infrastructure devices (NSA, “Configuring”). IPsec is another example of a protocol that warrants analysis due to its ubiquitous use to secure network traffic.

3.1.7 Transmission Control Protocol and User Datagram Protocol

At the transport layer, there are two main protocols that interface with the IP protocol and move the vast majority of the traffic on the Internet. These protocols are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). Despite the fact that these two protocols take two very different approaches to data transport, their handling of encryption and security is nearly identical. Therefore, for analysis purposes, they are combined in this paper.

TCP was developed in conjunction with IP in the late 1970s before many of the security concerns of today's Internet emerged. Also like IP, TCP's early roots and its success has resulted in near-universal adoption. "The effectiveness of [TCP/IP] led to their early adoption in production environments, to the point that, to some extent, the current world's economy depends on them" (Gont 5). Indeed, TCP is used to carry web traffic, email, file transfer and sharing, secure shell, and a host of other applications. If it carried some flaw that could be exploited with efficient NP algorithms, a case could easily be made that $P = NP$ could jeopardize the ability of the Internet to function.

UDP was developed just shortly after TCP and released in RFC 768 in 1980. Unlike TCP, UDP is an extremely simple protocol and does the bare minimum a transport protocol must do (Kurose 211). Other than two fields for specifying source and destination ports, the only other thing it adds is a checksum and a length header (Postel, "RFC 768" 1). In fact the greatest indicator of UDP's simplicity is probably the length of its specification: RFC 768 is three pages long. By comparison, RFC 761, the original RFC for TCP is 84 pages long. And unlike TCP, there are only a small handful of clarifying or expansion RFCs for UDP. Still, many applications rely upon UDP including Domain Name System (DNS), streaming media, internet telephony, some forms of FTP, and some IP tunneling protocols. Essentially, if it doesn't use TCP, it most likely used UDP. So again, if UDP relies on any kind of asymmetric encryption, efficient NP algorithms could cause significant problems.

3.1.8 Secure Sockets Layer and Transport Layer Security

Secure Sockets Layer (SSL) and Transport Layer Security (TLS) are a pair of protocols that sit at the presentation layer and provide confidentiality, integrity, and

authentication services for applications (Kurose 727). Applications that require these services can include SSL or TLS code in the application. When the application sends data bound for the network, it invokes an SSL or TLS socket that handles the security services and then passes the encrypted and/or authenticated data to the TCP socket (or in some cases UDP) (Kurose 96).

SSL and TLS are very similar protocols (Dierks 86). In fact, the various versions of SSL and TLS all stem from the first version of SSL released by Netscape in 1995. (SSL 1.0 was in use as early as 1994 and T. Woo had released “SNP: An interface for secure network programming” in 1994, but SSL 2.0 was publicly released in 1995 and served as the foundation for all future versions of SSL and TLS (Kurose 727).) SSL 3.0 and later TLS versions 1.0, 1.1, and 1.2 effectively represent progressive security improvements to the same protocol (Dierks 86). They do not, however, interoperate so many of these versions are still in use on the Internet for compatibility, but SSL 3.0 and TLS 1.2 predominate (Dierks 6). Due to the close similarities between the versions of SSL and TLS, this paper will primarily analyze TLS version 1.2. The results and conclusions from that analysis have been checked by the author to apply to previous protocol versions.

TLS is supported by all major web browsers (all Mozilla variants, Microsoft Internet Explorer (all versions), Google Chrome, Safari, and Opera) and all major web server platforms (Apache, Microsoft Internet Information Services, nginx, and Google Web Server) (Kurose 727). Through these platforms, nearly all e-commerce purchases ride over a TLS connection, making it the protocol responsible for securing more than two hundred billion dollars worth of transactions annually (Winters 1). In addition to

securing web traffic, because TLS can be transparently implemented on top of TCP and UDP, it has been adopted to secure numerous other types of traffic. Some examples include Secure FTP, Simple Mail Transfer Protocol (SMTP), and internet telephony. TLS is even being used to tunnel the entire network stack in VPN applications like OpenVPN (Yonan).

TLS (and by extension SSL) obviously rely on encryption to provide the confidentiality, integrity and authentication services it was designed for. Due to its nearly ubiquitous use to secure e-commerce transactions, as well as numerous other types of traffic, TLS is a critical protocol to include for analysis in this research. If extensive use of asymmetric encryption in this protocol would make it vulnerable to attack with efficient NP algorithms, it would certainly become a huge target due to the potential economic impact. Moreover, a general loss of trust in TLS on the part of the public could have dramatic consequences for global economies.

3.1.9 Secure Shell

Secure Shell (SSH) is an application layer protocol that can be used for many purposes. While the main use for SSH is remote logon, SSH is also capable of secure file transfer, port forwarding or tunneling, establishing secure VPN, and mounting remote file systems (OpenSSH.com). For this paper, the main concern regarding SSH is the fact that it is used routinely to provide a logon shell on a remote host (OpenSSH.com). Often that remote host is a network security device such as a firewall, router, switch, or bulk encryptor. SSH's remote shell capability was actually the driving force for its creation as a protocol. In 1995 Tatu Ylonen designed the first version of SSH to replace rlogon and

Telnet after a password sniffing attack against those protocols at the Helsinki University in Finland.

Since then, SSH has gone on to widespread use and acceptance across many platforms including most Unix variants, Microsoft Windows, Apple's OS X, and Cisco's IOS. In fact, both Cisco and the U.S. National Security Agency recommend using SSH (with Diffie-Hellman group 14 key exchange) as the channel to administer Cisco routers (NSA, "Configuring"). Additionally, the other major router vendors, including Nortel, Juniper, Alcatel, Avaya, Hewlett Packard (formerly 3Com), and Huawei all provision their routers for remote administration via SSH. These are also the primary vendors of firewall and bulk encryption devices and those devices are also typically administered via SSH. (As previously noted, many of these devices also support IPsec VPN connections as an alternative logon method. We will see that the distinction is probably moot due to the similarities between the two protocols with relation to the $P = NP$ problem.)

Most network administrators do not locally administer their network infrastructure devices. Whether those devices are located in distant facilities or because it is more efficient to reach multiple machines from a single workstation, administrators usually log onto devices remotely. And SSH is the de facto standard for that remote logon. If SSH were compromised due to weaknesses in asymmetric cryptography, whether due to efficient NP algorithms or other reasons, the majority of the Internet's core infrastructure could be vulnerable to attack. Realistically, there are other prevention methods in place to prevent such an occurrence in many instances—often the administration interfaces for devices like this are placed out of band on private networks.

However a compromise of SSH would be a significant concern for the health of the Internet as a whole.

3.1.10 Domain Name System

The Domain Name System (DNS) exists to provide a bridge between the human preference for mnemonic naming conventions and the need for routers to have a hierarchical, fixed-length addressing system (Kurose 133). At the advent of the Internet, all devices on the network registered their addresses and host names with Stanford Research International. In turn, all hosts on the network downloaded this hosts file and used it to map host names to addresses (Klensin, “RFC 3467” 2). As hosts on the rapidly expanding network grew more numerous, this approach became unsupportable. In response, Paul Mockapetris designed a scalable, automated name resolution system, DNS, in 1983 (Mockapetris 1).

DNS is really the only name resolution service on the Internet for host lookups. It consists of a hierarchical server system deployed world-wide (Kurose 136). DNS servers store resources records in a distributed database and perform lookup requests to identify addresses from host names. DNS also provides other important network services such as host and mail server aliasing and load distribution (Kurose 134).

Since DNS relies upon the underlying UDP transport protocol, it is technically an application layer protocol. However, unlike the typical application DNS provides the aforementioned services to other applications. As such, DNS provides a “core Internet function,” (Kurose 134) and without it, locating hosts on the Internet would be an untenable task. Therefore, DNS is another protocol which must be considered when

determining how efficient NP algorithms would affect the Internet. If DNS could be crippled by such algorithms, it could be devastating to the Internet.

3.1.11 Electronic mail

Electronic mail is one of the oldest applications on the Internet and was the most popular application on the early Internet (Kurose 120). Electronic mail's success and popularity was a motivating factor in the growth and success of the Internet. Even today, e-mail is one of the most used and important applications on the Internet (Kurose 121). Imagine having to perform your job or maintain personal relationships without some form of e-mail! Without e-mail, many modern business processes would come to a screeching halt, therefore I choose to analyze it as one of the representative applications in this paper.

Admittedly, Simple Mail Transfer Protocol (SMTP) is not the only protocol used to move e-mail. It is however the principle protocol for doing so (Kurose 123). While mail clients use a few other protocols to actually retrieve e-mail, SMTP is the protocol used between e-mail servers to move traffic. Still, it is worth briefly examining some of the other protocols such as Post Office Protocol (POP) and Internet Message Access Protocol (IMAP). Email is even transferred over HTTP. However, as we will see, all of these protocols look very similar when viewed through a $P = NP$ lens.

3.1.12 Web and HyperText Transfer Protocol

While the Internet existed before Hypertext Transfer Protocol, it wasn't until HTTP was released (and with it the first Internet application, the World Wide Web) that the Internet became something the general public could use (Kurose 100). HTTP is the Web's application-layer protocol and despite the fact that all the other protocols

discussed until now comprise much of “the Internet,” the majority of the public view the Web and the Internet as synonymous. Until HTTP, the Internet was merely something that researchers, academics, and the military used for message traffic and file transfers. HTTP “dramatically changed, and continues to change, how people interact inside and outside their work environments” (Kurose 100). The web is the fundamental protocol that most people use to interact with the Internet. Surely, if the way this protocol functioned should be negatively impacted by efficient NP algorithms, there could be something to the claims that $P = NP$ would mean the end of the Internet.

IV. Analysis

This section, Analysis, will examine each of the protocols we selected for analysis in Section 3, Methodology. First it will determine what (if any) type of encryption each protocol uses. If that protocol uses encryption, I will determine how a loss of asymmetric encryption in that protocol would impact the integrity, availability, and confidentiality of common or important systems which rely upon that protocol. As described in Section 3, this analysis will follow the OSI model to organize the protocols into a logical flow.

Additionally, this section will analyze how a $P = NP$ proof may come about. There are different types of possible proofs in mathematics and depending on which type is used in a hypothetical $P = NP$ proof, the ramifications could vary. For each type of proof I will analyze how much time the network engineering community would have to implement replacement technologies to shore up vulnerabilities introduced from efficient NP algorithms. I will also look at a best and a worst case scenario for how efficient NP algorithms might be. From this analysis I will show a scale of possibilities of the actual impacts of a $P = NP$ proof on the security of the internet.

4.1 Physical Layer Protocols

At the lowest layer, the physical layer, the protocol defines the electrical, optical, or mechanical specifications for operating a link. These specifications include defining voltage levels, timing of voltage changes, and physical data rates. At this layer, encryption is rarely used. On wired networks, unauthorized access at the physical level is typically mitigated through making the wires difficult to access. Moreover, encrypting traffic at a higher layer in the OSI stack is much more effective.

