On the Role of Definitions in and Beyond Cryptography

Phillip Rogaway

Dept. of Computer Science, University of California, Davis, California 95616, USA, and Dept. of Computer Science, Fac. of Science, Chiang Mai University, Chiang Mai, Thailand 50200. `rogaway@cs.ucdavis.edu`, `www.cs.ucdavis.edu/~rogaway`

Abstract. More than new algorithms, proofs, or technologies, it is the emergence of *definitions* that has changed the landscape of cryptography. We describe how definitions work in modern cryptography, giving a number of examples, and we provide observations, opinions, and suggestions about the art and science of crafting them.

1 Introduction

This paper was written to accompany an invited talk at a broad computer science conference, ASIAN '04, and so I will make it higher-level, more conversational, and more pedantic than a conventional paper.

I would like to begin by thanking Ajarn Kanchana Kanchanasut, and the other organizers of ASIAN '04, for their kind invitation to give a talk for this conference. It is a special treat to give a talk here, in my adoptive home of Thailand, at my adoptive university of Chiang Mai University.

My topic today is odd and ambitious. It's about something that has been a theme of my work for some fifteen years, and yet about which I have never spoken in an abstract or general way. The theme is *the importance of definitions*. My plan is to intermix abstract musings about definitions with nice examples of them. The examples, and indeed my entire perspective, are from *modern cryptography*, my area of research. But the talk isn't meant to be a survey of cryptographic definitions; it is something more personal and philosophical.

Before we really get going, I should explain that there are many uses of the word *definition*. When you say something like *let n =10* you are making a definition, and people use the word *definition* for an informal description of an idea, too. Here I'm not interested in definitions of either flavour. For this talk, *definitions* are things that specify, in a mathematically rigorous way, some significant notion in a field. You've all seen such definitions in computer science—things like *a Turing-computable function* or *a language that is NP-complete*.

Cryptography is about making *schemes* (or *mechanisms* or *protocols*) that accomplish some goal despite the attack of an *adversary*. A definition in this domain must therefore specify *what the adversary is allowed to do* as it attacks the scheme, and *when it is successful* in its attack. You will also need to define the *syntax* for the scheme—the "type" of object that you aim to make.

Definitions in cryptography emerged rather suddenly, in 1982, with the work of Shafi Goldwasser and Silvio Micali [GM]. Before then, cryptography was all about schemes and attacks, and there was no way to gain confidence in a scheme beyond that which was had when smart people failed to find an attack. What Goldwasser and Micali did was, first of all, to *define* cryptography's classical goal, message privacy (the goal of an *encryption scheme*). The [GM] definition was strong[1] and satisfying, and they proved it equivalent to very different-looking alternatives. Next they gave a *protocol* for encryption, and *proved* their protocol satisfied their definition, given a complexity-theoretic assumption. The proof took the form of a *reduction*: if the encryption protocol didn't meet the security definition, some other protocol wouldn't satisfy *its* security definition. The *definition–protocol–proof* approach has come to be called *provable security*.[2]

Provable security would dramatically change the character of my field. No longer would cryptography be solely an art; in an instant, a science of cryptography was born. Literally thousands of papers would come to be written within this framework. Nowadays, roughly half of the papers in cryptography's top conferences are in the provable-security tradition. In recent years, provable-security has come to have a large impact on practice, too, delivering concrete mechanisms like HMAC [BCK] (the message-authentication method used in your web browser) as well as high-level ideas that were silently absorbed into practice, such as the need for a public-key encryption scheme to be probabilistic [GM].

Provable security begins with definitions. It embraces what one might call the *definitional viewpoint*: the belief that our technical goals can be formally defined, and that by understanding the properties and realizations of these definitions we are better able to address our original goals.

Many people assume that a field's definitions are just to make something formal. They conclude that definitions are an incredible bore. But definitions aren't about formalism; they're about *ideas*. In making a definition in cryptography we are trying to capture, in precise language, some human notion or concern dealing with privacy or authenticity. When you ask a question like *What is an encryption scheme supposed to do?*, or *What does a digital signature accomplish?*, it is a definition that you should be aiming for in answer—not an algorithm, an example, a piece of code, or some descriptive English prose.

Definitions enable theorems and proofs, but they do more than that, and can be useful even in the absence of theorem and proofs. For one thing, definitions facilitate meaningful communication. When someone says *Here you need an encryption scheme that achieves semantic security under an adaptive chosen-ciphertext attack*, this tells you an enormous amount about what kind of object is

[1] A definition is *strong* if the adversary's job is seemingly easy. If you satisfy a strong definition you satisfy *weaker* definitions, too. If you are proving that your protocol achieves some goal, the proof will mean more if you choose a strong definition.

[2] This paragraph greatly simplifies the actual history. Papers setting the context for [GM] include [Co,DH,BBS,B,Sha,SRA]. Nearly contemporaneous provable-security work, for a different problem, is [BM,Y]. No one paper is fully responsible for the idea of doing definitions and proofs in cryptography, but [GM] played a pivotal role.

expected. I believe that a good deal of non-productive discourse in cryptography is attributable to muddled, definitionless speech.

Definitions help you to think, and shape how you think. I have seen how, often times, one can hardly get anywhere in thinking about a problem until things have been named and properly defined. Conversely, once a definition *is* spelled out, what was obscure may become obvious. I'll give an example in Section 5.

Let me now enumerate the first set of points I have made. When the spirit moves me, I'll list random claims, viewpoints, or pieces of unsolicited advice.

▷ **1** The emergence of definitions, and the definitional viewpoint, can usher in a huge transformation in the character of that field.

▷ **2** Being formal isn't the purpose of a definition. While only something precise deserves to be called a definition, the purpose lies elsewhere.

▷ **3** Definitions are about ideas. They arise from distilling intuitively held notions. They capture the central concepts of a field. They enable theorems and proofs, allow you to engage in more productive discourse, and help you to think.

▷ **4** Definitions can be worthwhile even in the absence of theorems and proofs.

2 Pseudorandom Generators

Let's cease all this abstraction and give a first example. I'll describe a lovely definition due to Blum and Micali [BM] and Yao [Y]. They ask *What does it mean to generate random-looking bits*?

Setting a high bar, we aim to produce *pseudo*random bits that are so much like the genuine article that no feasible program D can tell the difference. This might sound impossible. In fact, if we generate any fixed sequence of pseudorandom bits Y, we have no chance: for any Y there is a simple algorithm D that does a good job at telling if it is given Y or random bits. Thus our method of making pseudorandom bits must *itself* be randomized. We therefore focus on an object, a *pseudorandom generator* (PRG), that stretches a random *seed*, s, into a longer, pseudorandom string. Formally, a PRG is any function G: $\{0,1\}^n \to \{0,1\}^N$ where $N > n \geq 1$ are constants associated to G.

