Practice-Oriented Provable Security and the Social Construction of Cryptography

Phillip Rogaway | University of California, Davis

Practice-oriented provable security serves as the backdrop for what might be called a constructionist view of cryptography—that cryptographic work isn't a consequence of what a disciplinary community aims to study so much as a reflection of its own sensibilities.

Cryptographers used to associate *provable security* with public-key cryptography, asymptotic analyses, and inefficient proof-of-concept designs. These associations are outdated, undermined by the development of *practice-oriented* provable security by Mihir Bellare and me.

Here I tell the story of practice-oriented provable security from a rather personal and sociological perspective. My aim is to use the area's emergence to bolster the claim that, overwhelmingly, cryptography is socially constructed.

Constructionism and Realism

What does it mean to say that cryptography is socially constructed? To say that some thing, *C*, is socially constructed emphasizes that *C* need not be as it is today. It's not determined by the nature of mathematical truth or physical reality. Rather, *C* exists in its present form because of social or historical forces. For cryptography, the relevant forces are to be found in the cryptographic community's disciplinary culture.

Constructionism isn't the only view of why a field is the way it is. A different explanation is scientific realism. In that view, *C* is the way it is because it's true (or at least a good and ever-improving approximation to what's true). There wasn't a lot of choice in the matter. Physicists arrived at the notion of quarks, for example, because they're real. They were discovered. Nobody *invented* quarks, contrary to the provocative title of Andrew Pickering's *Constructing Quarks*.[1] Another community of physicists—maybe aliens on some distant planet—would have come to similar ideas in their own development of particle physics.

Although most scientists subscribe to realist views, many who study the history and sociology of science embrace constructionist views, at least to some extent.

The case for constructionism in theoretical physics can seem a stretch, but the case for constructionism in technology isn't controversial. Technology, after all, is the business of the synthetic, and our artifacts aren't just metaphorically constructed, they're *literally* constructed.

But cryptography is a bit of a problem child, because parts of it seem to rest squarely in engineering and are clearly constructed (nobody would claim that AES, the Advanced Encryption Standard, is anything but construction), while other parts of the field—potentially including all of provable security—might seem better understood as scientific realism.

It's hard to ascertain the extent to which cryptographers hold realist versus constructionist views.

 Copublished by the IEEE Computer and Reliability Societies

Usually such matters are only vaguely implied by technical work. One case where an author's view is made clear is Oded Goldreich's essay "On Post-Modern Cryptography."[2] The piece is decidedly realist. Goldreich's essay emphatically asserts that "*cryptographic research is indeed part of science*." He regards this as the dominant view among cryptographers.

In contrast, I claim that cryptography, even in its most pure and scientific persona, is quite strongly constructed. I use the story of practice-oriented provable security to make this case.

Provable Security

Provable security began around 1982, most especially with the landmark paper of Shafi Goldwasser and Silvio Micali, "Probabilistic Encryption."[3] The authors were only graduate students at the time, yet their paper laid the foundation for the definition-based, reduction-centered approach to cryptography. It has come to be called *provable security*.

The provable-security approach begins with a precise definition of the problem at hand. This entails specifying *what* the adversary is allowed to do and *when* it is deemed successful. A protocol is then given for the problem. Almost invariably, that protocol will rely on some other, lower-level protocol. That lower-level protocol aims to solve a different, hopefully more basic cryptographic aim. That aim too is defined, specifying what *that* kind of adversary can do and when it succeeds. Good protocols are those for which reasonable adversaries are rarely successful. Evidence of security for the higher-level protocol takes the form of a *reduction*: one shows how to transform an adversary attacking the higher-level protocol into an adversary attacking the lower-level one. A belief that the lower-level protocol does its job thus engenders a belief that the higher-level protocol does *its* job.

Provable security has had a profound impact on cryptography. Roughly half of all academic work in the field follows this course. For developing the idea of provable security and for numerous contributions in this domain, Goldwasser and Micali shared the 2012 Turing Award.