In wireless networking, there is a growing interest in physical layer security because eavesdroppers can use Media Access Control (MAC) address information in various attacks (Gollakota 1). Encryption at the physical layer would protect MAC information. However, there are currently no major standards for physical layer security over wireless or wired networks either proposed or in implementation. In the instances available for review it appears that existing physical layer encryption schemes rely upon symmetric algorithms (Gollakota 10). Presumably, asymmetric algorithms would be ill-suited for any kind of physical layer encryption because physical layer links need to be efficient and place a premium on speed of transmission. It seems fair to conclude that physical layer protocols do not currently rely on encryption, and if encryption were to be used in the near future, it would not be vulnerable to efficient NP algorithms.

4.2 Data Link Layer Protocols

At the data link layer, protocols tend to provide the local delivery of frames between devices. Often, these protocols work within the same LAN or between adjacent nodes in a WAN. Much like the physical layer, these protocols are mainly focused on the work of actually moving bits versus security or encryption, with a few notable exceptions. Bulk encryption obviously addresses security for the purposes of protecting large amounts of traffic to include address data. For much the same reasons, WPA2 includes encryption to protect data link layer traffic due to the broadcast nature of wireless and the risk of eavesdropping. The data link layer protocols reviewed were bulk encryption techniques, Ethernet and WPA2.

4.2.1 Bulk Encryption

Bulk encryption protects information at the data link level as it is transmitted between two points within a network. Often, the two points are external interfaces or bottlenecks on a network in which all traffic flows to and from another network, possibly across a public, unprotected network. As a result, these points are mainly used for high-speed, high-data throughput between telecommunication facilities. While bulk encryption is important for protecting data in transmission, it is equally important that the encryption process not appreciably affect throughput. In order to achieve the required high-speeds, bulk encryption techniques primarily employ hardware encryption devices (Thales 2). These devices typically use an encryption algorithm on task-specific logic arrays rather than a general-purpose computer to perform the encryption. The main intent of these hardware devices is to perform the encryption tasks as efficiently and quickly as possible, although they also often provide a greater degree of abstraction from attacks and greater scalability.

As discussed in Section 2.2, asymmetric encryption algorithms require orders of magnitude more processing to perform encryption tasks compared to symmetric algorithms. As a result, nearly all known bulk encryption is carried out using symmetric algorithms, and each link will typically use a separate key to encrypt traffic (Open University). U.S. government agencies are required by the Committee on National Security Systems to follow the High Assurance Internet Protocol Encryptor (HAIPE) Interoperability Specification for bulk encryption devices (Grimes). HAIPE has also become the de facto standard for many commercial and banking bulk encryption applications.

HAIPE compliance requires the use of approved symmetric encryption algorithms. HAIPE Suite A cryptography contains classified algorithms that the US government does not release publicly (NSA “Suite B”). However, those algorithms are incorporated into bulk encryption devices used to secure classified networks such as SIPRNET and JWICS. HAIPE Suite B cryptography includes “security standards that are appropriate for protecting information up to the SECRET level” (NSA “Suite B”). HAIPE Suite B cryptography typically uses the Advanced Encryption Standard (AES) algorithm. The AES algorithm is a symmetric block cipher that uses keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits (NIST, “AES”). HAIPE Suite B devices are used by the United States to bulk encrypt traffic on unclassified networks such as NIPRNET.

Since bulk encryption’s demand for high efficiency necessitates symmetric algorithm use, it does not use any of the potentially vulnerable one-way functions concerned in the $P = NP$ problem. Networks like NIPRNET, SIPRNET, and JWICS or their civilian counterparts would not directly be vulnerable to efficient NP algorithms. (However, there are secondary concerns that could impact the availability of such networks depending on how the underlying routing infrastructures are implemented and managed as we will discuss in further sections.) Therefore, data confidentiality conferred by bulk encryption would not be at increased risk in a $P = NP$ environment.

4.2.3 Ethernet

Ethernet as a standard is mainly concerned with connecting a computer to a local area network at the media access level. The standard specifies methods to share communications media, data transmission rates, and reliability--but not security or

encryption. A search on the IEEE 802.3 standard specification for the word “encryption” does not return a single match (IEEE, “Ethernet”). A search on “security” returns recommendations to apply appropriate physical and application security measures to an Ethernet implementation. As succinctly put in an advertisement for bulk encryption devices on Certes Network’s web page, “as a shared infrastructure technology, Ethernet has no inherent security” (Certes).

Like many of the other protocols examined in this paper, Ethernet is a fairly old protocol. When networking was in its infancy, function rather than security dictated how protocols were developed. As more diverse applications began leveraging the underlying infrastructure these protocols provided, security became a concern, but typically security was implemented at OSI layers closer to the actual application. For protocols like Ethernet, security considerations tend to impact efficiency and are therefore pushed up the OSI stack.

Like bulk encryption and routing (as we will see later), there are some potential implementation concerns with Ethernet hardware that must be taken into consideration to guarantee there are no impacts given a $P = NP$ environment. However, in general, because the Ethernet standard contains no provisions for encryption of any kind, efficient algorithms for solving NP problems would have no bearing on the protocol.

4.2.4 Wi-Fi Protected Access 2 (WPA2)

Wired network environments provide an inherent level of protection insofar as an unauthorized user must gain physical access to a wire or some other medium to eavesdrop on network traffic. Wireless networks, on the other hand, work in an innately broadcast domain. In order to provide a comparable level of privacy to that of wired

implementations, wireless access protocols generally incorporate encryption at the protocol level. However, wireless protocols are still data link layer protocols and must place a high degree of importance on throughput and performance (like Ethernet and bulk encryption) and therefore tend to use symmetric algorithms that have low computational overhead.

WPA2 can operate in two modes, either Enterprise or Personal, depending on the requirements of the network. Regardless of the mode used, WPA2 uses the Counter Mode with Cipher Block Chaining Message Authentication Code Protocol (CCMP) protocol, based on the AES algorithm for authentication and data encryption (Jacobs). AES is a symmetric algorithm. The following explanations of WPA2 Personal and Enterprise are mainly derived from the IEEE 802.11i standard (IEEE, “802.11i”).

WPA2 Personal Mode is generally used in smaller implementations such as home networks or small businesses. This is because WPA2 Enterprise Mode requires additional infrastructure for user authentication as we will see shortly. The connection process in WPA2 Personal begins with an administrator distributing (out of band) a pre-shared key (PSK) to all devices that will connect to the network via WPA2. Some hardware supports multiple PSK, but often a single PSK is used throughout a WPA2 Personal implementation. Next, the client device initiates association with the access point (AP) and both devices verify that they possess the same PSK. The devices then begin an authentication process which involves a four-way handshake. During the four-way handshake, the PSK and the Service Set Identifier (SSID) are used to generate a pairwise master key (PMK). The client and AP exchange messages using the PMK to create the pairwise transient key (PTK) at both the client device and the access point.

Finally, AES keys are derived at both the client and the AP from the PTKs to encrypt data exchanged between client device and access point.

Although this process may seem similar to the process used to generate asymmetric keys, no one-way functions are used in the four-way handshake processes. The security of the WPA2 Personal authentication is based on the sharing of encryption keys between the client and the AP prior to association.

WPA2 Enterprise uses many of the same procedures as Personal, but it adds IEEE 802.1X authentication. 802.1X is a standard for authentication with network access control features (Wi-Fi Alliance 1). In WPA2, there are a variety of possible “Extensible Authentication Protocols” (EAP), each with various costs and benefits. Table 4 lists the common EAP types. Like WPA2 Personal, some of these methods require the pre-sharing of user certificates as a first step in authentication. WPA2 Enterprise also requires the client to associate with the AP. However after association, the AP hands the client off to an authentication server as specified in the 802.1X standard. At this point, the authentication server evokes one of the EAP methods to perform authentication.

Table 4 – WPA2 / 802.1X EAP Types (Allied Telesys)

EAP Type	Server Authentication	Supplication Authentication	Dynamic Key Delivery	Security Risks
EAP-MD5	None	Password Hash (MD5)	No	Man-in-the-middle (MitM) attack, Session hijacking
LEAP	Password Hash	Password Hash (MD5)	Yes	Identity exposed, Dictionary attack.
EAP-TLS	Public Key (Certificate)	Public Key (Certificate or SMART Card)	Yes	Identity exposed
EAP-TTLS	Public Key (Certificate)	CHAP, PAP, MS-CHAP (v2), EAP	Yes	MitM attack
PEAP	Public Key (Certificate)	Any EAP such as EAP-MS-CHAPv2 or Public Key	Yes	MitM attack; identity hidden in phase 2 but potential exposure in Phase 1

EAP-Message Digest 5 (MD5) uses a 128-bit hashed value of a server challenge and the user's password to verify the authenticity of the client. MD5 is known to have critical weaknesses and is not suitable for use in most environments, especially wireless networks. It also only provides one-way authentication (of the client) and without mutual authentication passwords and hashes can be sniffed and used in man-in-the-middle (MitM) attacks (Allied Telesys). While EAP-MD5 is weak and not typically implemented as of the writing of this paper, it does not rely upon asymmetric cryptography as discussed in section 2.2.

LEAP is a proprietary EAP developed by Cisco Systems. It is similar to EAP-MD5 and also uses the MD5 hash function. The main difference is that LEAP supports mutual authentication and uses dynamically generated WEP keys to encrypt data transmissions. It is therefore less at risk to MitM attacks than EAP-MD5. However, station identities and passwords remain vulnerable to attackers armed with sniffers and dictionary attack tools (Allied Telesys). Like EAP-MD5, LEAP also does not rely upon asymmetric cryptography.

EAP-Transport Layer Security (TLS) is based on the TLS protocol which we will discuss at length later in the paper. As it relates to WPA2 Enterprise, EAP-TLS requires certificate-based asymmetric encryption and mutual authentication of the client and the network. EAP-TLS uses an encrypted TLS tunnel, making it resistant to MitM and sniffing attacks. It also requires administrators to maintain identity certificates, which also use asymmetric infrastructures, on all devices.

EAP-Tunneled TLS (EAP-TTLS) and Protected EAP (PEAP) were both designed to simplify 802.1X application, and use similar means of authentication. EAP-TTLS and

PEAP use certificate-based asymmetric encryption and provides mutual authentication of the client and network. Unlike EAP-TLS, EAP-TTLS and PEAP only require server-side certificates to achieve the mutual authentication (Allied Telesys). Both of these EAP methods are resistant to MITM and sniffing attacks. Once the authentication server accepts the client identity, it sends its credentials to the client device. The client device then identifies the server, and if it is validated, it in return submits user credentials for validation. If validated, the client and authentication server then generate a PMK and PTK. Then the authentication server passes the client to a secure port on the AP and the client and AP begin the four-way handshake process exactly as described in WPA2 Personal Mode. Finally, AES encryption keys are derived from the PTKs to encrypt data exchanged between client device and access point.