The above definition gives the *syntax* of a PRG. We hope that $G(s)$, for a random $s \in \{0,1\}^n$, will look like a random string of length N, but nothing like that is part of the definition. The definition might seem woefully inadequate for this reason. It tells us, for example, that the function G: $\{0,1\}^{100} \to \{0,1\}^{200}$ defined by $G(s) = 0^{200}$ is a PRG. If the intent of a PRG is that you can use $G(s)$, for a random s, in lieu of N random bits, then how is it that outputting a constant earns you the right to be called a PRG?

Despite the above, you should separately define the syntax of an object and its measure of worth. You're in no position to define what an object is supposed to until you've defined what the object is.

▷ **5** Always separate the syntax of the object you are defining from the measure of its quality.

The quality of a PRG will be captured by a real number. Actually, the number is associated not to the PRG but to the pair consisting of a PRG, G, and a *distinguisher*, D. A distinguisher is just an algorithm, possibly a probabilistic one, equipped with a way to interact with its environment. One can think of it as an *adversary*—an agent trying to break the PRG. We equip D with an *oracle* that has a "button" that D can push. Each time D pushes the button ("makes an oracle query"), it gets a string. For a PRG $G\colon \{0,1\}^n \to \{0,1\}^N$ and a distinguisher D, consider running (G, D) in either of two different *games*:

- Game 1: every time the distinguisher D makes an oracle query, choose a random string $s \in \{0,1\}^n$ and give the distinguisher $Y = G(s)$.
- Game 0: every time the distinguisher D makes an oracle query, choose a random string $Y \in \{0,1\}^N$ and give the distinguisher Y.

The distinguisher's goal is to ascertain if it is playing game 1 or game 0. To that end, when it is ready, it outputs a bit $b \in \{0,1\}$ and halts. If it thinks it is playing game 1, it should output 1, and if it thinks it is playing game 0, it should output 0. The advantage of D in attacking G is defined as $\mathbf{Adv}^{\mathrm{prg}}_G(D) = \Pr[D^{\mathrm{Game1}} \Rightarrow 1] - \Pr[D^{\mathrm{Game0}} \Rightarrow 1]$, the probability that D outputs a 1 when it is plays game 1, minus the probability that D outputs a 1 when it plays game 0. This difference measures how well D is doing.

A large advantage, like 0.9, means that the distinguisher is doing well. A small advantage, like 2^{-40}, means that it is doing poorly. A negative advantage means that the distinguisher ought to flip when it says 0 and 1.

Let's return to the example generator $G(s) = 0^{200}$ for each $s \in \{0,1\}^{100}$. That this is a bad PRG is captured by the fact that there is an efficient distinguisher D that gets high advantage in breaking G. All D has to do is to request from its oracle a single string Y. If $Y = 0^{200}$ then D outputs 1; otherwise, it outputs 0. The advantage of D is $1 - 2^{-200} \approx 1$, so D is doing great. The existence of this distinguisher shows that G is a poor PRG.

You have now seen a cryptographic definition. Creating it took us from intuition (*the bits should look random*) to a definition, $\mathbf{Adv}^{\mathrm{prg}}_G(D)$, that refines and formalizes it. The [GM,Y] notion of PRG security has proven to be deep and useful, but we will not explore that here; our goal has only been to illustrate how one turns a vague idea into a proper definition.

3 Asymptotic vs. Concrete Security

When should a PRG G be deemed *secure*? Note that it will always be *possible* to get high advantage in breaking G: consider the distinguisher D that asks for a single string Y and then returns 1 iff $Y = G(s)$ for some $s \in \{0,1\}^n$. Then $\mathbf{Adv}^{\mathrm{prg}}_G(D) \geq 1 - 2^{n-N} \geq 0.5$. But this distinguisher is extremely inefficient, even for modest n, because it needs 2^n steps to enumerate the strings of $\{0,1\}^n$. Unreasonable distinguishers will be able to get significant advantage. We would be satisfied if every *reasonable* distinguisher D earns *insignificant* advantage. How should we define *reasonable* and *insignificant*? There are two approaches, the *asymptotic* approach and the *concrete-security* approach.

The *asymptotic approach* usually equates *reasonable* with *polynomial time* and *insignificant* with *negligible*, where $\epsilon(n)$ is said to be *negligible* if for all $c > 0$ there exists a $K > 0$ such that $\epsilon(n) < n^{-c}$ for all $n \geq K$. To use this approach there needs to be a *security parameter*, n, relative to which we speak of polynomial-time or negligible advantage. For a PRG, the security parameter can be the length of the seed s. We need to go back and adjust the syntax of a PRG so that it operates on seeds of infinitely many different lengths—for example, we could redefine a PRG to be a function $G\colon \{0,1\}^* \to \{0,1\}^*$ where $|G(s)| = \ell(|s|)$ for some $\ell(n) > n$. Then we could say that a PRG G is *secure* if every polynomial-time distinguisher D obtains only negligible advantage in attacking G. The asymptotic approach is the traditional one, and all of the early definitions in cryptography were originally formulated in this way.

The *concrete-security approach* is what we illustrated in Section 2. It punts on the question of what is *reasonable* and *insignificant*, choosing never to define these terms. As a consequence, it can't define when an object is *secure*. The viewpoint is that a definition of advantage already *is* a measure of security, and making the "final step" of defining *reasonable* and *insignificant* is often unnecessary, and even artificial. It is ultimately the user of a scheme who will decide what is *reasonable* and *insignificant*, and not based on any formal definition. The concrete-security approach was popularized by Mihir Bellare and me.

You get different definitions for an object if you use asymptotic or concrete-security. I'd like to ask if the difference is *important*. I think the answer is both *yes* and *no*. We begin with *no*.

Not important. The essential idea in our treatment of PRGs transcends the definitional choice of asymptotic vs. concrete security. That idea was to think about a *distinguisher* that is asked to differentiate between two kinds of things: the result of applying the PRG to a random string; or a bunch of random bits. We measure, by a real number, how well the distinguisher does this job. At this level, the asymptotic and concrete notions coincide. In addition, a small amount of experience lets one easily translate between the two notions, and the former effectively has the latter built-in. Asymptotic-security proofs embed a concrete-security one, and, again, a little experience lets you extract it. All of this argues that asymptotic vs. concrete security is not an important difference.

There were additional choices silently built into our treatment of PRGs that seem likewise tangential. Choices like allowing multiple oracle queries, and (in the asymptotic case) making adversaries "uniform" algorithms across different values of n. In general, important definitions seem to be quite robust, but maybe not in the sense that is often assumed.

