Proofs in Cryptography*

Ananth Raghunathan†

Abstract

We give a brief overview of proofs in cryptography at a beginners level. We briefly cover a general way to look at proofs in cryptography and briefly compare the requirements to traditional reductions in computer science. We then look at two security paradigms, indistinguishability and simulation based security. We also describe the security models for Secret Key and Public Key systems with appropriate motivations. Finally, we cover some advanced topics and conclude with a few exercises and hints.

1 Introduction

TALK ABOUT THE NEED FOR WORKING WITH REDUCTIONS AND DIFFERENT SECURITY DEFINITIONS. THE SUBTLETIES THAT ARISE IN DEFINITIONS, THE TIME IT TOOK TO COME UP WITH GOOD DEFINITIONS OF SECURITY.

Before we go about describing a general outline to prove the security of constructions, let us have an informal discussion. Proving the impossibility of a particular computational task is extremely difficult. It is very close to the P vs. NP question (ADD REFERENCE) that is the central question in computer science today. It becomes doubly difficult when do not know what class of computational tasks we are allowed to utilize (unlike the case of P vs. NP, where a Turing machine has been rigorously and beautifully formalized). Therein lies the central difficulty in proving the security of cryptographic constructions.

One way to address the latter question is to explicitly model what an adversary who intends to break our cryptosystem is allowed to do. Notice this requires us to define at least two things: what do we mean by an "adversary" and what it means to "break" the cryptosystem. A lot of emperical and theoretical work goes into the current models of "adversary" and "break" and currently the definitions we use have been used widely in practice and give us good results emperically.

However, as it happens often in this field, the adversary in the real world is not constrained to behave as we dictate him[1]. There have been several real world attacks (side-channel attacks, cold-boot attacks are recent innovative attack (ADD REFERENCES)) that completely side-step the adversarial model and as expected completely breaks the security of the scheme. However, this only serves as motivation to model stronger adversaries and thereby construct more robust cryptosystems that are provably robust (with respect to the new models).

In conclusion, remember two important things: To analyse a cryptosystem you must define an adversary model and a security game.

*__Rough Draft__ of a handout for CS255: Introduction to Cryptography by Dan Boneh.

†`ananthr@stanford.edu`

[1]or her. I will restrict myself to male pronouns for simplicity.

1. **Adversary model:** This defines formally the power of the adversary. It includes specifics whether the adversary is deterministic/randomized, uniform/non-uniform, interactive/non-interactive and how he interacts with the security game.

2. **Security game:** This defines formally the power the adversary has over the cryptosystem. Whether he has access to a single ciphertext, multiple ciphertexts, multiple keys, etc. It also defines when an adversary is said to break the system.

Often adversaries are modeled similarly, but depending on whether you want weak or strong security guarantees, you modify the security game accordingly.

Two broad paradigms for security games are indistinguishablity games and simulation games. Subsection 1.3 will talk about it in more detail.

1.1 General Outline

In this section, I will give a general outline for simple proofs of security in cryptosystems. This broadly follows the outline in Katz-Lindell.

1.2 Relation to reductions in Complexity Theory

Readers not intimately familiar with the complexity classes P and NP and notions of NP-hardness, NP-completeness and witnesses and membership in languages should skip this section. It is not required to understand proofs in crytography but provides an interesting comparison.

At this juncture, it is interesting to compare proofs in cryptography to proofs in complexity theory. In particular, for an undergraduate audience, the most familiar reductions proofs involve proving the NP-completeness of problems. Let us recall how a proof that a language $\mathscr{L}$ is NP-complete. Proving that $\mathscr{L} \in NP$ is **not a reduction**. It requires us to present an algorithm that verifies that w a string is a witness to the fact that $x \in \mathscr{L}$. This must be done efficiently (i.e., in time polynomial in $|x|$, the input length) with a witness string w that is not too much longer than $|x|$. We require a reduction proof to show that $\mathscr{L}$ is NP-hard.

To prove $\mathscr{L}$ is NP-hard we are required to show that deciding membership in $\mathscr{L}$ is **as hard as** deciding membership of **any** NP language. Because there exist NP-complete languages, like 3SAT it suffices to show that 3SAT is **no harder** than deciding membership in $\mathscr{L}$. In other words, given an efficient algorithm that decides membership in $\mathscr{L}$, we need to construct an efficient algorithm that decides whether a boolean formula φ is satisfiable or not.