Like WPA2 Personal, WPA2 Enterprise relies upon symmetric keys for actual transfer of data. However, for scalability in large enterprise networks, it automates the authentication process using the authentication server. As we see, some of the EAP methods used in that authentication process depend on asymmetric encryption algorithms. Efficient NP algorithms could therefore compromise these authentication methods. Depending on how efficient the NP algorithms were, they could reduce EAP-TLS, TTLS, and PEAP to a security level equivalent to EAP-MD5 or LEAP, neither of which are recommended for securing important data (Cisco). These potential vulnerabilities could cause loss of data from previously secure wireless networks if administrators do not take preventative action.

Should WPA2 Enterprise implementations be rendered insecure, administrators could fall back to WPA2 Personal which relies solely on PSK for authentication.

However, this may not be feasible for large implementations due to the complexity of distributing PSK material to dozens or hundreds of devices. Resorting to WPA2 Personal would also cause non-repudiation problems on a large network since many clients would log on with the same PSK and other authentication methods would need to be used in conjunction. Finally, although EAP-MD5 is considered cryptologically weak, it is because the MD5 algorithm is flawed—hashing as an authentication method is still quite viable. As a result, the EAP-MD5 standard could serve as a foundation with MD5 substituted with a more secure hashing functions such as Secure Hash Algorithm-1 (SHA-1), codified in FIPS PUB 180, to provide an EAP solution that would be immune to the effects of efficient NP algorithms.

4.3 Network Layer Protocols

Until now, we have discussed protocols at the physical and data link layers of the OSI model. Those protocols have been primarily concerned with identifying devices on the network and enabling them to send traffic to the next node on the network. Often that next node is going to be the destination or gateway router for the client device. We can think of the physical and data link protocols as being the local roads that connect individual houses and businesses within a town. To access the highways and interstates of the internet, we must move up to the network layer and discuss routing and Internet Protocol.

4.3.1 Routing Protocols

Much like some of the other workhorse protocols we've examined to this point, routing protocols and the hardware solutions that implement them place primary emphasis on performance and efficiency. The standard in the routing industry is for a

router to be able to perform at “line speed,” meaning that the forwarding decision (which outbound link to use) must be performed in less time than it takes for the entire packet to be received on the inbound link (Kurose 333). Backbone routers on the Internet must operate at high speeds and typically perform millions of route lookups per second. Interior routers must perform similarly because network congestion at the infrastructure level is rarely tolerated. Surprisingly, some routing protocols are able to implement security processes while achieving these desired performance benchmarks.

In OSPF, routers on a network construct a complete topographical map of all of the computers located on the local autonomous network. Those routers also exchange this map information with other routers on the local network by broadcasting their routing information to one another. They perform this task whenever there is a change in the state of a link on the network or periodically if there are no changes. Each router then locally runs a shortest path algorithm to determine the shortest path to all nodes on the local network. By default, all of these tasks are performed without resorting to any kind of authentication or encryption (Kurose 399).

OSPF can, however, be configured such that inter-router link state update broadcasts can be authenticated (Kurose 398-399), which would prevent malicious actors from injecting unauthorized information into routing tables, typically for the purposes of eavesdropping or denial of service. Under OSPF, there are two authentication options, simple and MD5. In either case, a password or shared secret key must be configured on each router within the authentication. In simple authentication, when a router sends link updates, the password is included in plaintext, which is obviously not very secure. Under MD5 authentication, when a router sends a link update, it uses the shared secret key to

compute a MD5 hash of the transmitted data and includes that hash. The receiving router then uses the data and the shared secret key to compute its own hash and compares it to the received hash. If they match, the link update is validated. If they do not match, the update is discarded. OSPF also includes sequence numbers in MD5 authentication to prevent replay attacks (Kurose 399).

In BGP, pairs of routers form Transport Control Protocol (TCP) connections in the clear. Once a connection is established, they send link updates using the portion of the BGP routing table they are allowed to exchange by policy (Rekhter 7). They then exchange link updates as their routing tables change or periodically. Through this process, routers are able to learn which destinations are available through neighboring routers (Kurose 402). Like OSPF, BGP can be configured to use MD5 authentication between router pairs. However, because these routers are often owned by different organizations, the process of establishing, exchanging, and maintaining pre-shared keys is significantly more difficult and is often not implemented. Even if ISPs do use MD5 authentication, there are routing attacks and other problems it cannot protect against such as reset routing protocol sessions (DoS) and incorrect routing information from legitimate or compromised sources. As a result, ISPs have evolved additional methods for protecting their routers and route tables such as filtering and sinkholes (Ulrich 14).

As we've seen, the only security process built into OSPF or BGP is MD5 authentication. As previously discussed, MD5 is considered a flawed algorithm because it is possible for attackers to produce hashes that match multiple hashed values relatively easily. But for use in routing protocols, it is still fairly useful. This is because the time between which a link update is created, hashed, transmitted and then validated is so short

that an attacker is unlikely able to intercept and modify the information effectively. Moreover, MD5 is a fairly fast hashing algorithm and, compared to more sophisticated algorithms, it places a fairly small computational load on the router. Still, the Internet Engineering Task Force (IETF) has stated that MD5 is not strong enough for future use (Bellovin) and has initiated a request for comments to propose future solutions (Behringer). However, these improvements, which were initiated in 2006, are still far from even achieving draft form at the time of this research, possibly because there is no strong perceived need for additional security in these protocols.

Regardless of MD5's strength and appropriateness for use in OSPF and BGP, it is currently sufficient to keep the Internet running. And it does not rely upon any asymmetric encryption algorithms, nor do OSPF or BGP. Therefore, we can also conclude that the main routing protocols in use on the Internet are not vulnerable to a $P = NP$ environment or the use of efficient NP algorithms. In fact, it is likely that advances in efficiency calculations resulting from NP algorithms could actually be used to *improve* the performance of Internet routing algorithms.

4.3.2 Internet Protocol

As discussed, the Internet Protocol does not call for encryption of any kind in its specification. Indeed, RFC 791 for IPv4 is exceedingly old—written in 1981! At that time, Internet security was rarely even thought of. Interestingly a search of RFC 791 for “security” will show that the “security” options within IP specify old DoD classification handling markings to proscribe how information in packets was classified (Postel, “RFC 791” 15).

The lack of security or encryption in IP shouldn't be a huge surprise. The protocol is designed for the efficient transfer of information. As stated in the specification scope, "The internet protocol is specifically limited in scope to provide the functions necessary to deliver a package of bits (an internet datagram) from a source to a destination over an interconnected system of networks. There are no mechanisms to augment end-to-end data reliability, flow control, sequencing, or other services commonly found in host-to-host protocols" (Postel, "RFC 791" 1). Indeed, the designers of the protocol intended for anything beyond simple message transfer to be handled by protocols higher in the stack. Therefore, we can show that IP does not rely upon asymmetric encryption and would not be materially affected by efficient NP algorithms.

Of course, all of the above applies to IPv4 and some might wonder if IPv6 provides an additional level of security. That is a common myth. Generally speaking, IPv6 does next to nothing more for security than does IPv4" (Convery 1). IPv6 was originally intended to include the use of IPsec, but as of RFC 6434, the RFC has softened its language making IPsec just a suggestion (Jankiewicz 17). Since IPsec deployments in both IPv4 and IPv6 are fairly complex, both versions of IP are usually deployed without cryptographic protections of any kind (Convery 2). Due to this fact, for the purpose of this analysis we will generally treat both active versions of IP as distinct from IPsec. Moreover, IPv6 has the same risk profile as IPv4 regarding the issue of $P = NP$.

4.3.3 Internet Protocol Security

IPsec can operate in two modes (or both at the same time): Authentication Header (AH) and Encapsulating Security Payload (ESP). AH provides integrity checks on a packet's headers and data. ESP provides encryption for a packet's data, but does not

specifically protect the headers. However, AH is rarely used because ESP can be used to provide the same level of integrity with or without encryption (Frankel ES1). In fact, the IPsec architecture has gone so far as to make AH an optional component of IPsec implementations (Kent 8). Still, AH includes some compelling characteristics.

AH mainly provides data integrity by computing integrity check values (ICV) over the IP datagram header and data. AH also provides datagram sequence numbering to prevent replays. The sequence numbers are sequential (and therefore predictable) and must be used in conjunction with the ICV values to identify replayed datagrams. AH ICVs are computed with algorithms which are determined by the implementer, but the IPsec architecture recommends several algorithms including AES, MD5, SHA-1, and SHA-256 (Kent 9). Since the ICV algorithms are intended to be either symmetric algorithms or hash functions, the AH portion of IPsec would be unaffected by efficient NP algorithms.

In ESP mode, IPsec acts as a tunnel between two devices and as a boundary at the interface of a device. IPsec allows a great degree of granular control, ranging from a singular encrypted tunnel for all traffic between two gateways to separate tunnels for multiple connections between a pair of hosts (Kent and Seo 9). An administrator must configure IPsec policies at the interface of each device to identify what traffic is subject to the IPsec services. These policies dictate how IPsec handles packets traversing the boundary including what kinds of inbound and outbound connections are permitted. Inbound packets that are part of an established IPsec security association are decrypted and passed up the stack. Inbound packets that aren't part of an IPsec security association

are discarded (Kent and Seo 8). Outgoing packets subject to ESP are typically encrypted using the Internet Key Exchange (IKE) protocol.

Once the administrator builds IPsec policies at both ends of a connection and those respective devices establish a connection they exchange keys via IKE which consists of two phases. The first phase establishes a security association using Diffie-Hellman key exchange (an asymmetric algorithm) to generate symmetric session keys (Mason). These session keys (as well as those derived in phase two) will either be AES or Triple Data encryption Standard (DES) as specified by the RFC (Manral 3). Before the devices perform the key exchange, they must authenticate one another. Typically, if an IKE implementation uses Diffie-Hellman, it will employ a certificate validated by a trusted certificate authority (Kent 36). Pre-shared keys may also be used for authentication in smaller IPsec implementations (Harkins 15). (It is important to note here that the pre-shared keys in phase 1 are not used to derive symmetric session key material—this is always done using a Diffie-Hellman key exchange.)

IKE's second phase uses the symmetric keys established in the first phase to negotiate a security association on behalf of IPsec. It does this by passing nonces back and forth using the original symmetric keys and then uses those nonces to derive new symmetric secret key material. In fact, IKE periodically renegotiates new key material using Diffie-Hellman key exchange so as to limit the amount of data sent under a single key. Moreover, the nonces can also be used like sequence numbers to prevent replay attacks (Harkins 16). The second phase results in two (or more) unidirectional, one inbound and one outbound, secure connections (Harkins 18). These connections constitute the IPsec tunnel over which data is exchanged.