$\triangleright$ **6** Good definitions are robust, not in the sense that we can modify definitional choices and leave the object being defined unchanged, but in the sense that diverse elaborations leave intact a core definitional idea.

Important. I believe that the culture and character of modern cryptography has been dramatically influenced by the fact that early definitions were always asymptotic. The choice not only reflected shared sensibilities, it reinforced them.

Let's first ask *why* early definitions were always asymptotic. Provable security evolved within the theory community, in the intellectual tradition of other asymptotic notions like big-O notation and NP-completeness. An asymptotic treatment made for convenient discourse by defining when an object is *secure*. The convenient language helped bring out broad relationships between notions. Early workers in provable-security cryptography aimed at answering questions that seemed most fundamental and aesthetic to them, and minimalist notions, particularly *one-way functions*, were seen as the best starting point for building other kinds of objects. This pushed one towards complex and inefficient constructions, and the definitional choices that would simplify their analyses.

Cryptography might have developed quite differently if concrete security had been more prominent from the start. Concrete security encourages a higher degree of precision in stating results and exploring the relationships between notions. It is a better fit for blockciphers, which rarely have any natural security parameter, and thus a better fit for goals usually achieved using blockciphers, particularly symmetric encryption and message authentication codes. Concrete security makes for a more accessible theory, with fewer alternating quantifiers and complexity-theoretic prerequisites. It encourages a more applied theory.

Practitioners were alienated by the language of asymptotic complexity, the high-level statements of results that fell under it, and the algorithmic inefficiency that seemed endemic to early work. There emerged a pronounced culture gap between cryptographic theory and practice. Theorists and practitioners ignored one other, attending disjoint conferences. Neither group regarded the other as having anything much to say.

The asymptotic approach isn't responsible for the theory/practice gap (which is, after all, endemic to many fields), but it has exacerbated it, effectively encouraging a less relevant style of theory. When Bellare and I wanted to push provable security in a more practice-cognizant direction, we saw abandoning asymptotics as a key element of our program.

Making concrete security more visible and viable has had a big impact on the type of work that now gets done. Papers are published that give tighter analyses of existing protocols; new protocols are invented so as to admit better security bounds; notions are compared by looking at the concrete security of reductions and attacks; and blockcipher-based constructions are designed and analyzed. All of these activities are fostered by the concrete-security view.

▷ **7** Definitional choices can dramatically affect the way that a theory will develop and what it is good for. They impact the types of questions that are likely to be asked, the level of precision expected in an answer, and what background is needed to understand it.

▷ **8** Definitions arise within a particular scientific culture. They reflect the sensibilities of that culture, and they also re-enforce it, distancing it from concerns outside of that culture.

▷ **9** To change the character of work within a field, change its definitions.

4 Blockciphers

Let's next look at blockciphers, objects like DES and AES. You've all used blockciphers; they're in every ATM machine and web browser. Here I want to ask what *is* a blockcipher, and how do we measure a blockcipher's security? Our treatment is based on that of Bellare, Kilian, and Rogaway [BKR] which, in turn, builds on Goldreich, Goldwasser, and Micali [GGM] and Luby and Rackoff [LR].

Again beginning with syntax, a blockcipher is a function $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ where $\mathcal{K}$ is a finite, nonempty set (the *key space*) and $n \geq 1$ is a number (the *blocksize*) and $E(K, \cdot)$ is a permutation (on $\{0,1\}^n$) for each $K \in \mathcal{K}$. When $Y = E(K, X)$ we call X the *plaintext-block* and Y the *ciphertext-block*.

For measuring security, there are lots of adversarial goals that one might focus on. Goals like an adversary's inability to recover plaintext blocks from ciphertext blocks, or its inability to recover a key K from (X_i, Y_i)-pairs, where $Y_i = E(K, X_i)$. But experience leads in a different direction.

To define blockcipher security it is useful to distinguish the *model* (or *attack model*) from the *goal*. The model says what the adversary can do—how the system runs in the adversary's presence. The goal says what the adversary is trying to accomplish as it operates within the model. For our attack model, we consider an *adaptive, chosen-plaintext attack*: the adversary can get the ciphertext block for any plaintext block that it names, and each plaintext block that the adversary asks about may depend on the ciphertext blocks that it has received. For the goal, we'll say that the adversary is trying to distinguish the ciphertext blocks that it receives from an equal number of random, distinct strings.

Our notion of security, what is called PRP (pseudorandom permutation) security, associates a real number to a blockcipher E and a distinguisher D. In game 1, a random key $K \in \mathcal{K}$ is chosen and then, whenever the adversary asks its oracle a question $X \in \{0,1\}^n$, we return $Y = E(K, X)$. In game 0, a random permutation π is chosen among all the permutations from $\{0,1\}^n$ to $\{0,1\}^n$ and then, whenever the adversary asks its oracle a question X, we return $\pi(X)$. Distinguisher D wants to figure out if it is playing game 1 or game 0. It outputs a bit $b \in \{0,1\}$ which is its guess. We define D's advantage, $\mathbf{Adv}_E^{\mathrm{prp}}(D)$, as $\Pr[D^{\mathrm{Game1}} \Rightarrow 1] - \Pr[D^{\mathrm{Game0}} \Rightarrow 1]$, the difference in the probabilities that D outputs 1 when it plays games 1 and 0. Intuitively, a blockcipher E is "good" if no reasonable distinguisher D gets high prp-advantage.

The reader may notice that what I called the *model* has sort of vanished in the actual definition. I never formally defined the model, and it effectively got merged into the definition of the goal. This situation is common. Thinking in terms of an attack model is an important precursor to doing a definition, but the attack model might get abstracted away. For simple definitions, like blockcipher security, this is not a problem. For complex definitions, the model should be left intact, for the definition will be easier to understand.

$\triangleright$ **10** It is often useful to develop a model prior to giving a definition. Later, the definition might absorb the model. For complex goals, formalize the model and have it remain intact in the definition.

Our definition of PRP security may seem unrealistically strong. Why should the adversary be allowed to make arbitrary queries to an enciphering oracle when, in typical usages of a blockcipher, its capabilities will be much more constrained? The answer is, in large part, that it works. First, good PRP security seems to be achieved by objects like AES; the assumption doesn't over-shoot what we can efficiently create. Second, the PRP definition is strong enough to give rise to simple and provably-good constructions. Finally, natural, weaker definitions have not been shown effective for designing new constructions, nor in justifying popular constructions that predate notions of PRP security. The above experience effectively becomes the rationale for the definition. It would be wrong to focus on the fact that the definition gives the adversary unrealistic power, because the definition was never intended to directly model an adversary's real-world capabilities in attacking some particular use of the blockcipher.

$\triangleright$ **11** In defining low-level primitives, simple, easily used, pessimistic definitions are better than more complex and possibly more faithful ones.