Now we can take a step back and look at the parallels. Clearly, this is a proof of the contrapositive of the statement that we are really after: deciding membership in $\mathscr{L}$ is as hard as 3SAT. Secondly, we assume the existence of an efficient algorithm that already decides $\mathscr{L}$ and use it to construct an efficient algorithm that solves 3SAT. Thus, superficially, it looks similar to proofs in cryptography. But there are a few subtle differences that I hope will further strengthen your understanding of proofs in cryptography:

1. Notice that we assume an algorithm for $\mathscr{L}$ that solves **all** instances of $x \in \mathscr{L}$. Adversaries in cryptography are generally probabilistic and are required to work only for a non-negligible fraction of the inputs. In particular, an adversary that steals "some" credit card numbers is sufficient; we are not particularly impressed with a proof that there is no adversary who steals "all" credit card numbers possibly used in the system because that adversary is unreasonably strong.

This implies that in complexity theory proofs, it suffices to take a 3SAT instance φ and convert it to **some** instance x for $\mathscr{L}$. But in cryptography, we require that this instance x look identical to an adversary in the security game. In other words, it must appear as though it came from a real distribution over possible instances (for eg., refer to the solution of HW2 problem 4 where most people tripped up over this fact).

This leads to a more general and very deep and interesting topic in the intersection of complexity and cryptography—the notion of **average-case complexity** vs. **worst-case complexity**. Cryptography problems are required to be proven hard **on the average**[2]. Proving that some instances are hard (i.e., there are no adversaries who break **all** the crypto instances) does not inspire confidence[3]. Therefore remember, you are required to show that **average** instances of certain problems are hard. Therefore, you must not assume an adversary works on arbitrary values, rather that he works only if the values given to him look identical (or indistinguishable) from values that might arise in a security game (that he purportedly breaks).

2. The second, and slightly less important distinction is that proofs involving NP-completeness do not require you to conver the solution of the membership of $x \in \mathscr{L}$ back to whether $\varphi \in$ 3SAT or not. Because it is a binary result ("yes" or "no") the same is true for the satisfiability of φ. But in cryptography, sometimes the proof is incomplete unless you use the values given by the adversary to break the original problem you assumed were hard. And note that both **constructing inputs** to the adversaries (i.e., simulating him) and **recovering hard-to-compute values** to break your assumption must be efficient algorithms.

3. Finally, proofs of NP-completeness are almost always deterministic. Therefore algorithms work with probability 1. However, in cryptography, you have the notion of the advantage of an adversary, the probability that he succeeds over a naïve algorithm. You are required to take an adversary that works with non-negligible advantage and use it to construct an adversary to break the assumption with non-negligble advantage.

   In cryptography, as in complexity, a lot of research goes into constructing reductions that are "tight", i.e., they are efficient and succeed with the **same, or close to same advantage** as the purported adversaries. These are important in practice becaues they help keep security parameters small. Otherwise, slack in reductions will require us to ramp up the parameters to make the adversary's advantage non-negligible. To take a toy example, if a system has an adversary with advantage ε, we can construct an algorithm that inverts RSA with probability ε^2. Clearly this is a weak reduction. Since we require $\varepsilon < 2^{-80}$, we must choose parameters such that RSA adversaries have advantage 2^{-160}. This requires us to make the parameters 8 times as big as what we would have with a tight reduction[4]!

1.3 Two security paradigms*

TALK ABOUT THE TWO SECURITY PARADIGMS, MANOJ PRABHAKARAN'S SLIDES WILL BE USEFUL.

FOR NEXT YEAR. NOT REQUIRED FOR CS255.

[2]This is highly informal and only described thus to provide intuition. More formally, you define easily samplable distributions corresponding to the real instantiation and average here refers to instances drawn from this distribution. It isn't required to be uniform, although that is most intuitive.

[3]Imagine if your bank only guaranteed that there are definitely some customers whose credit card information is secure!

[4]Remember, the best algorithm today is roughly exponential in the **cube-root** of the security parameter

1.3.1 IND*

1.3.2 SIM*

2 Secret Key Cryptography

In this section, I will discuss some of the security models for various symmetric (or secret) key constructions. I will briefly provide motivation for the definitions and a paragraph or two on how to provide proofs. Formal definitions can be found in standard text-books.

