Ph 20.7 – Encryption

Due: Week 10 -v20170314-

This Assignment

As the term winds down and the Ph 20 TAs begin to badger you about submitting your corrections to previous assignments before it ends, this final assignment of Ph 20 is meant to be fairly relaxing in comparison to last week's. The assignment is a choose your own adventure through a portion of the crypto ecosystem. Your first resource for most of this assignment should be Wikipedia, a good starting point to learn the basics of most or all of the concepts we dicuss here. In particular it should sketch the basics of (and perhaps even pseudocode for) the key-exchange mechanism you implement in the third part of the assignment.

If you're feeling a lack of creative writing opportunities in your Caltech education thus far, we encourage you to take full advantage of the opportunity given in the first part of the assignment!

Introduction

This assignment deals with basic crypto technology. Cryptography is a huge and ancient field. It is not possible to cover it in detail, so we're going to pick up some of the basics. By now you are familiar with Linux and the command line environment, which is already half the battle.

The basic premise of encryption is to make it difficult to read data, at least for non-authorized people. In practise, encrypting your whole hard drive is not much use if you are unable to access it, or communicate with other people. Secure exchange of secret information, such as medical records, money, or government documents is still an area of active research. There are many useful online resources such as eff.org, schneier.com, and wikipedia, as well as program specific documentation available online.

Modern crypto is based on extremely powerful mathematics. Many people develop crypto as a defense against oppressive regimes, as no amount of physical force or torture can solve a math problem. That said, raw cryptography doesn't guarantee security or safety. With that in mind, **DON'T use the code you write in this assignment to protect anything you care about**! (In cryptography jargon, "don't roll your own crypto.") Your code is meant to give you a glimpse of the math behind the cryptography which powers much of the internet and commerce these days, but in many ways the math is the simplest part, as the many codes which have been proven mathematically secure only to be abandoned because of glaring practical vulnerabilities indicate.

The Assignment

Each section should correspond to a paragraph or two - there is no need to write a textbook!

1. Develop a threat model. What are you trying to protect, and who? Data, identities, location? Communications? Examples of possible adversaries include
   (a) Hackers looking for financial information or blackmail material
   (b) Spy agencies (public or private)
   (c) Rival researchers
   (d) Snooping spouses
   (e) Botnets, viruses, etc

Describe a credible or interesting adversary model, including their strengths and weaknesses. Points for creativity and plausibility. Do they have physical access to your computer? Do they have legal authority? Do they have the ability to monitor network traffic? Do you have legal avenues of defence?

Reference this threat model throughout the rest of this assignment. In particular, how are the remaining parts relevant?

2. Nearly all encrypted communications protocols begin with communicating entities exchanging keys through public networks. Of course any eavesdroppers could also get access to those keys, so ensuring privacy is a non-trivial issue. Read the wikipedia page about RSA or the Diffie-Hellman protocol, or another key exchange mechanism of your choice. Summarize its method. Briefly describe two known weaknesses or potential vulnerabilities of the method, and how they are currently addressed.

3. Implement a demonstration toy version of your chosen key exchange mechanism in mathematica, python, or another language of your choice. Try to avoid using 'black box' functions or libraries which do most of the work for you. Using a sufficiently short key (8 bits is heaps!), attempt to demonstrate a collision attack. There is more than one kind of collision attack, do whichever one you like, and describe how it works.

4. Examples of commonly used crypto programs include: truecrypt, ecrypt, lyx, tor, otr/jabber, tails, gpg, ssh, strongbox, bitlocker, bleachbit, silent circle. Select one of these or another of your choice. Install it and take it for a test drive. Briefly summarise how it works, what situations it can and cannot address and some known vulnerabilities or drawbacks.

5. Crypto currencies such as Bitcoin rely on 'proof of work'. Devise a simple proof of work to prove that you actually used the program you described in the previous section. Generalized away from the context of Bitcoin, proof of work means something that your TA can easily verify would be easier for you to produce using your chosen program than to fake without it. Note that honest signalling is also a part of evolutionary biology!

6. Summarize the essential elements for a secure authentication mechanism (which might or might not include a text password). Describe one easy way to compromise a strong password.

7. Write a sentence or two on what you've learned about encryption. How do the systems you've implemented or described help address your threat model?

8. Encrypt your assignment's pdf using the TA's public key and, after taking steps to prevent a MITM (Man In The Middle) attack, submit the encrypted assignment. You'll need to generate a public/private keypair using the `gpg` unix utility and use your private key and the TA's public key to do the encryption. Consult `man gpg` or a `gpg` tutorial of your choice, for example `https://www.madboa.com/geek/gpg-quickstart/`, for more details. You can find your TA's public key on the Ubuntu keyserver, `http://keyserver.ubuntu.com/`; you're encouraged to upload your own public key there as well.

9. Don't forget to submit those corrections to previous assignments!

10. Bask in the feeling of having completed Ph 20. Don't forget to fill out your course evaluations! As you may have noticed from the version numbers on each assignment, Ph 20 is constantly

in flux and in order to improve it we need to know what worked and what didn't. Thanks in advance!

Finally, a reminder that there are two further courses in the Ph 2x sequence, which can be taken independently and in either order: Ph 21, which emphasizes data analysis, and Ph 22, which focuses on numerical methods.