1.4 The Basic Principles of Modern Cryptography

The previous section has given a taste of historical cryptography. It is fair to say that, historically, cryptography was more of an art than any sort of science: schemes were designed in an ad-hoc manner and then evaluated based on their perceived complexity or cleverness. Unfortunately, as we have seen, all such schemes (no matter how clever) were eventually broken.

Modern cryptography, now resting on firmer and more scientific foundations, gives hope of breaking out of the endless cycle of constructing schemes and watching them get broken. In this section we outline the main principles and paradigms that distinguish modern cryptography from classical cryptography. We identify three main principles:

1. *Principle 1* — the first step in solving any cryptographic problem is the formulation of a rigorous and precise definition of security.
2. *Principle 2* — when the security of a cryptographic construction relies on an unproven assumption, this assumption must be precisely stated. Furthermore, the assumption should be as minimal as possible.
3. *Principle 3* — cryptographic constructions should be accompanied by a rigorous proof of security with respect to a definition formulated according to principle 1, and relative to an assumption stated as in principle 2 (if an assumption is needed at all).

We now discuss each of these principles in greater depth.

1.4.1 Principle 1 – Formulation of Exact Definitions

One of the key intellectual contributions of modern cryptography has been the realization that formal definitions of security are *essential* prerequisites

for the design, usage, or study of any cryptographic primitive or protocol. Let us explain each of these in turn:

1. *Importance for design:* Say we are interested in constructing a secure encryption scheme. If we do not have a firm understanding of what it is we want to achieve, how can we possibly know whether (or when) we have achieved it? Having an exact definition in mind enables us to better direct our design efforts, as well as to evaluate the quality of what we build, thereby improving the end construction. In particular, it is much better to define what is needed *first* and then begin the design phase, rather than to come up with a *post facto* definition of what has been achieved once the design is complete. The latter approach risks having the design phase end when the designers' patience is tried (rather than when the goal has been met), or may result in a construction that achieves *more* than is needed and is thus less efficient than a better solution.

2. *Importance for usage:* Say we want to use an encryption scheme within some larger system. How do we know which encryption scheme to use? If presented with a candidate encryption scheme, how can we tell whether it suffices for our application? Having a precise definition of the security achieved by a given scheme (coupled with a security proof relative to a formally-stated assumption as discussed in principles 2 and 3) allows us to answer these questions. Specifically, we can define the security that *we* desire in our system (see point 1, above), and then verify whether the definition satisfied by a given encryption scheme suffices for our purposes. Alternatively, we can specify the definition that we need the encryption scheme to satisfy, and look for an encryption scheme satisfying this definition. Note that it may not be wise to choose the "most secure" scheme, since a weaker notion of security may suffice for our application and we may then be able to use a more efficient scheme.

3. *Importance for study:* Given two encryption schemes, how can we compare them? Without any definition of security, the only point of comparison is efficiency, but efficiency alone is a poor criterion since a highly efficient scheme that is completely insecure is of no use. Precise specification of the level of security achieved by a scheme offers another point of comparison. If two schemes are equally efficient but the first one satisfies a stronger definition of security than the second, then the first is preferable.[5] There may also be a trade-off between security and efficiency (see the previous two points), but at least with precise definitions we can understand what this trade-off entails.

[5] Of course, things are rarely this simple.

Of course, precise definitions also enable rigorous proofs (as we will discuss when we come to principle 3), but the above reasons stand irrespective of this.

It is a mistake to think that formal definitions are not needed since "we have an intuitive idea of what security means". For starters, different people have different intuition regarding what is considered secure. Even one person might have multiple intuitive ideas of what security means, depending on the context. For example, in Chapter 3 we will study four different definitions of security for private-key encryption, each of which is useful in a different scenario. In any case, a formal definition is necessary for communicating your "intuitive idea" to someone else.

An example: secure encryption. It is also a mistake to think that formalizing definitions is trivial. For example, how would you formalize the desired notion of security for private-key encryption? (The reader may want to pause to think about this before reading on.) We have asked students many times how secure encryption should be defined, and have received the following answers (often in the following order):

1. *Answer 1 — an encryption scheme is secure if no adversary can find the secret key when given a ciphertext.* Such a definition of encryption completely misses the point. The aim of encryption is to protect the message being encrypted and the secret key is just the means of achieving this. To take this to an absurd level, consider an encryption scheme that ignores the secret key and just outputs the plaintext. Clearly, no adversary can find the secret key. However, it is also clear that no secrecy whatsoever is provided.[6]

2. *Answer 2 — an encryption scheme is secure if no adversary can find the plaintext that corresponds to the ciphertext.* This definition already looks better and can even be found in some texts on cryptography. However, after some more thought, it is also far from satisfactory. For example, an encryption scheme that reveals 90% of the plaintext would still be considered secure under this definition, as long as it is hard to find the remaining 10%. But this is clearly unacceptable in most common applications of encryption. For example, employment contracts are mostly standard text, and only the salary might need to be kept secret; if the salary is in the 90% of the plaintext that is revealed then nothing is gained by encrypting.

