

Lecture 1: Class Introduction

*Instructor: Brent Waters**Scribe: Rashid Kaleem*

1 Class Overview

The course is intended to focus on some recent topics, covering those topics in detail to get an idea of the current state of the art. It is important to collaborate as well as work through the material in the class.

1.1 Goals

1. Able to read a CRYPTO/EUROCRYPT paper, digest it and understand the main idea.
2. Able to summarize a paper's main idea or the gain the intuition the paper's core novelty.
3. Able to give a 20 minute talk presenting a cryptography paper (selected from a set of papers shortlisted by the instructor).

1.2 Grading

Grading would be distributed as follows.

1. Class participation (15%). Everyone is encouraged to participate in the class. Since this is a topics course, everyone is expected to contribute to the discussion.
2. Scribe notes (15%). Everyone has to volunteer to take notes for two lectures. The notes should reflect what had been discussed in the class and have to be written up in LaTeX.
3. Problem Sets (35%). There will be 3 plus/minus one problem sets. You are free to discuss the problems with anyone, but you must acknowledge this and write up your own solution separately.
4. Presentations (35%). After about four weeks, a topics list will be uploaded. Everyone will select one topic from the list and prepare a 20 minutes conference-talk styled presentation.

With the course logistics out of the way, let us talk about cryptography.

2 Cryptography

Let us consider various situations where we can use some crypto techniques to solve.

1. In graphics research, artefacts (videos, high definition images etc) generated can be very very large. When submitting these artefacts to a conference, the researcher would like to wait till the last minute to put in as much effort as possible (possibly hoping to solve some remaining issues). But if everyone waits till the last moments to submit their artefact, the submission server can crash due to the overwhelming number of requests to upload artefacts. How do we solve this problem?
 - Compress the artefact. But it might be the case that the artefact is already compressed (for instance a compression codec) and still be very very large. In this case, compression would not be of any use.
 - Use hashing. We can hash the artefact and upload the small hash value. Later the server can request the user to upload the artefact at a convenient time and check the hash value of the uploaded artefact to ensure that the artefact is the same as that at the time of submission. But it should be hard to find a collision for the hash function since that would enable a malicious user to modify the artefacts and still get the same hash value, hence cheat the system.
2. Jenna wants to authorize Brent for a F26 parking permit via email. The parking permit is to be obtained from the parking office. A malicious user (or a grad student) might forge such an email and take it to the parking office and obtain Brent's parking permit. How can Jenna ensure that her email isn't forged?
 - Jenna can sign all her emails using her private key and anyone can verify whether the email is actually sent by Jenna by decrypting the message using her public key (which would be published publicly). If anyone tried to forge her emails, it would not be possible since only Jenna has the private key to encrypt those messages. Note however that this does not address the identity of Brent when he goes to the parking office as anyone claiming to be Brent can be assumed so unless a certificate from a trusted third party is presented (ID card, University card etc).
3. Databases are used to store large amounts of data, sometimes this data contains personal information and in the event of a security breach or theft, all the data is easily accessible to the intruder. How do we protect data in such events?
 - Restrict who has access to database. But we cannot rely on individuals since any one of them could collude.
 - Hash the data and verify the hash values. This would ensure the integrity of the data, but it might still be possible for an intruder to read the personal data and use it, though modification of the data would be noticed.
 - Encrypt the data. When an intruder breaks into the system, he/she cannot read the contents of the database since it is encrypted. Even if the physical disk drives are stolen, it would not be possible to read the data.

4. Suppose (eventually) you make a scientific breakthrough and would like to publish it as soon as possible. But meanwhile, you would also like to brag about it. Suppose one of your colleagues is skeptical, and asks you to present the breakthrough. At the same time you are skeptical that if you do present it, someone else might claim it before you do. How do you address this issue?
 - Use a zero knowledge proof, which is a proof that just shows if a statement is true and does not provide any knowledge to the audience other than the truth of the statement.
5. A software company finds a bug in its software. Suppose that if a particular header contains a particular sequence of values, it is some malicious code and should be dropped. A patch that drops such packets may be desirable but releasing it can expose the vulnerability, and someone might reverse engineer the patch to exploit users who have not patched their software.
 - Drop more packets to make it difficult to exploit the actual vulnerability. But this could lead to the protocol not functioning properly. As a limiting case, the protocol can be modified to drop all packets.
 - Release the encrypted version of the patch, and at a later time, release the decryption key thus enabling the patch. This would lead to the problem of ensuring that everyone has both the key and the patch, since anyone not having one component would be vulnerable.
 - Distribute an obfuscated patch. This should make it hard to reverse engineer the patch and determine where the vulnerability lies and at the same time provide protection.

3 Analysis of Topics

Generally the class would aim at the following five step analysis of any topic.

1. Informal definitions: Identify a real-life situation where security is an issue. Try to understand what security would mean in the scenario.
2. System properties: Define the parties involved, their roles, how they interact with each other and what is to be expected from a secure system.
3. Formalize security: Formalize the definitions of security, for instance what it means to be hard, secure or what sort of forgery is possible.
4. Describe System: Once the processes required to implement the system have been defined, it is time to implement the algorithms that fulfill those definitions and create the system.
5. Proof of Security: Prove that the system is secure. This is usually done in the following manner. We usually derive an encryption scheme based on an assumption T (i.e. factoring is hard). Now we assume that an attacker A is capable of breaking the

security of our system. We will show that we can create a procedure B which is allowed to make calls to A as desired such that the procedure B contradicts our assumption T (i.e. factors in P). Hence such an attacker A cannot exist.

Now we consider Public key cryptography as an example.

3.1 Public Key Cryptography

1. Alice generate a public/private key pair and publishes her public key globally. To communicate with Alice, anyone can use her public key to encrypt the message he/she intends to send her. Alice, on receipt of the message can use her private key to decrypt the message. We would not like the message to be easily decrypted by any key besides the private key that Alice owns.