The PRP definition of blockcipher security has by now been used in numerous papers. This fact, all by itself, is the best evidence that a definition is doing its job. Definitions are best evaluated retrospectively.

If you want to make a definition of value, you need a market. A definition has to formalize a notion that people would like to have defined, and it has to do so in a way of use to that community. Of course *all* scientific work should be done with the interests of some community in mind. But for a definition, what will best serve this community has to be the focus of ones concern.

$\triangleright$ **12** The primary measure of a definition's worth is how many people and papers use it, and the extent to which those people and papers say interesting things. A definition is crafted for the benefit of some community.

5 Authenticated Encryption

Authenticated encryption (AE) allows a sender to transmit a message to a receiver in such a way as to assure both its privacy and authenticity. The sender and the receiver share a secret key K. The AE goal has been known to cryptographic practice for decades, but it was only recently provided with a definition and provable-security treatment [KY,BR1,BN]. I'll follow the nonce-based treatment from [RBB]. To capture privacy, we'll formalize that ciphertexts are indistinguishable from random strings. To capture authenticity, we'll formalize that an adversary can't devise authentic messages beyond those that it has seen.

Beginning with syntax, an AE-scheme is a pair of deterministic algorithms $(\mathcal{E}, \mathcal{D})$ where $\mathcal{E}\colon \mathcal{K} \times \mathcal{N} \times \{0,1\}^* \to \{0,1\}^*$, $\mathcal{D}\colon \mathcal{K} \times \mathcal{N} \times \{0,1\}^* \to \{0,1\}^* \cup \{\perp\}$, sets $\mathcal{K}, \mathcal{N} \subseteq \{0,1\}^*$ are nonempty and finite, $|\mathcal{E}(K,N,M)| = |M| + \tau$ for some constant τ, and $\mathcal{D}(K,N,C) = M$ whenever $C = \mathcal{E}(K,N,M)$. Algorithms $\mathcal{E}$ and $\mathcal{D}$ are called the encryption algorithm and the decryption algorithm, and strings K, N, M, and C are called the key, nonce, plaintext, and ciphertext.

The nonce is supposed to be a non-repeating value, such as a counter, and the $\perp$ symbol is used to indicate that the ciphertext is inauthentic.

To quantify security we associate a real number to an AE-scheme $(\mathcal{E}, \mathcal{D})$ and an adversary A. This is done by imagining two different "contests" that A may enter, a privacy contest and the authenticity contest. The adversary may enter either contest, getting a score. Our measure of security is the real number $\mathbf{Adv}^{\mathrm{ae}}_{(\mathcal{E},\mathcal{D})}(A)$ which is the score that A earns from the contest that it enters.

For the privacy contest the adversary A plays one of two games. In game 1, a key $K \in \mathcal{K}$ is randomly chosen at the beginning of the game. Then, when the adversary asks an oracle query of (N, M) it gets back the string $C = \mathcal{E}(K, N, M)$. In game 0, when the adversary asks an oracle query of (N, M) it gets back a random string C of length $|M| + \tau$. Playing either game, the adversary may not repeat an N-value. When it is ready, the adversary outputs a bit $b \in \{0, 1\}$. The score that adversary gets is $\Pr[A^{\mathrm{Game1}} \Rightarrow 1] - \Pr[A^{\mathrm{Game0}} \Rightarrow 1]$, meaning the difference in the probabilities that A outputs 1 in games 1 and 0. Intuitively, adversary A is trying to distinguish if it is receiving actual ciphertexts for the plaintexts that it asks about (game 1) or an equal number of random bits (game 0).

For the authenticity contest, the adversary again has an oracle, but this time the oracle always behaves according to the above game 1: a key $K \in \mathcal{K}$ is randomly chosen and each query (N, M) is answered by $C = \mathcal{E}(K, N, M)$. As before, the adversary may not repeat an N-value as it interacts with its oracle. When the adversary is ready, it outputs a *forgery attempt* (N, C). The adversary A is said to *forge* if $\mathcal{D}(K, N, C) \neq \perp$ and A never asked an oracle query (N, M) that returned C. The adversary's score is the probability that it forges. Intuitively, adversary A is trying to produce a valid ciphertext that is different from any ciphertext that it has seen.

Authenticated encryption is an important goal of shared-key cryptography, and the prerequisites for defining it and investigating it have been available for a long time. Why would a definition for AE wait until the year 2000 to first appear? There are several answers, but I think that the most important one is that nobody noticed there was anything that needed to be defined. There was already a notion for the privacy of a shared-key encryption scheme, and there was already a notion for what it means to create a good tag, or *message authentication code*, to ensure a message's authenticity. People understood that if you wanted to achieve privacy *and* authenticity, you just did both things. It seemed to fall beneath anyone's radar that one would still need to *prove* that the composite mechanism worked; that there are several ways to do the combining and they *don't* all work; and one couldn't even speak of the composite scheme working, or not, until there was a definition for what the composite scheme was supposed to do. If you think too much in terms of mechanisms, not definitions, then you may fail to notice all of this. In general, the first step in creating a definition is to notice that there is something that needs to be defined. Provable-security cryptography began with the realization that privacy needed to be defined, which was already a crucial and non-obvious observation.

$\triangleright$ **13** It is easy to overlook that something needs to be defined.

Shortly after AE was defined, a new AE-scheme was put forward by Jutla [J]. By better blending the parts of the algorithm used for privacy and for authenticity, Jutla was able to construct a scheme that achieves the AE-goal with nearly the efficiency of traditional, privacy-only mechanisms. Having a definition for AE was essential for figuring out if the scheme was correct.

In the absence of a definition for AE, one can easily design schemes that look sound but aren't. Indeed following the publication of Jutla's scheme and others, the U.S. National Security Agency (NSA) released, through NIST, its own scheme for efficient AE. The mechanism was called *dual counter mode*. I myself read the definition of the mode and broke it in less than a day, privately informing NIST. Donescu, Gligor, and Wagner likewise broke the scheme right away [DGW]. What is it that we knew that the folks from the super-secret NSA did not? The answer, I believe, is *definitions*. If you understood the definition for AE, it was pretty obvious that the NSA scheme wouldn't work. In the absence of understanding a definition, you wouldn't see it. Definitions, not initially intended to help people find attacks, nonetheless do just that. By specifying the rules and goal of the game, the attacker can think more clearly about strategies that conform to the rules but violate the goal. It seems to be the norm that the first person to set forth the definition for any complicated, unformalized goal will also break, without much effort, all the existing protocols for that goal. With no clear definition in mind, inventors can't do their job well.

▷ **14** Having definitions makes it easier to come up with attacks.

