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

[*]**Rough Draft** of a handout for CS255: Introduction to Cryptography by Dan Boneh.

[†]ananthr@stanford.edu

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.

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

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

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.,

[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

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]!

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 Two security paradigms*

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

FOR NEXT YEAR. NOT REQUIRED FOR CS255.

1.3.1 IND*

1.3.2 SIM*

2 Secret Key Cryptography

2.1 Information Theoretic Arguments

2.2 Semantic Security

2.3 PRGs

2.4 PRFs

2.5 Block Ciphers or Pseudorandom Functions

2.6 Message Authentication Codes

2.7 Collision-resistant Hash functions

3 Public Key Cryptography

3.1 Semantic Security against CPA

3.2 CCA

3.3 OWFs

3.4 Signature schemes

4 Advanced Topics*

4.1 Hybrid Arguments*

4.2 Randomized Self Reductions*

5 Exercises

1. Prove that truncating the output of a PRF is still secure

2. Prove that any subset of the bits of a PRG are still indistinguishable
3. Prove that a ℓ-expanding PRG is a small domain PRF
4. Introduce hardness of SVP in perp lattices and prove that $f_A(x) = A \cdot x \pmod{q}$ is a collision resistant hash function
5. Prove that a PRF implies a One Way Function