Since ESP is heavily dependent on asymmetric algorithms for both authentication and encryption, it would be vulnerable to attacks with efficient NP algorithms. That being said, it would be time-critical for an attacker to intercept the traffic during the IKE phase one key exchange (those keys act much like an initialization vector for the rest of the IPsec session). If they managed that feat, they would be able to derive the keys used to generate the nonces and the future session keys derived in phase two. If an attacker is only able to intercept traffic after phase two begins, all of the traffic is effectively symmetrically encrypted, including the later Diffie-Hellman exchanges (Harkins 24). However, much like modern attacks on WPA2, it may be possible for attackers to force an IPsec session to terminate. When a security association terminates, all existing keys are discarded and if the session is re-established, it must perform a new IKE phase one and phase two (Mason), thus providing an attacker an opportunity to capture the phase one keys and intercept subsequent traffic.

Therefore, IPsec in ESP mode would most likely be rendered untrustworthy should $P = NP$ and thus not usable for confidentiality, availability, or integrity. The use of IPsec in ESP mode for VPN applications would be seriously jeopardized and would likely need to be discontinued if information on those VPNs is critical to protect. IPsec could still provide some degree of integrity protection, however. As we saw, AH relies on algorithms that would be unaffected by efficient NP algorithms. Where available, administrators interested primarily in integrity versus confidentiality could switch to an AH implementation of IPsec. Unfortunately, AH support is falling by the wayside as discussed above, and many modern IPsec implementations no longer include it.

4.4 Transport Layer Protocols (TCP and UDP)

The protocols at the next OSI layer, the transport layer, begin to add many more options and services beyond the simple transmission of data. Additional options result in more complexity, and it might be reasonable to start seeing a greater reliance on encryption and possibly asymmetric encryption. Oddly, though, there are very few protocols at the transport layer, and the vast majority of applications either use TCP or UDP, which are the two protocols we will evaluate.

TCP is a robust protocol that provides the application layer with, basic data transfer, reliability, flow control, multiplexing, connections, traffic precedence, and congestion control. UDP, on the other hand, only offers multiplexing and basic error detection. While there are over 20 RFCs that govern the TCP architecture (Gont 6-7), a search through each of them revealed that TCP does not use encryption, either symmetric or asymmetric. In fact, neither TCP nor UDP provides any kind of encryption (Kurose 96)

The closest feature that TCP has relating to encryption is the incorporation of hash functions in a few limited applications. The first is the use of a simple hashing function for the selection of ephemeral port numbers (Gont 14). There are some security concerns due to the relative weakness of that hashing function. As long ago as 1996, there have been proposals to replace the TCP hash function with MD5 (Gont 21); however, few TCP implementations have implemented that measure. This lack of concern over the hash function, probably because it is typically only used to obfuscate the initial sequence number rather than for signing TCP data, shows that even this small use of cryptography isn't critical to TCP's functionality.

The other area in which TCP uses hashing is an optional mechanism for authenticating segments using the MD5 algorithm. The purpose of this feature is to protect against forged TCP segments (Gont 43). However, since this option requires that both devices in a connection have a pre-shared password, it is seldom used.

TCP and UDP were developed when asymmetric cryptography was in its infancy and when the Internet was a much less hostile place. The only cryptographic function either protocol uses is hashing, and only TCP uses that. Since hashing does not use NP functions neither TCP nor UDP would be affected by efficient NP algorithms.

4.5 Presentation Layer Protocols (SSL and TLS)

Presentation layer protocols provide information formatting between the application and transport layers. Often this involves encoding data for transmission, compression, serialization, and encryption. These protocols help standardize transmission formats and ensuring information is ordered and meaningful, but do not actually perform transmission functions. The two common presentation layer security protocols are SSL and TLS.

The TLS protocol has two main components: the handshake protocol and the record protocol. The handshake protocol is responsible for authenticating the parties in the connection and then negotiating a key exchange. The handshake protocol then passes the selected keys, ciphers, hash, and compression algorithms to the record protocol which exchanges encrypted application data (Dierks 3). A graphical representation of the TLS message sequence is shown below in Figure 3.

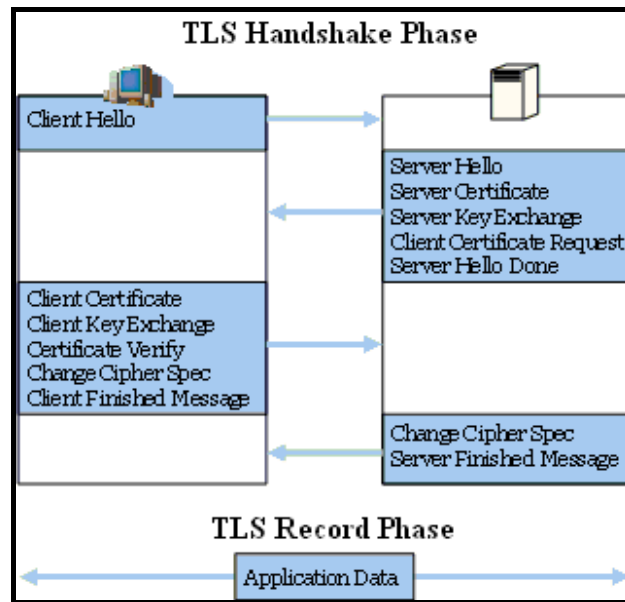


Figure 3: TLS Message Sequence (Microsoft)

The handshake protocol begins when the connection is started and uses an asymmetric algorithm to authenticate the parties in the exchange. Authentication in TLS is optional, but is generally required for at least one of the peers (Dierks 5). In a normal HTTPS web browsing session, only the server is authenticated because typically it is only the customer of the web site that needs to validate that the server is who it claims to be—the server will verify the customer with a password after the TLS session begins. For other applications, such as SMTP over TLS, both parties would need to authenticate.

TLS authentication requires an established public key infrastructure. This is because TLS uses identity certificates, typically X.509 certificates, as required by the specification (Dierks 48). These identity certificates are issued and validated by a mutually-trusted, third-party Certificate Authority (CA) using either RSA or another asymmetric algorithm to cryptographically sign them. The identity certificates consist of a copy of the certificate holder’s public key, a hash of the public key that has been signed

with the CA's private key, and a few other pieces of information such as a validity period, serial number, etc. The certificate is intended to prove that the certificate holder owns the corresponding private key. This is validated first by the decrypting the private key signature of the CA on the certificate with the CA's published public key (thus showing that it was indeed the CA that validated the certificate holder's public key). Then the certificate holder's ability to decrypt traffic encrypted with the public key in the certificate proves that they hold the private key corresponding to the public key.

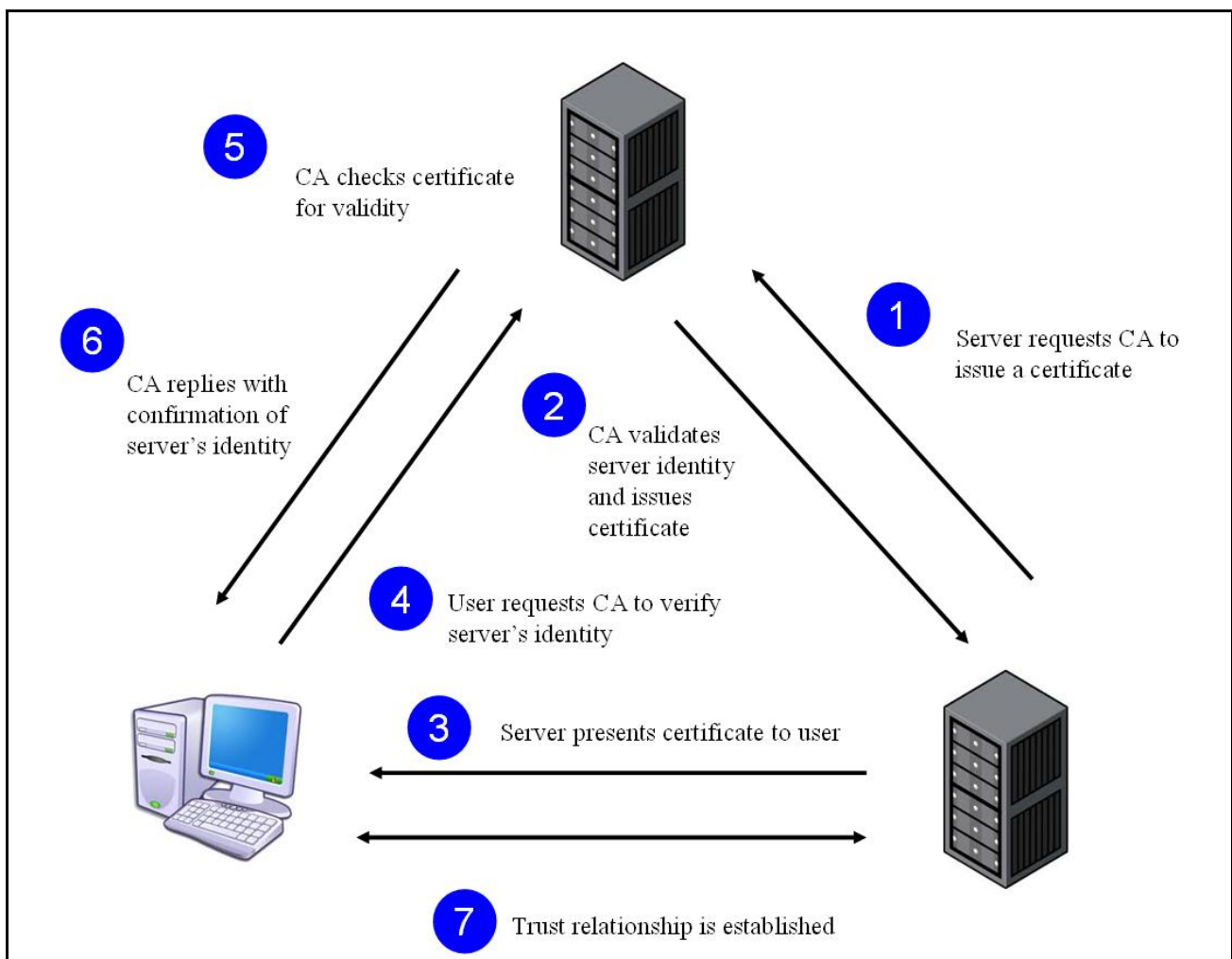


Figure 4: TLS Authentication Using Certificate Authority

The use of certificates and public key infrastructure during authentication is the first component in the TLS protocol that is of concern should efficient NP algorithms be found. If so, it could be relatively simple to derive the CA's private key from the public key and sign falsified identity certificates. This would allow anyone to impersonate banks or other e-commerce sites and capture confidential information. Moreover, an attacker could present a legitimate certificate signed by a CA and then use efficient NP algorithms to derive the certificate holder's private key. This would also allow the attacker to masquerade as a certificate holder. Worse, since certificates are fairly long-lived (on the order of a year or more), attackers would have large amounts of time to break key pairs used to sign certificates, making this a likely and lucrative target. Therefore, efficient NP algorithms would completely undermine the authentication protocols within TLS.