I would like to end this section by emphasizing that a definition like that we have given for AE does not suddenly appear; it is part of an evolving line of definitions. Here the sequence of definitional ideas flow from [GM] and [GMRi] to [BKR] and [BDJR] to [BR1,KY,BN] and [RBB]. In general, definitions in cryptography seem to be constantly evolving. They evolve for several reasons. New problems get provable-security treatments (oftentimes these problem having already been considered by practitioners). Shortcomings are discovered in prior definitions—oversights, undesirable consequences, or outright bugs. More elegant ways to do old definitions are discovered. Or the intended use of a definition changes, motivating a different way of doing things. That definitions evolve doesn't contradict point ▷ 6, which might help you to feel better when your definitional choices get antiquated.

▷ **15** Definitions emerge, change, and die more than people think.

6 Session-Key Distribution

In a distributed system, communication between parties typically takes place in *sessions*, a relatively short period of interaction between two parties, protected by an associated *session key*. A party can maintain multiple sessions, even to a particular partner. A protocol for *session-key distribution* (SKD) aims to securely distribute a pair of session keys. There are several *trust models* for SKD, a trust model saying who has what keys. Session-key distribution is addressed by the Kerberos and SSL/TLS protocols.

Definitions for SKD and the related problem of *entity authentication* begin with Bellare and Rogaway [BR2], continuing with work like [BR3,BPR,CK,Sho]. Models and definitions in this domain are more complex than those we've met so far and so, for concision, we will have to be less thorough. Our description is loosely based on [BPR], following [BR3,BR2].

Our model for SKD pessimistically assumes that all communication among parties is under the adversary's control. It can read messages produced by any party, provide its own messages to them, modify messages before they reach their destination, delay or replay them. It can start up entirely new *instances* of any party. We'll also let the adversary learn already-distributed session keys, and we'll let it *corrupt* parties, learning all that they know.

To capture all of these possibilities, we imagine providing an adversary A with an infinite collection of oracles. See Fig. 1. There's an oracle Π_i^s for each party i and each natural number s. Oracle Π_i^s represents instance s of party i. Each oracle has its own private coins and its own state, which it remembers between calls. Oracle initialization depends on the trust model.

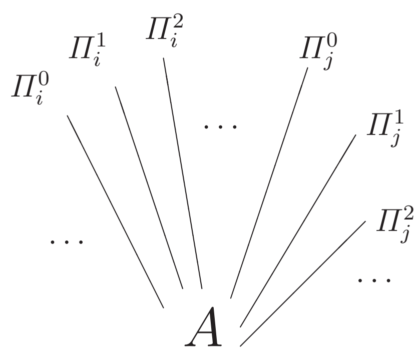

Fig. 1. *The model for session key distribution envisages an adversary, A, in a sea of oracles. Oracle Π_i^s models instance s of party i. Each oracle runs the SKD protocol.*

The adversary's capabilities are embodied by the types of oracle queries we permit, and how we say to answer them. There are three types of queries corresponding to the adversarial capabilities we have mentioned.

(1) $\mathsf{Send}\ (i,\ s,\ M)$ — This sends message M to oracle Π_i^s. The oracle computes what the protocol says to, and sends back the response. Should the oracle *accept*, regarding itself as having now arrived at a session key, this fact is made visible to the adversary, along with a value called a *session-id* (SID), and a *partner-id* (PID). The actual *session key* that the oracle holds, *sn*, will not be visible to

the adversary. To initiate the protocol between an initiator i and a responder j, the adversary should send message $M = j$ to an unused instance of i.

(2) $\mathsf{Reveal}\,(i,\ s)$ — If oracle Π_i^s has accepted, holding some session key *sn*, then this query returns *sn* to the adversary. This query will be used to capture the idea that loss of a session key shouldn't be damaging to other sessions.

(3) $\mathsf{Corrupt}\,(i, K)$ — This is directed against a party, not an instance of a party. The adversary obtains any long-lived or secret keys associated to party i, as well as the private states of all utilized instances of i. The query may also be used to replace any keys under i's control; this is the role of the argument K.

With a model now sketched out, let's describe a measure for adversarial success. The idea is to test if the adversary can distinguish a session key embedded inside an oracle from a *random* session key drawn from the same distribution. Thus we extend the model above to include a Test query, $\mathsf{Test}(i, s)$, directed to some oracle Π_i^s. There are two ways to answer this query. In game 1, a query $\mathsf{Test}(i, s)$ returns the session key *sn* inside oracle Π_i^s. In game 0, it returns a random session key *sn* drawn from the distribution that session keys are supposed to be drawn from. Both cases require that Π_i^s has accepted. The advantage of the adversary, $\mathbf{Adv}_{\Pi}^{\mathrm{skd}}(A)$, is, once again, the probability that A outputs 1 in game 1 minus the probability that it outputs 1 in game 0.

If you're still with me, you should complain that this definition makes no sense, because the adversary can *trivially* figure out the session key *sn* in an oracle Π_i^s. In fact, there are multiple ways to do this. First, the adversary can make a Corrupt query to party i. Alternatively, it can make a Reveal query to instance Π_i^s. More problematic still, an SKD protocol is *supposed* to distribute a session key to a pair of oracles, so when Π_i^s accepts, there may well be an oracle Π_j^t (different from Π_i^s) that *should* have the same session key *sn* as Π_i^s. A Corrupt query to j or a Reveal query to Π_j^t would reveal the session key of Π_i^s and make it easy to distinguish games 0 and 1.

We handle this issue by saying that an oracle Π_i^s starts off *fresh*, but ceases to be so if there is a call $\mathsf{Corrupt}(i, K)$, $\mathsf{Reveal}(i, s)$, $\mathsf{Corrupt}(j, K)$, or $\mathsf{Reveal}(j, t)$, where Π_j^t is *partnered* to Π_i^s. The oracles are partnered if each has accepted and they share the same SID. In addition, we demand that only they have that SID, one oracle is an initiator and the other is a responder, and Π_i^s has a PID of j while Π_j^t has a PID of i. Now we simply demand that the adversary perform its Test query on an oracle that is *fresh*; choosing an unfresh oracle earns no credit.

One aspect of the definitional approach above is seen often enough to single out: the idea of lifting something unavoidable into a definition. That is, a common definitional motif is to formalize that which one can not avoid, and then capture the idea that there is nothing beyond that of significance.

As before, a key step in arriving at a definition for SKD was to realize that there was a significant goal to be defined. I learned this from talking to security architects when I worked at IBM. Back then, the security folks were interested in something called *Kerberos*, which I had never heard of. Nobody could adequately explain to me what problem this *Kerberos* thing aimed to do. It was frustrating. Either the security people were nuts, obsessing on some non-existent or trivial problem, or else they weren't. I decided to assume the latter.