   If you find the above counterexample silly, refer again to footnote 6. The point once again is that if the definition as stated isn't what was meant, then a scheme could be proven secure without actually providing the necessary level of protection. (This is a good example of why *exact* definitions are important.)

[6] And lest you respond: "But that's not what I meant!", well, that's exactly the point: it is often not so trivial to formalize what one means.

3. *Answer 3 — an encryption scheme is secure if no adversary can determine any character of the plaintext that corresponds to the ciphertext.* This already looks like an excellent definition. However, other subtleties can arise. Going back to the example of the employment contract, it may be impossible to determine the actual salary or even any digit thereof. However, should the encryption scheme be considered secure if it leaks whether the encrypted salary is greater than or less than $100,000 per year? Clearly not. This leads us to the next suggestion.

4. *Answer 4 — an encryption scheme is secure if no adversary can derive any meaningful information about the plaintext from the ciphertext.* This is already close to the actual definition. However, it is lacking in one respect: it does not define what it means for information to be "meaningful". Different information may be meaningful in different applications. This leads to a very important principle regarding definitions of security for cryptographic primitives: *definitions of security should suffice for all potential applications.* This is essential because one can never know what applications may arise in the future. Furthermore, implementations typically become part of general cryptographic libraries which are then used in may different contexts and for many different applications. Security should ideally be guaranteed for all possible uses.

5. *The final answer — an encryption scheme is secure if no adversary can compute* any *function of the plaintext from the ciphertext.* This provides a very strong guarantee and, when formulated properly, is considered today to be the "right" definition of security for encryption. Even here, there are questions regarding the attack model that should be considered, and how this aspect of security should be defined.

Even though we have now hit upon the correct requirement for secure encryption, conceptually speaking, it remains to state this requirement mathematically and formally, and this is in itself a non-trivial task (one that we will address in detail in Chapters 2 and 3).

As noted in the "final answer", above, our formal definition must also specify the attack model: i.e., whether we assume a ciphertext-only attack or a chosen-plaintext attack. This illustrates a general principle used when formulating cryptographic definitions. Specifically, in order to fully define security of some cryptographic task, there are two distinct issues that must be explicitly addressed. The first is what is considered to be a break, and the second is what is assumed regarding the power of the adversary. The break is exactly what we have discussed above; i.e., an encryption scheme is considered broken if an adversary learns some function of the plaintext from a ciphertext. The *power of the adversary* relates to assumptions regarding the actions the adversary is assumed to be able to take, as well as the adversary's computational power. The former refers to considerations such as whether the adversary is assumed only to be able to eavesdrop on encrypted messages

(i.e., a ciphertext-only attack), or whether we assume that the adversary can also actively request encryptions of any plaintext that it likes (i.e., carry out a chosen-plaintext attack). A second issue that must be considered is the computational power of the adversary. For all of this book, except Chapter 2, we will want to ensure security against any *efficient* adversary, by which we mean any adversary running in polynomial time. (A full discussion of this point appears in Section 3.1.2. For now, it suffices to say that an "efficient" strategy is one that can be carried out in a lifetime. Thus "feasible" is arguably a more accurate term.) When translating this into concrete terms, we might require security against any adversary utilizing decades of computing time on a supercomputer.

In summary, any definition of security will take the following general form:

> *A cryptographic scheme for a given task is secure if no adversary of a specified power can achieve a specified break.*

We stress that the definition never assumes anything about the adversary's *strategy*. This is an important distinction: we are willing to assume something about the adversary's capabilities (e.g., that it is able to mount a chosen-plaintext attack but not a chosen-ciphertext attack), but we are *not* willing to assume anything about *how it uses* its abilities. We call this the "arbitrary adversary principle": security must be guaranteed for *any* adversary within the class of adversaries having the specified power. This principle is important because it is impossible to foresee what strategies might be used in an adversarial attack (and history has proven that attempts to do so are doomed to failure).

Mathematics and the real world. A definition of security essentially provides a mathematical formulation of a real-world problem. If the mathematical definition does not appropriately model the real world, then the definition may be useless. For example, if the adversarial power under consideration is too weak (and, in practice, adversaries have more power), or the break is such that it allows real attacks that were not foreseen (like one of the early answers regarding encryption), then "real security" is not obtained, even if a "mathematically-secure" construction is used. In short, a definition of security must accurately model the real world in order for it to deliver on its mathematical promise of security.