2.1 Information Theoretic Arguments

It wasn't until the early 50's, after Shannon's paper that people began to formulate and think of what it meant to be secure. Shannon gave an elegant definition that did not involve any notion of computation (partly because the Turing machine itself was only a decade old). Instead he looked at probability distirbutions and information theory.

The first definition of security, **perfect secrecy** states that a ciphertext leaks no information about the plaintext (to **any**, even an all-powerful adversary). This is equivalent to stating that the probability that a given message maps to a given ciphertext is exactly identical for every pair of messages and ciphertexts (for randomly chosen keys).

Therefore, given a ciphertext, one learns nothing about which message it encrypts. This is a neat definition which does not require us to talk about adversaries and provide reductions, only work with probabilities. But as mentioned in class, perfectly secure systems require the keys to be at least as long as the message. Therefore they are thoroughly impractical. This requires us to relax the definition of security.

To prove that a scheme is perfectly secure, you must be able to show that for any pair of messages, the probability that they map to a given ciphertext is identical. This would usually be a straightforward probability argument. One example of a perfectly secure system is a one-time pad.

2.2 Semantic Security

Now that we know perfect security in some sense is overkill, we look to relaxations. In their landmark paper in (CITE PAPER, YEAR) Goldwasser and Micali came up with a brilliant concept called semantic security. They argued that, in the spirit of Shannon's argument, we require that a ciphertext must reveal nothing about the message. If the probability distributions of messages, given ciphertexts are identical this is trivially true. But it would also be true if no efficient adversary can discern a difference because in the real world we are only worried about adversaries that are efficient.

The formal definition of semantic security precisely states this. Given **any** two pair of messages m_0 and m_1, if the ciphertexts of m_0 and m_1 cannot be distinguished by efficient adversaries, the scheme is said to be semantically secure.

2.3 Semantic Security against CPA

2.4 PRGs

2.5 PRFs

2.6 Block Ciphers or Pseudorandom Functions

2.7 Message Authentication Codes

2.8 Collision-resistant Hash functions

3 Public Key Cryptography

3.1 CCA

3.2 OWFs

3.3 Signature schemes

4 Advanced Topics*

4.1 Hybrid Arguments*

4.2 Randomized Self Reductions*

5 Exercises

1. *Prove that flipping bits of a PRG is still secure.* Given a PRG G that is secure (i.e., satisfies the indistinguishability game) construct a PRG $G'(x) := \overline{b_1}\overline{b_2}\cdots\overline{b_n}$ where $(b_1 b_2 \cdots b_n) = G(x)$, the output of G on x and $\overline{b}$ is the bit b flipped. Show that G' is still secure. To do this, construct an adversary $\mathscr{B}$ that given a string decides whether it is from G (i.e., G evaluated on a random input) or completely random with probability better than $1/2$. To do this, it uses $\mathscr{A}$ who claims to break the distinguishability test for G'. Therefore, $\mathscr{B}$ must somehow use its challenge string w and concoct a string for $\mathscr{A}$ to help it decide whether it is random or not.

2. *Prove that permuting the bits of a PRG is still secure.* Given a PRG G, and a permutation π (and its inverse permutation π^{-1})[5], consider $G_\pi(x) = (b_{\pi(1)} b_{\pi(2)} \cdots b_{\pi(n)})$ where $(b_1 b_2 \cdots b_n) = G(x)$. Show, largely along the lines of Exercise 1 that G_π is secure. In particular if π is the identity this is trivial. If $\pi(i) = n+1-i$, i.e., G' simply reverses the bits of the output, the proof is covered in the section. You need to use π^{-1}.

3. *Prove that a ℓ-expanding PRG is a small domain PRF.* This is HW1 problem 7b. The formal statement can be found on the website. Here, to prove the security, given an adversary $\mathscr{A}$ for the small domain PRF, you are required to construct an adversary $\mathscr{B}$ that interacts with the PRG-Challenger and distinguishes whether the string given were random or pseudorandom. Remember, $\mathscr{B}$ must present a PRF-Challenger Chal to $\mathscr{A}$ and ensure that he is **faithfully** simulating either a random function or a pseudorandom function.