Following authentication, TLS performs a key exchange using either RSA or DH algorithms (Dierks 75). During this phase, the parties agree on an asymmetric key exchange algorithm and a symmetric algorithm to use for bulk data encryption (typically AES or RC4) (Dierks 26). Then each party generates a public / private key pair and exchanges public keys. This key exchange is the second concern with TLS. An attacker armed with efficient NP algorithms could intercept this key exchange and potentially derive the private keys, decrypt the messages, and extract the session keys that will be exchanged in the next step. Hence the attacker could intercept and read all traffic exchanged during the TLS session, thereby defeating confidentiality and integrity of TLS.

Following the key exchange, the parties exchange information using their asymmetric keys which they use to generate their symmetric session keys using the agreed upon symmetric algorithm. From this point forward, all traffic passed in the TLS session will be symmetrically encrypted. At this point, TLS begins the record protocol phase and exchanges application data. Since the record protocol uses symmetric encryption, it is not directly vulnerable to attack with efficient NP algorithms. However, as we saw, if the session keys are captured during the asymmetric key exchange, even the record protocol could be compromised.

In the event that asymmetric cryptography should be rendered transparent, the TLS protocol would be rendered unable to guarantee the authentication, confidentiality, and integrity services it provides. TLS would still function and connections using the protocol would still be able to pass information, but confidentiality and integrity would no longer be guaranteed. Due to TLS's widespread usage in e-commerce, this is a particularly troubling finding. Unlike some other applications that use asymmetric algorithms, the direct financial link to TLS is very clear. Moreover, the e-commerce targets are very obvious and easy to locate. This is likely the first protocol that would be attacked with efficient NP algorithms and publicly released exploits. It would be critical for users to discontinue relying on TLS and replace it with some form of symmetric encryption and key exchange.

4.6 Application Layer Protocols

The application layer is the closest to the end user and provides an interface that the user can directly interact with and access information on the network. The application layer will manage identification of communication partners and determining

resource availability. Some applications manage their own encryption, but most rely upon lower layers to provide that service.

4.6.1 Secure Shell

Typical of many application layer protocols, SSH uses a hybrid encryption mechanism. It uses asymmetric algorithms to initiate secure communication and then negotiates a pair of symmetric keys for bulk data transfer. The first step in starting a SSH session is the establishment of a TCP connection between two devices. Once the connection is made, the two devices initiate a key exchange method. The key exchange method specifies how mutual authentication is performed and subsequently how one-time session keys are generated for encryption (Ylonen, "RFC 4253" 13). The SSH key exchange protocol as defined in IETF RFC 4253 requires both Diffie-Hellman group 1 and group 14 key exchange methods, although additional methods may be defined. Regardless, any key exchange method would by nature have to be an asymmetric algorithm according to the architecture (Ylonen, "RFC 4253" 13).

After the two devices authenticate, both systems agree on a session key. This key is symmetric and temporary (i.e. is discarded after the session). The SSH protocol architecture does not require a specific symmetric algorithm for the creation of session keys. It suggests several ciphers including 3DES, ARCFOUR, twofish, serpent, blowfish and AES (Ylonen, "RFC 4251" 13). Indeed, most implementations of SSH allow users to choose from some subset of the above ciphers (OpenSSH.com)(SSH Comm Sec). Once the session keys are selected, the devices initiate a secure tunnel and begin data communication.

Following the key exchange, the actual data communication would not be impacted by efficient NP algorithms since it uses symmetric encryption. However, since the key-exchange process uses asymmetric algorithms, that session could be intercepted and with sufficiently strong NP algorithms, and the key exchange could be intercepted. This would render the entire SSH session transparent, allowing attackers to capture router, firewall and server passwords passed during the SSH session. If efficient NP algorithms existed, and if tools could then be developed to attack SSH sessions, this could be worrisome for Internet security. In a worst case scenario, critical routing and security infrastructure could be compromised via SSH.

In terms of functionality, SSH would continue to work, as would the devices that use it. However, to reduce the risk of those devices being hijacked, administrators would need to immediately stop using SSH to remotely log onto infrastructure devices or risk compromising credentials to attackers. Alternatively, infrastructure devices would need to be protected with another layer of security either involving some out-of-band communication path or a symmetrically encrypted VPN that does not rely on asymmetric encryption. For some devices, this may not be possible immediately—they may require configuration changes for business or operational needs and local logon may not be an option.

4.6.2 Domain Name System

Like many of the other old, foundational protocols reviewed thus far, DNS does not use any form of encryption. A review of RFCs 1034 and 1035, which describe the DNS architecture, revealed no mention of encryption or security. And while there have been numerous flaws discovered and corrected in DNS (as evidenced by a couple dozen

updates to the RFC (Mockapetris 1)), DNS is a remarkably robust system. “To date, there hasn’t been an attack that has successfully impeded the DNS service” (Kurose 146). Thus, the development of efficient NP algorithms would not have an effect on DNS.

There are, however, proposed security extensions to the DNS specification. DNS Security, or DNSSEC, was published in RFC 4033 in March of 2005. It is designed to provide origin authentication and integrity assurance services for DNS data (Arends 6). The goal of DNSSEC is to protect clients from forged DNS responses such as those created in DNS cache poisoning. In DNSSEC, authoritative servers digitally sign lookup records using asymmetric algorithms such as RSA (Arends 13). When a client receives a resource record from a DNS query, they can then authenticate the record.

Since DNSSEC relies on asymmetric signatures for its authentication of resource records, DNSSEC would be vulnerable to efficient NP algorithms. Attackers could derive the private keys authoritative DNS servers use to sign records and use them to falsify signed records. This is however, a fairly minor risk because, despite the 2003 US National Security Strategy to Secure Cyberspace making DNS security a priority, very little has been done to deploy DNSSEC widely.

The main impediments to DNSSEC adoption are cost of deployment, lack of centralized authority over deployment, and a lack of registrant demand (Ozment 8). This lack of demand is likely because DNS functions without DNSSEC, and most attacks can be addressed by configuring DNS servers in accordance with the most current specifications (Kurose 146). Therefore, even if DNSSEC were to become widely deployed, it would be an addition to a fairly robust system. Even if efficient NP

algorithms were used to attack DNSSEC implementations, it would very likely not cause a significant service interruption on the Internet as a whole.

4.6.3 Electronic mail

In its simplest form, e-mail works by an e-mail client agent making a connection to an e-mail server (using IMAP, POP, or even HTTP) and either requesting mail or pushing email to send to another user. The server then replies with messages it has received for the user, or accepts the outgoing email. Once the server has outgoing email, it uses SMTP to contact the destination e-mail server (or a mail relay) and transfers the message via SMTP. This step most likely will also require a DNS lookup for the MX (mail) record of the destination mail server (Kurose 124-132).

There are numerous RFCs that govern how these different e-mail protocols work. RFC 5321 handles SMTP, RFC 1225 governs POP version 3, and RFC 3501 (along with a host of extensions that do not handle encryption) specifies IMAP. A review of these protocols (and their extensions) reveals that none of them contain inherent security provisions for mail encryption. In fact, the SMTP specification says, “SMTP mail is inherently insecure” (Klensin, “RFC 5321” 74).

The SMTP RFC goes on to state that, “Real mail security lies only in end-to-end methods involving the message bodies, such as those that use digital signatures (e.g., Secure/Multipurpose Internet Mail Extensions (S/MIME) or Pretty Good Privacy (PGP)).”(Klensin, “RFC 5321” 74). These methods involve encrypting the e-mail contents prior to sending the message to POP, IMAP, or SMTP. However, this could be problematic in a $P = NP$ environment because both S/MIME and PGP are public-key

systems that use asymmetric algorithms: RSA or DH for S/MIME, and RSA or Elgamal for PGP (Callas 61) (Ramsdell 4).

Since these e-mail encryption methods are not directly included in the e-mail specifications, how often is this suggestion followed? A study by Forrester Research in 2011 indicated that 33% of all businesses use some form of e-mail encryption on top of the basic e-mail protocols (Kindervag 7). They found that the motivating factor behind many of these implementations were compliance initiatives put in place by legislation such as the Health Insurance Portability and Accountability Act, Payment Card Industry Act, and the Health Information Technology for Economic and Clinical Health Act, and (Kindervag 6). Still, corporate e-mail encryption, whether S/MIME, PGP or another, generally requires a significant public-key infrastructure which can be a significant expense. This is probably why most private or personal email users do not routinely use e-mail encryption.

Regardless, the confidentiality and integrity of these public-key email systems would be at risk of data leakage should efficient NP algorithms be developed. It would be possible for an attacker to intercept an encrypted e-mail message, request the recipient's public key, derive the recipient's private key, and decrypt the message. This means that companies using encrypted email to transmit customer privacy information or sensitive business information would need to seek alternatives to keep that information confidential and unaltered. Again, however, we find that the e-mail systems would still function and be available.

Beyond encrypting the message body, POP, IMAP and SMTP all have provisions to secure the underlying TCP connection with SSL or TLS (Crispin 92) (Klensin,

“RFC 5321” 74) (Newman 5). As demonstrated in the SSL/TLS section, these transport layer protocols provide connection-level confidentiality and integrity. Using TLS to connect an IMAP or POP client works well to protect e-mail traffic. However for SMTP, TLS only authenticates from the origination server to the next server in the chain. A chain of relays and servers present opportunities for even TLS encrypted e-mail to be intercepted (Klensin, “RFC 5321” 74). Moreover, this research has shown that TLS would be weakened or made completely unsecure against efficient NP algorithms.

Presently, e-mail security is fairly weak and would function as it does today should P be proven equal to NP or asymmetric encryption be weakened in some other fashion. For the average email user, very little would change. However, those more secure implementations that rely upon public-key infrastructures or TLS/SSL for security could see a reduction in the confidentiality.

4.6.4 Web and HyperText Transfer Protocol

HTTP defines how web clients (typically browsers) and web servers interact and transfer files. It is a client-server protocol that functions in a request-response mode. In the very simplest terms, a client makes a TCP request to the server which replies by sending the requested file. Originally, HTTP was intended to serve up Hypertext Markup Language (HTML) files, but as a file-transfer protocol, it can also be used to transfer multimedia, scripts, and various other file formats. On the client side, these files are interpreted and displayed by a browser or other application.

Beyond a very basic form of authentication, the HTTP specification does not provide any form of confidentiality or integrity (Fielding 70). HTTP provides an optional challenge-response authentication mechanism which can be used by a server to challenge

a client request and by a client to provide authentication information. However, that authentication scheme is not a secure method of user authentication because the challenge-response is transmitted in cleartext (Franks 18).

Since HTTP passes data in the clear over the Internet, the increased use of HTTP to pass sensitive data motivated users to develop methods to secure that traffic (Rescorla 2). This was the motivation for HTTP Secure (HTTPS). HTTPS is not actually a protocol or an internet standard (Rescorla 1), instead it is common practice in which HTTP connections are tunneled over TLS or SSL. In fact, it was most likely the prevalence of HTTP and the need to secure HTTP traffic that propelled SSL and TLS to popularity. The original version of SSL was incorporated into the Netscape Web Browser in 1994 to secure HTTP. Moreover, the practice of securing an otherwise unsecured application layer protocol using TLS is fairly common and this analysis is representative of those other protocols.