It is quite common, in fact, for a widely-accepted definition to be ill-suited for some new application. As one notable example, there are encryption schemes that were proven secure (relative to some definition like the ones we have discussed above) and then implemented on smart-cards. Due to physical properties of the smart-cards, it was possible for an adversary to monitor the *power usage* of the smart-card (e.g., how this power usage fluctuated over time) as the encryption scheme was being run, and it turned out that this information could be used to determine the key. There was nothing wrong with the security definition or the proof that the scheme satisfied this

definition; the problem was simply that there was a mismatch between the definition and the real-world implementation of the scheme on a smart-card.

This should not be taken to mean that definitions (or proofs, for that matter) are useless! The definition — and the scheme that satisfies it — may still be appropriate for other settings, such as when encryption is performed on an end-host whose power usage cannot be monitored by an adversary. Furthermore, one way to achieve secure encryption on a smart-card would be to further *refine* the definition so that it takes power analysis into account. Or, perhaps hardware countermeasures for power analysis can be developed, with the effect of making the original definition (and hence the original scheme) appropriate for smart-cards. The point is that with a definition you at least know where you stand, even if the definition turns out not to accurately model the particular setting in which a scheme is used. In contrast, with no definition it is not even clear what went wrong.

This possibility of a disconnect between a mathematical model and the reality it is supposed to be modeling is not unique to cryptography but is something that occurs throughout science. To take an example from the field of computer science, consider the meaning of a *mathematical proof* that there exist well-defined problems that computers cannot solve.[7] The immediate question that arises is *what does it mean for "a computer to solve a problem"*? Specifically, a mathematical proof can be provided only when there is some mathematical definition of what a computer is (or to be more exact, what the process of computation is). The problem is that computation is a real-world process, and there are many different ways of computing. In order for us to be really convinced that the "unsolvable problem" is really unsolvable, we must be convinced that our mathematical definition of computation captures the *real-world process* of computation. How do we know when it does?

This inherent difficulty was noted by Alan Turing who studied questions of what can and cannot be solved by a computer. We quote from his original paper [140] (the text in square brackets replaces original text in order to make it more reader friendly):

> *No attempt has yet been made to show [that the problems we have defined to be solvable by a computer] include [exactly those problems] which would naturally be regarded as computable. All arguments which can be given are bound to be, fundamentally, appeals to intuition, and for this reason rather unsatisfactory mathematically. The real question at issue is "What are the possible processes which can be carried out in [computation]?"*
>
> *The arguments which I shall use are of three kinds.*
>
> (a) *A direct appeal to intuition.*

[7] Those who have taken a course in computability theory will be familiar with the fact that such problems do indeed exist (e.g., the Halting Problem).

(b) *A proof of the equivalence of two definitions (in case the new definition has a greater intuitive appeal).*

(c) *Giving examples of large classes of [problems that can be solved using a given definition of computation].*

In some sense, Turing faced the exact same problem as cryptographers. He developed a mathematical model of computation but needed to somehow be convinced that the model was a good one. Likewise, cryptographers define notions of security and need to be convinced that their definitions imply meaningful security guarantees in the real world. As with Turing, they may employ the following tools to become convinced:

1. *Appeals to intuition:* the first tool when contemplating a new definition of security is to see whether it implies security properties that we intuitively expect to hold. This is a minimum requirement, since (as we have seen in our discussion of encryption) our initial intuition usually results in a notion of security that is too weak.

2. *Proofs of equivalence:* it is often the case that a new definition of security is justified by showing that it is equivalent to (or stronger than) a definition that is older, more familiar, or more intuitively-appealing.

3. *Examples:* a useful way of being convinced that a definition of security suffices is to show that the different real-world attacks we are familiar with are ruled out by the definition.

In addition to all of the above, and perhaps most importantly, we rely on the *test of time* and the fact that with time, the scrutiny and investigation of both researchers and practitioners testifies to the soundness of a definition.

1.4.2 Principle 2 – Reliance on Precise Assumptions

Most modern cryptographic constructions cannot be proven secure unconditionally. Indeed, proofs of this sort would require resolving questions in the theory of computational complexity that seem far from being answered today. The result of this unfortunate state of affairs is that security typically relies upon some assumption. The second principle of modern cryptography states that assumptions must be precisely stated. This is for three main reasons:

1. *Validation of the assumption:* By their very nature, assumptions are statements that are not proven but are rather conjectured to be true. In order to strengthen our belief in some assumption, it is necessary for the assumption to be studied. The more the assumption is examined and tested without being successfully refuted, the more confident we are that the assumption is true. Furthermore, study of an assumption can provide positive evidence of its validity by showing that it is implied by some other assumption that is also widely believed.

If the assumption being relied upon is not precisely stated and presented, it cannot be studied and (potentially) refuted. Thus, a pre-condition to raising our confidence in an assumption is having a precise statement of what exactly is assumed.