$\triangleright$ **16** Practice that has not yet met theory is an excellent place for which to craft definitions. Practitioners won't be focusing on a pointless problem.

Soon after the appearance of [BR3], Rackoff came up with an example showing how our definition was not strong enough to guarantee security for certain applications that would use the distributed session key. We traced the problem to a simple issue: we had wrongly made the restriction that the adversary's Test query would be its final query. Removal of this restriction (as in the SKD notion described above) solved the problem.

The definition in [BR3] actually has a more basic problem. Our initial approach to defining partnering, used in [BR2], depended on the idea of a *matching conversation*: oracles Π_i^s and Π_j^t were said to be partnered if they engaged in conversations that are consistent with messages being faithfully relayed between them (except that the last message might not be delivered). This worked fine, but it focused on something that seemed syntactic and fundamentally irrelevant. So in [BR3] we tried something different, assuming an existentially guaranteed partnering function on the global transcript of Send-queries. Almost immediately, I regretted this choice. It did not carry with it the strong intuition of matching conversations, and there was no reason to think that it was the "right" way to identify a party's partner. The notion was hard to explain and hard to use.

Our exposition of $\mathbf{Adv}_{\Pi}^{\mathrm{skd}}(A)$ used SIDs to define partnering. That idea sprang from discussions in 1995 among Bellare, Petrank, Rackoff, and me. An explicit SID seems more intuitive and elegant than an existentially-guaranteed partnering functions, and less syntactic than matching conversations, so we switched to that approach in [BPR]. A recent paper by Choo, Boyd, Hitchcock, and Maitland [CBHM] correctly criticizes how partnering was done in [BR3].

Overall, my experience, shared by others, has been that it is hard to get complex definitions exactly right. I might even venture to say that initial definitional attempts will usually have sub-optimal choices, unexpected consequences, or outright bugs. If the definition is important, the issues will eventually get identified and cleaned up.

$\triangleright$ **17** Definitions, when they are first proposed, will often encompass poor decisions or errors. For a good definition, these don't greatly diminish the value of the contribution.

$\triangleright$ **18** Definitional choices that don't capture strong intuition are usually wrong, and may come back to haunt you.

Let me make one final point in this section. Our first paper in this space [BR2] was mostly on entity authentication (EA), not SKD, since EA was the more popular and well-known problem of the day. An EA protocol lets two parties have a conversation at the end of which each knows that he has just spoken to the other. (This is the *mutual authentication* version of the problem.) The model is nearly the same for EA and SKD. In working on EA, something that disturbed me greatly was that the problem seemed, to me, to be nearly pointless. If Alice learns that, a moment ago, Bob was present, of what value is that information? Bob may not be present anymore, and even if he is, Alice has learned nothing that will help her to have a continuing conversation. What kind of egotists are Alice and Bob that they care about saying *I am here* and nothing more?

There is some justification for working on EA. One is that I have probably overstated the pointlessness of the goal.[3] Another is that a problem becomes significant *because* people have agreed to focus on it. Still, I'm not so convinced.

▷ **19** The fact that a definitional goal is nearly pointless doesn't seem to bother people nearly as much as it should.

7 The Random-Oracle Model

The random-oracle (RO) model goes back to Fiat and Shamir [FS], but was popularized and made into an explicit design paradigm by Mihir Bellare and me [BR4]. The idea is to design definitions and protocols in an embellished model of computation in which all parties, including the adversary, are given access to a common random-oracle. This is a map H: $\{0,1\}^* \to \{0,1\}$ that associates a random bit to each and every string. One proves the protocol correct in this enriched model of computation. Then, as a heuristic final step, one instantiates the oracle H by something like a cryptographic hash function.

Experience has shown that many cryptographic problems can be solved within the RO-model by protocols that are simpler and more efficient than their best-known standard-model counterparts. For this reason, the RO-model has become popular for doing practical protocol design.

The RO-model has been the locus of much controversy. The question is what assurance, if any, should be invested in an RO-result.[4] The concern was first developed in a paper by Canetti, Goldreich, and Halevi [CGH], who give an RO-secure protocol whose standard-model counterpart is always insecure.

To some people, proofs in the RO-model are effectively not proofs. One well-known researcher calls them *heuristic arguments.* That isn't right. A proof in the RO-model is still a proof, it's just a proof in a model of computation that some people don't find worthwhile.

There is certainly a difference between defining a measure of worth for an object and modeling that object. But the modeling-approach isn't less scientific, and it is not at all clear that it yields something whose real-world significance is vastly inferior.

---

[3] In a smartcard setting, for example, EA may be exactly what you want.

[4] Our own paper was rather guarded on the question, but it did make it a thesis that there is value in an RO-result, and that RO-based design is better than design without definitions and proofs [BR4].

When you are working within the RO-model you are working within a specific model, and a not-so-realistic one at that. What is often not recognized is that when you are working within the standard model, you are *also* working within a specific model, and a not-so-realistic one. The standard-model *also* abstracts away key aspects of the real world—like the fact that real computation takes time, uses power, and leaks radiation. There *is* a big gap between the RO-model and reality (hash functions *aren't* like random oracles)—and there is *also* a big gap between the standard model and reality. Some recent work by Micali and Reyzin aims to close the latter gap [MR].

In cryptography, we are in the business of making models. We should always be skeptical that these models are accurate reflections of the world. Models always embed unrealistic and ultimately bogus assumptions, and yet, somehow, they often work. This is the wonder of scientific abstraction.

$\triangleright$ **20** The distinction between modeling an object and defining an object is real, but its impact is often overstated. Definitions always embed a model and are always subject to model-based limitations. Modeling something is not less scientific than defining it.

The nothing-but-the-standard-model sentiment that is behind some of the RO-criticism is the same sentiment that led to the partitioning of cryptography following Dolev and Yao [DY]. That paper's sin was to model encryption, not define it, and the prevailing sentiment within my community has been to say that such a thing is not real cryptography, and doesn't belong in our venues. A separate community emerged that works on cryptographic protocols, but where the primitives of the protocols are modeled, not defined. The partitioning of cryptography was unhealthy. The field is better off when a diversity of models are encouraged, when they vie for space at the same conferences, and when the relationships among different model becomes a significant point of attention.

8 Closing Comments

No paper on definitions in contemporary cryptography can ignore the emergence of general definitions for *secure protocols* by Backus, Pfitzmann, and Waidner [BPW1] and Canetti [Ca]. These ambitious works define when one protocol is *at-least-as-secure* as another. In doing so, they provide a framework for defining arbitrary protocol goals: a protocol for a given goal is *secure* if it is at-least-as-secure as the *ideal protocol* for that goal. In this way a description of the ideal protocol for some task (e.g., secure message transmission, digital signature, or electronic voting) yields a definition of security for that task. The definitions of [BPW1,Ca] build on work that includes [Bea,GMRa,GMW].