[5] I.e., $\pi^{-1}(\pi(i)) = i$ for all i.

4. *Prove that truncating the output of a PRF is still secure.* More formally, given a PRF $F : K \times X \to \{0,1\}^m$, construct $F'(k,x) = F(k,x)|_\ell$, where $\ell < m$ is the output length of $F' : K \times X \to \{0,1\}^\ell$ is also secure. The way the proof would go would be to take an efficient adversary $\mathscr{A}$ that breaks the PRF indistinguishability game for F' and construct $\mathscr{B}$ that has a challenger Chal as in the security definition for F and must guess whether Chal is giving him his queries evaluated on F or a random function. Note here $\mathscr{B}$ **must present** a challenger for $\mathscr{A}$. See Exercise 3 above for more ideas.

5. *Prove that any subset of the bits of a PRG are still indistinguishable.* This proof is largely along the lines of Exercise 4 above. This is much simpler because you aren't required to simulate queries (the challenger for a PRG simply presents one of two strings, and asks for a distinguisher).

6. *Introduce hardness of SVP in perp lattices and prove that $f_A(x) = A \cdot x \,(\text{mod}\, q)$ is a collision resistant hash function.* There is this hard problem that has recently received a lot of interest in the crypto community that we will refer to as SVP in random lattices. The formal statement is as follows: Consider a random matrix $A \in \mathbb{Z}_q^{n\times m}$. Finding a "short" $e \in \mathbb{Z}^m$ such that $A \cdot e = 0 \,(\text{mod}\, q)$ is hard. Here short implies that $\|e\|_2 \leq \beta$ for some parameter β. Therefore, an adversary $\mathscr{B}$ that breaks this system will, given a random matrix output e that satisfies the above conditions.

   Now, we can construct a simple collision-resistant hash function from this system. Consider $H : \{0,1\}^m \to [q]^n$[6], that takes m-bit strings and outputs n elements in $[q]$. For a random A, define $H_A(x) = A \cdot x \,(\text{mod}\, q)$. Check for yourself that the input and outputs of the hash function are correct. Show that finding collisions in $H_A(\cdot)$ are as hard as SVP in a random lattice.

   A proof would require you to construct an adversary $\mathscr{B}$ that takes an adversary $\mathscr{A}$ that outputs collisions from $H_A(\cdot)$ and use this to break SVP.

7. *Prove that PRGs imply One Way Functions.* Recollect in class, we mentioned several ways to construct one-way functions. The simplest way, and the one with least structure to exploit was this generic construction from pseudorandom functions. First, we will show that a PRG implies a one-way function. Let $G : \{0,1\}^s \to \{0,1\}^n$, $n > s$ be a PRG that takes a s-bit input and outputs n pseudorandom bits. Show that $F(x) := G(x)$ is a one-way function. An adversary for a OWF takes the image of a random element and outputs a preimage that evaluates to the same value. Therefore, given $G(r)$ for random r, the adversary outputs r' (that might be equal to r) such that $G(r) = G(r')$. Therefore show that you can construct an adversary $\mathscr{B}$ that distinguishes a PRG output from random given $\mathscr{A}$ that inverts the one-way function with non-negligble probability.

8. *Prove that a PRFs imply OWFs.* Now we show that we can construct PRFs from OWFs. Recall that given sufficiently expanding PRGs we can construct PRFs therefore for sufficiently expanding PRGs, this proves the previous exercise as well. Consider $\text{OWF}(x) := F(x,0)\|F(x,1)$ for simplicity. Show that an adversary $\mathscr{A}$ that inverts the OWF can be used to construct an adversary $\mathscr{B}$ that interacts with a PRF challenger Chal and decides whether it is dealing with a random function or a pseudorandom function. This means that $\mathscr{B}$ must behave sufficiently differently for the two possible games the challenger plays with him (see the definition of PRFs).

   Now, can you see how this proof extends to $\text{OWF}(x) := F(x,\alpha_1)\|F(x,\alpha_2)\ldots\|F(x,\alpha_q)$ for any **fixed constants** $\alpha_1,\ldots,\alpha_q$ where q is at most a polynomial in $|x|$?

[6]Recollect that $[q] = \{1,\ldots,q-1,q\}$.