2. *Comparison of schemes:* Often in cryptography, we may be presented with two schemes that can both be proven to satisfy some definition but each with respect to a different assumption. Assuming both schemes are equally efficient, which scheme should be preferred? If the assumption on which one scheme is based is *weaker* than the assumption on which the second scheme is based (i.e., the second assumption implies the first), then the first scheme is to be preferred since it may turn out that the second assumption is false while the first assumption is true. If the assumptions used by the two schemes are incomparable, then the general rule is to prefer the scheme that is based on the better-studied assumption, or the assumption that is simpler (for the reasons highlighted in the previous paragraphs).

3. *Facilitation of proofs of security:* As we have stated, and will discuss in more depth in principle 3, modern cryptographic constructions are presented together with proofs of security. If the security of the scheme cannot be proven unconditionally and must rely on some assumption, then a mathematical proof that "the construction is secure if the assumption is true" can only be provided if there is a precise statement of what the assumption is.

One observation is that it is always possible to just assume that a construction *itself* is secure. If security is well defined, this is also a precise assumption (and the proof of security for the construction is trivial)! Of course, this is not accepted practice in cryptography for a number of reasons. First of all, as noted above, an assumption that has been tested over the years is preferable to a new assumption that is introduced just to prove a given construction secure. Second, there is a general preference for assumptions that are simpler to state, since such assumptions are easier to study and to refute. So, for example, an assumption of the type that some mathematical problem is hard to solve is simpler to study and work with than an assumption that an encryption schemes satisfies a complex (and possibly unnatural) security definition. When a simple assumption is studied at length and still no refutation is found, we have greater confidence in its being correct. Another advantage of relying on "lower-level" assumptions (rather than just assuming a construction is secure) is that these low-level assumptions can typically be shared amongst a number of constructions. If a specific instantiation of the assumption turns out to be false, it can simply be replaced (within any higher-level construction based on that assumption) by a *different* instantiation of that assumption.

The above methodology is used throughout this book. For example, Chapters 3 and 4 show how to achieve secure communication (in a number of ways),

assuming that a primitive called a "pseudorandom function" exists. In these chapters nothing is said at all about how such a primitive can be constructed. In Chapter 5, we then discuss how pseudorandom functions are constructed in practice, and in Chapter 6 we show that pseudorandom functions can be constructed from even lower-level primitives.

1.4.3 Principle 3 – Rigorous Proofs of Security

The first two principles discussed above lead naturally to the current one. Modern cryptography stresses the importance of rigorous proofs of security for proposed schemes. The fact that exact definitions and precise assumptions are used means that such a proof of security is possible. However, why is a proof necessary? The main reason is that the security of a construction or protocol cannot be checked in the same way that software is typically checked. For example, the fact that encryption and decryption "work" and that the ciphertext looks garbled, does not mean that a sophisticated adversary is unable to break the scheme. Without a *proof* that no adversary of the specified power can break the scheme, we are left only with our intuition that this is the case. Experience has shown that intuition in cryptography and computer security is disastrous. There are countless examples of unproven schemes that were broken, sometimes immediately and sometimes years after being presented or deployed.

Another reason why proofs of security are so important is related to the potential damage that can result if an insecure system is used. Although software bugs can sometimes be very costly, the potential damage that may result from someone breaking the encryption scheme or authentication mechanism of a bank is huge. Finally, we note that although many bugs exist in software, things basically work due to the fact that typical users do not try to make their software fail. In contrast, attackers use amazingly complex and intricate means (utilizing specific properties of the construction) to attack security mechanisms with the clear aim of breaking them. Thus, although proofs of correctness are always desirable in computer science, they are absolutely essential in the realm of cryptography and computer security. We stress that the above observations are not just hypothetical, but are conclusions that have been reached after years of empirical evidence and experience.

The reductionist approach. We conclude by noting that most proofs in modern cryptography use what may be called the reductionist approach. Given a theorem of the form

> *"Given that Assumption X is true, Construction Y is secure according to the given definition",*

a proof typically shows how to *reduce* the problem given by Assumption X to the problem of breaking Construction Y. More to the point, the proof will typically show (via a constructive argument) how any adversary breaking

Construction Y can be used as a sub-routine to violate Assumption X. We will have more to say about this in Section 3.1.3.

Summary – Rigorous vs. Ad-Hoc Approaches to Security

The combination of the above three principles constitutes a rigorous approach to cryptography that is distinct from the ad-hoc approach of classical cryptography. The ad-hoc approach may fail on any one of the above three principles, but often ignores them all. Unfortunately, ad hoc solutions are still designed and deployed by those who wish to obtain a "quick and dirty" solution to a problem (or by those who are just simply unaware). We hope that this book will contribute to an awareness of the importance of the rigorous approach, and its success in developing new, mathematically-secure schemes.