Most HTTPS implementations use a different default port for HTTPS connections (port 443) as compared to HTTP (port 80). The use of different ports is merely a convention rather than a requirement; any port can initiate an HTTPS connection. When a client connects to an HTTP port and sends a TLS ClientHello message, that prompts the server to begin a TLS handshake as described previously.

HTTPS is the primary method used to secure web traffic, and is predicated on TLS; therefore, HTTPS suffers from all of the same vulnerabilities as TLS. This means that, with efficient NP algorithms, the security granted by HTTPS could be broken and many of the e-commerce applications provided over HTTP and secured with HTTPS

would be vulnerable to eavesdropping. HTTPS would continue to function but it may not be much more secure than just unencrypted HTTP.

Another, less known method to secure HTTP traffic, called Secure-HTTP (S-HTTP) also exists. Like HTTPS, S-HTTP is not a standard in its own right, but rather it is an “experimental” (i.e., not widely adopted) process for securing HTTP messages (Rescorla and Schiffman 1). S-HTTP was also originally defined in the mid-1990s, however the primary web browser developers (Microsoft and Netscape) supported HTTPS making it the de facto standard over S-HTTP. S-HTTP anticipates that users would use public key certificates, but the designers made an effort to avoid presuming a particular trust model (Rescorla and Schiffman 3). Therefore, S-HTTP supports some symmetric key-only operation modes that make it interesting in light of a $P = NP$ discussion.

S-HTTP works much like HTTPS but rather than wrapping the entire HTTP session in a TLS tunnel, S-HTTP makes use of additional tags within the body of the HTTP message to encrypt selected items such as POST fields (Rescorla and Schiffman 5). An S-HTTP sender prepares messages encrypting portions of the cleartext message and then specifying what key material was used in optional S-HTTP headers. The recipient then parses the S-HTTP headers to discover what cryptographic transformations were applied to the message and then recovers the original message by decrypting those fields.

S-HTTP supports multiple key management mechanisms including password-style manually shared secrets and public key exchange. This includes pre-shared symmetric session keys (in an earlier transaction or out of band) in order to send

confidential messages to those who have no public key pair (Rescorla and Schiffman 6). This is significant because it means that an already devised protocol could be used in conjunction with a symmetric key distribution infrastructure to replace HTTPS should it be rendered weak by efficient NP algorithms.

4.7 Protocol Analysis Summary

The objective of this protocol analysis was to determine which of the critical Internet protocols rely upon asymmetric encryption and, for those that do, to determine how losing that encryption would impact their functionality. A review of the selected critical protocols shows that a significant portion does not incorporate encryption of any kind. These protocols include Ethernet, routing protocols, IP, UDP, and DNS. Further, other critical protocols only rely upon symmetric encryption or hash functions which make them relatively immune to exploitation from efficient NP algorithms. Those include bulk encryption, WPA2 (without 802.1x), and TCP. The remaining protocols, WPA2 (with 802.1x), IPsec, SSH, SSL/TLS, email and web use asymmetric encryption in a significant part of their functionality. Results of this research analysis are summarized in Table 5.

This analysis shows that several common Internet protocols rely on asymmetric algorithms and could be affected by efficient NP algorithms. The main risk to these protocols is that the information they seek to protect with asymmetric encryption would be jeopardized (i.e., loss of data confidentiality and integrity). Despite this, each of the protocols would still be able to function and support information transfer as intended in their specifications (i.e. availability is maintained). In many cases (such as e-commerce, corporate VPNs, infrastructure administration, and e-mail transmission of sensitive

information), the loss of confidentiality and integrity would most likely make continued transmission undesirable. In these cases, we can expect that by virtue of the fact that data would no longer be transmitted due to policy, availability could also suffer.

Table 5 – Summary of Key Internet Protocols and Their Dependency on Asymmetric Cryptography

Protocol	OSI Layer	Encryption Employed	Optional Encryption	Affected by P = NP	Impacts
Bulk Encryption Techniques	Data Link	Symmetric (Various)	N/A	No	None (assuming protected device management)
Ethernet	Data Link	None	N/A	No	None (assuming protected device management)
WPA2 (without 802.1x)	Data Link	Symmetric (AES)	N/A	No	None
WPA2 (with 802.1x)	Data Link	Symmetric (AES)	Various hashing or asymmetric (TLS)	Yes	If EAP-TLS, TTLS, or PEAP are used, WPA authentication could be broken allowing eavesdropping or unauthorized access
Routing Protocols (OSPF, BGP)	Network	None	Hashing (MD5)	No	None (assuming protected device management)
IP	Network	None	IPsec	No	None
IPsec (AH Mode)	Network	Symmetric (Various) or Hashing (Various)	N/A	No	None
IPsec (ESP Mode)	Network	Asymmetric (DH) and Symmetric (AES)	Additional DH rounds	Yes	IPsec sessions could be efficiently broken jeopardizing many VPN implementations and remote logon sessions used to administer infrastructure devices
TCP	Transport	Simple Hashing	N/A	No	None
UDP	Transport	None	N/A	No	None
SSL / TLS	Presentation	Asymmetric (RSA, DH, DSS, etc.) and Symmetric (AES, RC4, etc.)	SSL / TLS allows negotiation of ciphers at connection establishment	Yes	TLS and SSL sessions used to secure e-commerce and other applications could be intercepted rendering them transparent
SSH	Application	Asymmetric (DH) and Symmetric	Asymmetric (protocol allows	Yes	SSH sessions could be intercepted allowing attackers to capture

		(AES)	optional public key types)		remote logon passwords used to administer infrastructure devices
DNS	Application	None	DNSSEC (RSA signatures)	No	None
E-mail (SMTP, POP, IMAP)	Application	None	SSL / TLS	Application dependent	Limited – e-mail functionality would not be directly affected by P = NP. However, since all modern mail protocols rely upon SSL or TLS for confidentiality and integrity, secure email would be jeopardized
Web and HTTP	Application	None	SSL / TLS	Application dependent	Limited – file transfer and web browsing functionality would be unaffected. However, since HTTPS relies upon SSL or TLS for confidentiality, integrity, and in some cases authentication, secure browsing and file sharing that rely upon these extensions would be in jeopardy.

4.8 Timeline Analysis

Not all proofs are created equal. Nor are all algorithms for that matter. One of the most overlooked problems with the doomsayers’ arguments about P = NP meaning the demise of the Internet is that they make an implied assumption that “efficient” NP algorithms would be really efficient. Many of them also imply that a P = NP proof would be constructive, which might not be the case. Both of these assumptions could be false, and if so, a P = NP proof might not have any significant consequences at all! There are really a range of possible outcomes depending on how a P = NP proof might be formulated and how efficient NP algorithms are once they are found.

Mathematical proofs come in two main flavors: constructive and non-constructive. A constructive proof provides a demonstration for the existence of a mathematical object by creating or providing a method for creating such an object. A non-constructive proof shows that if a certain proposition is false, a contradiction exists and thus the proposition must be true. If a $P = NP$ proof takes the form of a constructive argument, it would show not only that $P = NP$, but it would also provide an algorithm for solving a NP problem in P time. However, it is possible that a $P = NP$ proof might be non-constructive. Such a proof would indicate that $P = NP$ and that algorithms must exist, but it might not actually yield any algorithms for solving NP problems better than what we have today (Cook, “P vs. NP” 10).

If a $P = NP$ proof turns out to be non-constructive, the world will have a warning that efficient NP algorithms are imminent. How imminent is hard to predict, but with the stakes for finding them in the billions if not trillions of dollars, it is quite likely that a lot of smart people will begin trying even harder to produce such an algorithm. Still, it is likely that engineers and computer scientists would have fair warning to develop and deploy alternate encryption and defensive systems to shore up asymmetric weaknesses. In this scenario, it is possible that $P = NP$ would look much like the Year 2K event. By the time attacks became available for asymmetric encryption, most of the systems that used it will have been updated.

If a $P = NP$ proof is a constructive proof, then it would also provide a method for solving NP problems. In this case, the amount of lead time engineers and computer scientists will have to react depends on how quickly exploits can be fielded to take advantage of the new weakness in asymmetric encryption. Unfortunately, this would be

a bad situation. In 2006, Trend Micro published a report showing how new malware is typically released in a matter of just days after the announcement of a vulnerability (Trend Micro). They attributed the dramatically short timeline to the fact that malware has become big business. Malware authors need to get their code into the wild as soon as possible to beat system administrators to a patch and to beat competing malware writers to easy targets. As a result, malware writers have evolved sophisticated software development processes including online collaboration, peer reviewed code, modular code, and shared code libraries. It is hard to predict whether or not an efficient NP algorithm would be significantly different in implementation than say a buffer overflow exploit. In any case, if malware writers and even sponsored nation states have access to such an algorithm, a worst case expectation for seeing malware fielded would be a matter of a days or weeks at most.

Still, even a constructive proof that $P = NP$ may not produce algorithms that are very useful in practice. As discussed in the explanation of P and NP complexity class problems, there are polynomial algorithms that aren't that much more practical to run than an exponential problem. For example, N^{9999} is not that much more practical to run than one that runs in 1^N time. In theory it is more efficient, but in practice neither algorithm is going to crack asymmetric encryption in a fashion that will cause network security to be turned on its ear.

There is therefore a range of possible scenarios that might come to pass if P is proven to equal NP. In a best case scenario, a non-constructive proof would give us reaction time before algorithms are developed. In a worst case scenario, a constructive proof could provide very efficient algorithms that just need to be implemented. A middle

ground scenario would be for a constructive proof to yield algorithms that were not very practicable.

V. Conclusions and Recommendations

5.1 Conclusions

The Internet is a complex system of systems. It is neither a simple nor straightforward task to show that “the Internet” is or is not subject to massive failures due to the development of a new type of algorithm. However, it is possible to analyze the major protocols that comprise the Internet and allow it to perform its main function: to transfer information. After looking at between 13 and 19 (depending on how you choose to count them) of the core Internet protocols, it becomes possible to draw some conclusions about how efficient NP algorithms *might* affect the overall system. Simply stated, it appears claims that $P = NP$ would disable the Internet are incorrect. Moreover, claims that $P = NP$ would end e-commerce are probably overstated as well.

If P should be proven to equal NP , the technologies that rely upon asymmetric cryptography would still function. In terms of the CIA model, availability would be preserved. The danger from efficient NP algorithms would be to data integrity and confidentiality. Unauthorized parties could potentially gain access to the data or communications channels protected by obsoleted asymmetric cryptography. From there, they could possibly alter or copy data or send commands. But without hostile actors taking action, those protocols would continue to work as they do today.