The [BPW1,Ca] line of work represents a particularly important advance. It simultaneously defines a fundamental concept in our field and holds out the promise for more rigorously and manageably treating a variety of ambitious protocol problems. The latter hope stems, in part, from a focus on *composability*, a property built into the notion of security and intended to enable modular design and proofs. It is my view that cryptography is in a sort of "crisis of complexity" for many of the tasks that people now consider: as goals get more complex, what people call a definition and proof ceases, for me, to be rigorous and convincing. General definitional frameworks could improve the situation, and [BPW1,Ca] seem to be having considerable success. See [BPW2] as an impressive example.

Still, I am not without doubts. First, the definitions of [BPW1,Ca] are long and complex. Despite this, there are problems of precision and understandability. To be useful, definitions need to be clear and succinct. Second, the definitions that one arrives at by way of the [BPW1,Ca] framework seem less intuitive and less prescriptive than what one gets by a more direct route. There is an assumption that they are strictly stronger, but such claims are not always proven, and I doubt they're always true. Third, [BPW1,Ca]-derived definitions always involve a simulator. For some goals, simulatability does not seem to correspond to any intuitive understanding of the goal. Simulatability captures strong intuition when it is used to define zero knowledge, but when it is used to define a problem like secure key-distribution or bit commitment, it seems to play a much more technical role. Finally, I see no reason to think that all definitional considerations appropriate to a particular problem domain can be captured by the choices implicit in defining the ideal protocol. For example, is the presence or absence of receipt-freeness[5] in a voting protocol captured by what one does in the ideal voting protocol?

▷ **21** General definitional frameworks [BPW1,Ca] for secure protocols are an important direction—probably the most important definitional activity currently going on in cryptography. But it remains unclear how this will play out.

We are nearing the end of our journey, and I find that I have barely mentioned what is one of the most beautiful definitions from cryptography: the idea of *zero knowledge* (ZK), due to Goldwasser, Micali, and Rackoff [GMRa]. Zero knowledge is a testament to the power of a definition: it has created an area of its own, giving rise to derivative notions and impacting both cryptography and complexity theory. The notion of simulatability that came with ZK has spread across cryptography, becoming a central notion of the field. Briefly and informally, communications with a party P is *zero knowledge* if that which is seen from interacting with P can be created, just as effectively, *without* involving P.

It may sound flip, but I want to acknowledge that *zero knowledge* is a beautiful name. In just four syllables, it promises intrigue and paradox. How can something be *knowledge* and yet be *zero*? How can one be speaking of a mathematical notion of *knowledge* in the first place?

Names are important. They need to be succinct and suggestive. Accuracy is good, but it's not as important. (After all, you are going to provide the word or phrase with a definition; a name all by itself can't be expected to have a precise meaning.) The phrase *minimum disclosure proof* [BCC] is more accurate than *zero knowledge*, but it's nowhere near as good a term.

[5] A voting protocol is receipt-free if parties can't prove to someone whom they voted for [BT]. The property is desirable to help avoid coercion.

Sometimes it is hard to anticipate what will make a good name. In [BDPR] we needed a compact name for two flavours of chosen-ciphertext attack (CCA), a weak form and a stronger form. We tried everything, but nothing seemed to work. With much reluctance, we published using the terms CCA1 and CCA2. Amazingly, the terms caught on. I've even come to like them. Every time I see CCA2 in a talk or paper, I smile. Other names I am happy with are *plaintext awareness* and an *AXU-hash function* (almost-xor-universal).

The oddest thing can change a name from being terrible to OK. I used to regard *universal one-way hash function*, UOWHF, as one of the worst names ever invented for a lovely object [NY]. It is cumbersome without accurately suggesting what the object does. I warmed up to the name after hearing Victor Shoup give a talk in which he pronounced UOWHF as *woof*. Once there was a way to say UOWHF, and a fun way, the name seemed immeasurably better.

Notation should be chosen with great care. Names and notation are the currency of thought. Our minds can be imprisoned by poor notation and set free by good notation. Mathematics blossomed only after its notation matured. It is painful to read papers with ill-chosen notation but, more than that, it is hard to think deeply about things until the names and notation are right.

Good names and good notation are just one aspect of good writing, and it is the entirety of technical writing that is important. I reject the viewpoint that the ideas of a paper and the presentation of a paper are fundamentally distinct and orthogonal. A paper is, first and foremost, pages full of marks, not some transcendent concept of what the thing's about. A paper *is* its presentation.

While the quality and impact of any scientific paper is intimately connected to the quality of its writing, I believe that this is *especially* true for papers that aim to make a definitional contribution. These are simultaneously harder to write well and more important to write well. Writing definitions is hard because, in part, there will be no existing example for how best to communicate your idea. There is an extra premium on concision and elegance, and a huge penalty for ambiguity. If a proof is poorly written, people will skip over it or convince themselves that the result is right and they could re-prove it. It usually won't render the paper pointless. If a definition is poorly written and can't readily be understood, nothing that follows will make any sense. Nobody will be interested to do follow-on work. The value of the paper vanishes.

If you write a paper that gives a new definition, you need to carefully explain it. Your definitional choices should be justified. But the exposition shouldn't intermix a definition with descriptive prose; these need to be clearly separated.

In a paper that uses an existing definition, you should fully state that definition, not just reference it. A failure to do this will typically make your paper meaningless without tracking down the other work, which the reader shouldn't have to do. What's more, the paper you reference might exist in multiple versions, and small differences between their definitions can have a huge impact.

▷ **22** Names and notation matter. Choose them well.

▷ **23** Good writing, always important for a scientific paper, is even more important when the paper makes a definitional contribution.

I will wrap up. Of the different kinds of work that I have done, it is the definition-centric/notion-centric work that I believe to have the most value. Mechanisms come and go, are improved upon, rarely become popular, and are never really basic and compelling enough to bring satisfaction. Theorems are ignored or strengthened, forgotten, and hardly anyone reads their proofs or cares. What lasts, at least for a little while, are the notions of the field and that which is associated to making those notions precise: definitions.

I believe that definitions have been the most important thing to bring understanding to my field and make it into a science. Though I have no particular evidence for it, it is my guess that definitions can play a similar role in other areas of computer science too, where they have not yet played such a role. And that is the real reason that I have chosen this topic for today—that I might, possibly, help infect some other area of computer science with definitions and the definitional-viewpoint.