Most of the protocols examined in this research were found to be insensitive to the development of efficient NP algorithms. Of the protocols examined, these included: bulk encryption, Ethernet, WPA2 (without 802.1x), routing protocols, IP, UDP, TCP, and DNS. In fact, some of these protocols, namely the routing protocols and DNS may actually benefit from efficiencies imparted by efficient NP algorithms. The unaffected

protocols represent some of the most foundational protocols that comprise the Internet. The fact that these protocols would not be negatively affected by efficient NP algorithms indicates strongly that predictions stating that $P = NP$ would indicate that the predicted demise of the Internet is overstated.

Those protocols that would be affected by $P = NP$ as were TLS, SSL, SSH, IPsec, and WPA2, all of which are based on asymmetric cryptography. Most of these could be replaced by symmetric-based protocols discussed in this paper or some other alternative symmetric technology. For example, IPsec implementations could be replaced with hardware bulk encryptors. The use of SSH for remote administration could be abandoned in favor of logging into devices directly or over out-of-band administration networks. WPA2 implementations that rely upon EAP protocols could be replaced with pre-shared keys or Kerberos infrastructures. Admittedly, these solutions may involve the deployment of additional hardware and would not be convenient in many cases. But for those applications that are critical to operations, organizations will need to make the transition. And for those applications which are not critical to operations, organizations will stop performing that task or they may just continue using the weakened asymmetric algorithms until a better option comes along. In the end, it would come down to risk management.

Technologies like PPTP and Wireless Equivalent Privacy (WEP) provide interesting and pertinent cases that demonstrate organizations' propensity for continuing to use weakened cryptography. Both of these protocols have been known to be cryptographically faulty for over a decade. Yet, both protocols are also still commonly used, both in personal and enterprise networking. This is because, while there is a risk

that an attacker could easily compromise the security of these protocols, they still function as intended. The users of PPTP and WEP are either unaware of the risks, the actual instance of that risk being acted upon by a threat is small, or the consequence of losing data is relatively small compared to the cost of using a different protocol.

In the case of SSL and TLS, the stakes are somewhat higher, and the potential for more widespread disruptions do exist. However, as this paper showed, a $P = NP$ proof may not obviate these protocols overnight. If the $P = NP$ proof is non-constructive, industry may have sufficient warning to develop new infrastructures that rely upon symmetric encryption. Moreover, if efficient NP algorithms only slightly improve an attacker's ability to crack asymmetric encryption, it may be sufficient to increase key length on the TLS protocols to make them sufficient for e-commerce.

In the worst possible case, a constructive $P = NP$ proof would provide extremely efficient NP algorithms, but it is still unlikely that industry and the Internet will not adapt to the disruption. The world economy has become dependent on e-commerce and moving money electronically. Hundreds of billions of dollars move across the Internet annually, and business is not going to accept a return to paper checks and payments via "snail mail."

Alternative symmetric technologies to asymmetric infrastructures have existed since the 1980s (Kerberos Consortium). Wide-scale deployments of these technologies did not occur because asymmetric technologies appeared at the same time. Asymmetric technologies were much cheaper and simpler to deploy. Realistically, symmetric infrastructures probably could not have fueled the stratospheric rise of e-commerce like asymmetric technologies did. Yet, now that e-commerce and the Internet have achieved

today's level of integration into corporate, government, and civilian life, the world will not return to old ways of doing business. Rather than accept the destruction of the Internet or e-commerce, the world would obviously make the transition to symmetric infrastructures to perform the tasks done by asymmetric algorithms today. The transition would not be easy or cheap, but with the stakes being what they are, it would happen.

5.2. Recommendations for Future Research

Probably the most useful piece of future research related to this topic would be for someone to develop a practicable symmetric key infrastructure to replace existing asymmetric key infrastructures. There are foundations for such a system today like Kerberos. But fielding asymmetric encryption has been so much simpler and cheaper and so asymmetric infrastructures have dominated. The Kerberos authentication protocol developed by MIT is a likely place to start. It was developed when asymmetric encryption was largely proprietary and therefore utilizes only symmetric encryption. An Internet-wide Kerberos-like symmetric key authentication system could replace asymmetric infrastructures.

However there are several questions that must be answered and hurdles that must be overcome to implement such a system. For instance, can Kerberos scale to the size it would need to in order to replace existing asymmetric infrastructures? The Kerberos Consortium indicates that it can be made hierarchical and scale similarly to DNS (Kerberos Consortium). Is that true? How would a key management server securely store the multitude of keys it would have to manage? Also, with Kerberos (or really any symmetric infrastructure) on such a scale, how do you distribute keys to all of the users that wish to perform e-commerce transactions? Could keys be distributed on smart cards

via the mail like credit cards? How would a browser and any other associated hardware and software protect the keys on a client system? Could S-HTTP be used in conjunction with such a system to replace HTTPS?

Depending on how these questions are answered, I can envision a system in which companies that currently issue certificates (i.e. certificate authorities) expand to generate and issue symmetric keys to users and businesses. Perhaps these companies would partner with credit card issuing companies to establish customer identities and to send key tokens via mail or another channel. Most likely these companies could charge subscription fees for identity and key management. Then when a customer wishes to initiate secure communication with an e-commerce entity, they could use the CA as a trusted third party in an Internet-wide Kerberos implementation. (Likely there would be multiple competing CA services.) A system that provided such a replacement symmetric infrastructure and answered the above questions would be worth billions in a post-P=NP world.

Bibliography

- Aaronson, Scott. Lecture on Physics.
<http://www.scottaaronson.com/democritus/lec6.html>
- Aaronson, Scott. "3 Questions: P vs. NP." Interview by Larry Hardesty. MITnews,
17 Aug 2010.
- Aaronson, Scott. "Reasons to Believe." Shtetl-Optimized. 4 Sept 2006.
<<http://www.scottaaronson.com>>
- Allender, Eric, Michael C. Loui, and Kenneth W. Regan. Complexity Classes.
www.rutgers.edu. Rutgers University. 28 Mar 2012.
<<http://ftp.cs.rutgers.edu/pub/allender/ALRch33.pdf>>
- Allied Telesis. 802.1x White Paper. Tech. 2006. 1 May 2012.
<http://www.alliedtelesyn.com/media/pdf/8021x_wp.pdf>
- Arends, R., R. Austein, M. Larson, D. Massey, and S. Rose. "RFC 4033: DNS Security Introduction and Requirements." Internet Engineering Task Force (IETF),
Mar 2005. <<http://tools.ietf.org/html/rfc4033>>
- Behringer, M., Cisco Systems Inc. BGP Session Security Requirements (Draft). IETF,
10 Jul 2008. <<http://tools.ietf.org/html/draft-ietf-rpsec-bgp-session-sec-req-01>>
- Bellovin, S. and A. Zinin. RFC 4278: Standards Maturity Variance Regarding the TCP MD5 Signature Option and the BGP-4 Specification. IETF, Jan 2006.
<<http://tools.ietf.org/html/rfc4278>>
- Callas, J., L. Donnerhacker, H. Finney, R. Thayer, and D. Shaw. "RFC 4880: OpenPGP Message Format." IETF, Nov 2007. <<http://tools.ietf.org/html/rfc4880>>
- Cameron, James. "Why Not Use PPTP?" Letter to Sourceforge.net Mail List. 8 Oct 2005.
Sourceforge Poptop. 27 Apr 2012. <<http://poptop.sourceforge.net/dox/protocol-security.phtml>>
- Caesar, Matthew, and Jennifer Rexford. BGP Routing Policies in ISP Networks.
Princeton University, 2006. 2 May 2012.
<<http://www.cs.princeton.edu/~jrex/papers/policies.pdf>>
- Certes Networks. "Ethernet Encryption Made Easy." 1 May 2012.
<<http://www.certesnetworks.com/securitysolutions/ethernet-encryption.html>>
- Cisco Systems, Inc. "Cisco Security Notice: Dictionary Attack on Cisco LEAP Vulnerability." www.cisco.com. 3 Aug 2003.
<<http://www.cisco.com/warp/public/707/cisco-sn-20030802-leap.shtml>>

- Cisco Systems, Inc. "Internetworking Basics." Cisco DocWiki. Cisco Systems, Inc. 24 Apr 2012. <http://docwiki.cisco.com/wiki/Internetworking_Basics#OSI_Model_Physical_Layer />
- Convery, Sean, and Darrin Miller. "IPv6 and IPv4 Threat Comparison and Best Practice Evaluation." White Paper. Cisco Systems, Inc., 2004.
- Cook, Stephen A. "The Complexity of Theorem Proving Procedures." University of Toronto, May 1971. <<http://www.cs.toronto.edu/~sacook/homepage/1971.pdf>>
- Cook, Stephen A. "The P Versus NP Problem." www.claymath.org. Clay Mathematics Institute, Cambridge. 21 Jan 2012.
- Crispin, M. "RFC 3501: Internet Message Access Protocol - Version 4 Revision 1." IETF, Mar 2003. <<http://tools.ietf.org/html/rfc3501> >
- Dell. "Carrier Ethernet." Force10 Networks. 27 Apr 2012. <http://www.force10networks.com/solutions/carrier_ethernet.asp>
- Dierks, T., and R. Rescorla. "RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2." IETF, Aug 2008. <<http://tools.ietf.org/html/rfc5246>>
- Diffie, Whitfield and M. E. Hellman. New directions in cryptography. IEEE Transformation Information Theory, IT-22:644-654, Nov 1976.
- Fielding, R., J. Gettys, J. Mogul, et al. "RFC 2616: Hypertext Transfer Protocol – HTTP/1.1." IETF, Jun 1999. <http://tools.ietf.org/html/rfc2616>
- Fortnow, Lance. "The Status of the P Versus NP Problem." Communications of the ACM 52.9, 2009. 14 Mar 2012. <<http://cacm.acm.org/magazines/2009/9/38904-the-status-of-the-p-versus-np-problem/fulltext>>
- Fortnow, Lance, and Steve Homer. Proceedings of Conference on Computational Complexity, University of Aarhus, Denmark. www.computationalcomplexity.org, 7 Jul 2003. 11 Feb 2012. <<http://people.cs.uchicago.edu/~fortnow/beatcs/column80.pdf>>
- Franks, J., P. Hallam-Baker, J. Hostetler, et al. "RFC 2617: HTTP Authentication." IETF, 2 Jun 1999. <<http://tools.ietf.org/html/rfc2617>>
- Frankel, Sheila, Karen Kent, Ryan Lewkowski, et al. "Guide to IPsec VPNs." Special Publication 800-77. National Institute of Standards and Technology, Dec 2005.

- Gollakota, Shyamnath, and Dina Katabi. "Physical Layer Wireless Security Made Fast and Channel Independent." Diss. Massachusetts Institute of Technology. 2011. 24 Apr 2012. <<http://people.csail.mit.edu/gshyam/Papers/ijam.pdf>>
- Goldreich, Oded. "Foundations of Cryptography." Cambridge University Press, New York, 2001.
- Goldwasser, Shafi, and Mihir Bellare. "Lecture Notes on Cryptography." Cryptography and Cryptanalysis. Massachusetts Institute of Technology, Cambridge MA. Jul 2008. Lecture.
- Gont, F. "Security Assessment of the Transmission Control Protocol (TCP)." IETF Network Working Group Draft. Feb 2009. <<http://tools.ietf.org/html/draft-gont-tcp-security-00>>
- Grimes, John G. National Policy Governing the Use of High Assurance Internet Protocol Encryptor (HAIPE) Products. Committee on National Security Systems. Feb 2007.
- Harkins, D. and D. Carrel. "RFC 2409: The Internet Key Exchange." IETF, Nov 1998. <<http://tools.ietf.org/html/rfc2409>>
- Hartmanis, J. Godel, Von neumann and the P=NP problem. In Current Trends in Theoretical Computer Science, pages 445-450. World Scientific Press, New York, 1986.
- "IEEE 802.3 ETHERNET." LAN/MAN Standards Committee (Project 802). The Institute of Electrical and Electronics Engineers, Inc., 26 Dec 2008.
- "IEEE 802.11i-2004 Amendment 6: Medium Access Control (MAC) Security Enhancements." LAN/MAN Standards Committee (Project 802). The Institute of Electrical and Electronics Engineers, Inc., 23 Jul 2004.
- Jacobs, David B. "Wireless Security Protocols." Networking Information, News and Tips. TechTarget.com, Mar 2008. 26 Apr 2012.
- Jankiewicz, E., J. Loughney, and T. Narten. "RFC 6434: IPv6 Node Requirements." IETF, Dec 2011. <<http://tools.ietf.org/html/rfc6434>>
- Kent, S., and K. Seo. "RFC 4301: Security Architecture for the Internet Protocol." IETF, Dec 2005. <<http://tools.ietf.org/html/rfc4301>>
- Kent, S. "RFC 4302: IP Authentication Header." IETF, Dec 2005. <<http://tools.ietf.org/html/rfc4302>>
- Kerberos Consortium. "Why Kerberos." Massachusetts Institute of Technology, 2008.

- Kindervag, John. "Killing Data." Forrester Research, 30 Jan 2012.
- Klensin, J. "RFC 3467: Role of the Domain Name System." IETF, Feb 2003.
<<http://tools.ietf.org/html/rfc3467>>
- Klensin, J. "RFC 5321: Simple Mail Transfer Protocol." IETF, Oct 2008.
<<http://tools.ietf.org/html/rfc5321> >
- Kurose, James F., and Keith W. Ross. Computer Networking: A Top-down Approach (Fifth Ed.). Pearson, Boston MA, 2010.
- Manral, V. "RFC 4835: Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)." IETF, Apr 2007. <<http://tools.ietf.org/html/rfc4835>>
- Markoff, John. "Step 1: Post Elusive Proof. Step 2: Watch Fireworks." The New York Times, 16 Aug 2010. 12 Feb 2012.
http://www.nytimes.com/2010/08/17/science/17proof.html?_r=2
- Mason, Andrew. "IPSec Overview." Cisco Press, 31 Oct 2003. 09 May 2012.
<<http://www.ciscopress.com/articles/article.asp?p=25474>>
- Microsoft. "SSL/TLS in Detail." Microsoft Technet. 31 Jul 2003.
<<http://technet.microsoft.com/en-us/library/cc785811%28v=ws.10%29.aspx>>.
- Mockapetris, P. "RFC 882: Domain Names – Concepts and Facilities." IETF, Nov 1983.
<<http://tools.ietf.org/html/rfc882>>
- National Security Agency (NSA). Configuring a Cisco Router for Remote Administration Using the Router Console. Systems and Network Analysis Center, 2007. 3 May 2012. <www.nsa.gov/ia/_files/factsheets/I733-002R-2007.pdf>
- NSA. "NSA Suite B Cryptography." 15 Jan 2009. 1 May 2012.
<http://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml>
- "National Strategy to Secure Cyberspace." Office of the President of the United States. Feb 2003.
- Newman, C. "RFC 2595: Using TLS with IMAP, POP3 and ACAP." IETC, Jun 1999.
<<http://tools.ietf.org/html/rfc2595>>
- National Institute of Standards and Technology (NIST). Federal Information Processing Standards Publications. Advanced Encryption Standard (AES). 26 Nov 2001.

- Open University. "Link Layer Encryption." [Http://www.openlearn.open.ac.uk](http://www.openlearn.open.ac.uk).
1 May 2012.
- Ozment, Andy, and Stuart E. Schechter. "Bootstrapping the Adoption of Internet Security Protocols." I3P under Air Force Contract FA8721-05-0002, Jun 2006.
<<http://weis2006.econinfosec.org/docs/46.pdf>>
- Perrin, Chad. "The CIA Triad." TechRepublic, 30 Jun 2008.
<<http://www.techrepublic.com/blog/security/the-cia-triad/488>>
- Postel, J. "RFC 768: User Datagram Protocol." IETF, 28 Aug 1980.
<<http://www.ietf.org/rfc/rfc768.txt>>
- Postel, Jon. "RFC 791: Internet Protocol Specification." Internet Engineering Task Force, Sept 1981. <<http://www.ietf.org/rfc/rfc791.txt>>
- Ramsdell, B. "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification." IETF, Jul 2004. <<http://tools.ietf.org/html/rfc3851>>
- Rekhter, Y., T. Li, and S. Hares. "A Border Gateway Protocol 4 (BGP-4)." Internet Engineering Task Force. Jan 2006.
- Rescorla, E. "RFC 2818: HTTP over TLS." IETF, May 2000.
<<http://tools.ietf.org/html/rfc2818>>
- Rescorla, E., and A. Schiffman. "RFC 2660: The Secure HyperText Transfer Protocol." IETF, Aug 1999. <<http://tools.ietf.org/html/rfc2660>>
- Rose, M. "RFC 1225: Post Office Protocol – Version 3." IETF, May 1991.
<<http://tools.ietf.org/html/rfc1225> >
- Schneier, Bruce, and Mudge. "Cryptanalysis of Microsoft's PPTP Authentication Extensions." Proceedings of 5th ACM Conference on Communications and Computer Security. Nov 1998. <<http://www.schneier.com/paper-pptpv2.html>>
- SSH Communications Security. Tectia Server 6.2 User Manual. Tectia Corp, 19 Sept 2011.
- Stern, Hal. "Why You Care If P=NP." Snowman On Fire. 10 Oct 2010. 01 Apr 2012.
<<http://www.snowmanonfire.com/2010/10/why-you-care-if-pnp/>>
- Taleb, Nassim. *The Black Swan: The Impact of the Highly Improbable*. New York: Random House, 2007.
- Thales Security Corp. "Layer 2 Encryption vs. Layer 3 Encryption." www.thalessec.com. 30 Apr 2012.

- Trappe, Wade, and Lawrence C. Washington. Introduction to Cryptography with Coding Theory. New Delhi: Pearson Education, 2006.
- Trend Micro. "Vulnerability Exploits Break Records." Trend Micro Incorporated, 2005.
- Ulrich, Steve. "Secure Routing Security Primer." Lecture. Wwww.scribd.com. 2 May 2012. <<http://www.scribd.com/doc/66732420/87/Secure-Routing-Route-Authentication>>
- Verizon. "Ethernet: Features - Verizon Medium Business." Verizon Enterprise Solutions Worldwide Site. 27 Apr 2012. <<http://www.verizonbusiness.com/Medium/products/networking/ethernet/features.xml>>
- Wi-Fi Alliance. "The State of Wi-Fi Security." Wi-Fi Alliance, Jan 2012. <https://www.wi-fi.org/sites/default/files/uploads/files/wp_State_of_Wi-Fi_Security_20120125.pdf>
- Winters, Timothy, William Davie, and Deanna Weidenhamer. Quarterly Retail E-Commerce Sales, 4th Quarter 2011. Washington DC: US Department of Commerce, 2012. US Census Bureau News.
- Woeginger, G.J. "The P-versus-NP Page." www.win.teu.nl, 12 Mar 2012.
- Ylonen, T., and C. Lonvick. "RFC 4251: The Secure Shell (SSH) Protocol Architecture." IETF, Jan. 2006. <<http://tools.ietf.org/html/rfc4251>>
- Ylonen, T., and C. Lonvick. "RFC: 4253: The Secure Shell (SSH) Transport Layer Protocol." IETF, Jan. 2006. <tools.ietf.org/html/rfc4253>
- Yonan, James, and Mattock. "OpenVPN." SourceForge. 11 May 2012. <<http://sourceforge.net/projects/openvpn/>>

REPORT DOCUMENTATION PAGE				<i>Form Approved OMB No. 074-0188</i>	
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) 14-06-2012		2. REPORT TYPE Graduate Research Project		3. DATES COVERED (From – To) 16 May 2011 – 14 June 2012	
4. TITLE AND SUBTITLE An Analysis Of The Computer Security Ramifications Of Weakened Asymmetric Cryptographic Algorithms				5a. CONTRACT NUMBER N/A	
				5b. GRANT NUMBER N/A	
				5c. PROGRAM ELEMENT NUMBER N/A	
6. AUTHOR(S) Bixby, Eric R., Major, USAF				5d. PROJECT NUMBER N/A	
				5e. TASK NUMBER N/A	
				5f. WORK UNIT NUMBER N/A	
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way, Building 640 WPAFB OH 45433-7765				8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/ICW/ENG/12-02	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Intentionally left blank				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.					
13. SUPPLEMENTARY NOTES This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.					
14. ABSTRACT This paper explores the ramifications of what a proof that non-deterministic polynomial (NP) time algorithms could be solved in polynomial (P) time would mean for computer networking and the Internet as a whole. The P = NP problem and is a famous and unresolved mathematical question. If the P and NP classes of problems are really one in the same, there would be significant ramifications across numerous fields, including and especially asymmetric cryptography. Therefore, a great deal of effort in the computer science and mathematics fields has been devoted to this problem over the past 40 years. A significant subset of modern cryptographic systems rely on mathematical principles that make the assumption that P ≠ NP. If P = NP, these cryptographic systems would be in imminent danger of being weakened or completely obviated. As a result, there are many who speculate that the consequences of a P = NP proof would be the ultimate demise of the Internet. However, rarely are such claims substantiated with an analysis demonstrating how such effects would be caused. Therefore, this research attempts to determine the veracity of those claims through analysis of critical Internet protocols. The paper includes an explanation of the P = NP debate by describing what a P problem is and the contrasting it with an NP problem and then showing how they are related. It will then show how certain commonly-used cryptographic systems rely upon problems that fall within NP and describe how a P = NP proof would affect the security of those systems. Next it will examine critical components of computer networking and the Internet and determine how they rely upon potentially weakened cryptologic systems. That examination will include an analysis of how those dependencies impact network security (including data confidentiality, integrity and availability).					
15. SUBJECT TERMS Encryption, Non-deterministic Polynomial, Network Security					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Robert F. Mills, PhD (ENG)
U	U	U	UU	92	19b. TELEPHONE NUMBER (Include area code) (937) 257-3636 x4527; robert.mills@afit.edu

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18