Acknowledgements

Most of what I understand about definitions I learned from my former advisor, Silvio Micali, who is a master of crafting them. If there is anything right and of value in this note, it probably stems from Silvio. My frequent co-author, Mihir Bellare, shares with me a fondness for definitions and has no doubt influenced by sensibilities about them. I received helpful comments from Mihir Bellare, John Black, Chanathip Namprempre, and Tom Shrimpton. This note was written with funding from NSF 0208842 and a gift from Cisco Systems. Most especially for this odd paper, it goes without saying that all opinions expressed here are my own.

References

[B] M. Blum. Coin flipping by phone. *IEEE Spring COMPCOM*, pp. 133–137, 1982.

[BBS] L. Blum, M. Blum, and M. Shub. A simple secure unpredictable unpredictable pseudo-random number generator. *SIAM J. on Computing*, vol. 15, pp. 364–383, 1986.

[BCC] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *JCSS*, vol. 37, no. 2, pp. 156–189, 1988.

[BCK] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. *Crypto '96*, LNCS vol. 1109, Springer, 1996.

[BDJR] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption: analysis of the DES modes of operation. *FOCS 1997*, 1997.

[BDPR] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. *Crypto '98*, LNCS vol. 1462, Springer, 1998.

[Bea] D. Beaver. Secure multiparty protocols and zero-knowledge proof systems tolerating faulty minority. *J. of Cryptology*, vol. 4, no. 2, pp. 75–122, 1991.

[Bel] M. Bellare. Practice-oriented provable security. *Lectures on Data Security 1998*, LNCS vol. 1561, pp. 1–15, 1999.

[BGW] M. Ben-or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. *STOC 1988*, ACM Press, pp. 1–10, 1988.

[BKR] M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. *JCSS*, vol. 61, no. 3, pp. 262–399, 2000.

[BM] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. on Computing,* vol. 13, no. 4, pp. 850–864, 1984. Earlier version in *FOCS 1982*.

[BN] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *Asiacrypt '00*, LNCS vol. 1976, Springer, pp. 531–545, 2000.

[BPR] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. *Advances in Cryptology —Eurocrypt '00*. LNCS vol. 1807, Springer, 2000.

[BPW1] M. Backes, B. Pfitzmann, and M. Waidner. Secure asynchronous reactive systems. Cryptology ePrint report 2004/082, 2004. Earlier version by Pfitzmann and Waidner in *IEEE Symposium on Security and Privacy*, 2001.

[BPW2] M. Backes, B. Pfitzmann, and M. Waidner. A universally composable cryptographic library. Cryptology ePrint report 2003/015, 2003.

[BR1] M. Bellare and P. Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient encryption. *Asiacrypt '00*, LNCS vol. 1976, Springer, pp. 317–330, 2000.

[BR2] M. Bellare and P. Rogaway. Entity authentication and key distribution. *Crypto '93*. LNCS vol. 773, pp. 232–249, Springer, 1994.

[BR3] M. Bellare and P. Rogaway. Provably secure session key distribution: the three party case. *STOC 1995*, pp. 57–66, 1995.

[BR4] M. Bellare and P. Rogaway. Random oracle are practical: a paradigm for designing efficient protocols. *Conference on Computer and Communications Security, CCS '93*, pp. 62–73, 1993.

[BT] J. Benaloh and D. Tuinstra. Receipt-free secret ballot elections. *STOC 1994*, pp. 544–553, 1994.

[Ca] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. Cryptology ePrint report 2000/67, 2001. Earlier version in *FOCS 2001*.

[CBHM] K. Choo, C. Boyd, Y. Hitchcock, and G. Maitland. On session identifiers in provably secure protocols, the Bellare-Rogaway three-party key distribution protocol revisited. *Security in Communication Networks '04*, LNCS, Springer, 2004.

[CGH] R. Canetti, O. Goldreich, and H. Halevi. The random oracle methodology, revisited. *JACM*, vol. 51, no. 4, pp. 557–594, July 2004.

[CK] R. Canetti and H. Krawczyk. Universally composable notions of key exchange and secure channels. *Eurocrypt '02*, LNCS vol. 2332, pp. 337–351, Springer, 2002.

[Co] S. Cook. The complexity of theorem-proving procedures. *STOC 1971*, ACM Press, pp. 151–158, 1971.

[DGW] P. Donescu, V. Gligor, and D. Wagner. A note on NSA's Dual Counter Mode of encryption. Manuscript, 2001. Available from Wagner's webpage.

[DH] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. on Inf. Th.*, vol. 22, pp. 644-654, 1976.

[DY] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Trans. on Information Theory*, vol. 29, no. 12, pp. 198–208, 1983.

[FS] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. *Crypto '86*, LNCS vol. 263, pp. 186–194, Springer, 1986.

[GGM] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions, *JACM*, vol. 33, no. 4, 210–217, 1986.

[GM] S. Goldwasser and S. Micali. Probabilistic encryption. *JCSS*, vol. 28, pp. 270–299, 1984.

[GMRa] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. on Computing*, vol. 18, no. 1, pp. 186–208, 1989.

[GMRi] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. on Computing,* vol. 17, no. 2, pp. 281–308, 1988.

[GMW] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game, or a completeness theorem for protocols with honest majority. *STOC 1987*, pp. 218–229, 1997.

[Go1] O. Goldreich. The foundations of modern cryptography. Manuscript, 2000. Available from Goldreich's webpage. Earlier version in *Crypto 97.*

[Go2] O. Goldreich. *The Foundations of Cryptography*. Cambridge University Press. Volume 1 (2001) and Volume 2 (2004).

[J] C. Jutla. Encryption modes with almost free message integrity. *Eurocrypt '01*, LNCS vol. 2045, Springer, pp. 529–544, 2001.

[KY] J. Katz and M. Yung. Unforgeable encryption and adaptively secure modes of operation. *Fast Software Encryption, FSE 2000.* LNCS vol. 1978, Springer, pp. 284–299, 2000.

[LR] M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. on Computing*, vol. 17, no. 2, April 1988.

[MR] S. Micali and L. Reyzin. Physically observable cryptography. *TCC 2004*, LNCS vol. 2951, Springer, pp. 278–296, 2004. Cryptology ePrint report 2003/120.

[NY] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. *STOC 1989*, pp. 33–43, 1989.

[RBB] P. Rogaway, M. Bellare, and J. Black. OCB: a block-cipher mode of operation for efficient authenticated encryption. *ACM Transactions on Information and System Security* (TISSEC), vol. 6, no. 3, pp. 365–403, 2003.

[Sha] C. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, vol. 28, pp. 656–715, 1949.

[Sho] V. Shoup. On formal methods for secure key exchange. Cryptology ePrint report 1999/012, 1999.

[SRA] A. Shamir, R. Rivest, and L. Adleman. Mental poker. MIT/LCS report TM-125, 1979.

[Y] A. Yao. Theory and applications of trapdoor functions. *FOCS 1982*, pp. 80–91, 1982.