Two-Hat Approach

I came to cryptography just after the advent of provable security. It was a wonderful time to be at MIT and studying under Micali. The cryptographic papers then emerging from MIT's cryptographers were beautiful, visionary creations. One has only to reread some of their titles (or phrases therein) to relive a bit of the otherworldliness of that time: "How to Play Mental Poker" (1982), "A Paradoxical Solution to the Signature Problem" (1984), "Proofs that Yield Nothing but Their Validity" (1986), "How to Play ANY Mental Game" (1987). The MIT cryptographers seemed to live in a playful world of unbridled imagination.

As I saw it, at least, cryptography at MIT in the 1980s didn't much countenance pragmatic concerns. The field was a branch of theoretical computer science. The culture of the leading theory conferences, STOC (Symposium on the Theory of Computing) and FOCS (Foundations of Computer Science), was the culture we lived. While a word or two might be uttered in a paper to play up some conceivable application, genuinely practical considerations would have to wait for some less ecstatic day. Philosophy and beauty had primacy over utility in determining what work was good to do.

After MIT, fellow graduate student Bellare and I went off to IBM. I thought it would be a good place to inject into practice some of the wonderful ideas from crypto theory—ideas like secure multiparty computation, where entities could collaborate to compute functions in ways that would protect the privacy of each party's contribution. I thought I'd spend two or three years bringing the science of cryptography to crypto-illiterate practitioners.

The idea was as naive as it was arrogant. In my defense, I believe that many other people had similar delusions, and do to this day. In grandiloquent terms, I imagined the practical denouement of cryptographic work to arise like this: Provable-security would provide the needed paradigms, techniques, and viewpoints. These ideas would then get picked up, refined a bit, and concretely embodied. Finally, they would become objects of material culture that companies like IBM could sell.

The science → technology → society model is sometimes called the "conveyor-belt" or "linear-development" model. The idea is nicely captured in the creepy slogan of the 1933 World's Fair: Science Finds, Industry Applies, Man Conforms. It's a highly inaccurate view of how science, technology, and society interact, yet it provides the implicit framework under which many scientists situate their work.

I soon realized that nobody at IBM was actually trying to inject some worked-out crypto theory into the compliant body of cryptographic practice. For one thing, cryptographic practice was in many ways far ahead of theory, attending to problems we had never even heard of. To deal with the gap, Bellare and I observed our colleagues taking what we came to call the *two-hat approach*. You could wear your theory hat, in which case you'd try to write a nice paper for a theory or cryptography conference; or you could wear your practice hat, in which case you'd use intuition or cleverness to try to design, attack, or refine some real-world thing that, against all odds, had found its way to you.

Bellare and I didn't like this approach. The problem was that when wearing the practice hat, you had to abandon not just your theory-rooted knowledge but, worse, the most basic theory-rooted sensibility: the belief that one should formalize any complex problem before constructing or analyzing a solution. We weren't willing to do this. Yet people around us were in need of solutions to problems that contemporary theory hadn't seen and seemingly couldn't touch.

The Kerberos Problem

The first set of questions I heard concerned something called *Kerberos*—an authentication service that had come from MIT. It was embarrassing to have just come from there yet not to have heard of the thing. But it turned out that no MIT cryptographer had been involved in Kerberos' creation. The irony of this continues to astound me.

At the center of Kerberos is a six-flow protocol among three parties. It's pretty complicated, involving nonces, time stamps, tickets, and a variety of shared keys. Two of the parties, however, don't yet share a key, and it's the protocol's job to see that they get issued one—a short-lived *session key*.

Kerberos is an elaboration and refinement of an earlier three-party protocol for entity authentication and key distribution created by Roger Needham and Michael Schroeder.[4] Besides that protocol and the Kerberos protocol, there turned out to be a veritable zoo of protocols like this—as well as techniques for breaking them or proving them secure (for various notions of secure, formalized and not). The techniques were mostly logical or algebraic in character, effectively abstracting the cryptography as some sort of black-box encryption. The lineage of ideas goes back to Danny Dolev and Andrew C. Yao,[5] although the most popular approach, when I got to IBM, seemed to be BAN logic, named for a fascinating paper by Michael Burrows, Martín Abadi, and Needham.[6] By the early 1990s, many people were doing this kind of work. Few of them were a part of the cryptographic community I knew.

I was shocked to learn of the existence of a sort of "shadow" cryptographic community—an academic community that did cryptography, at least in any catholic understanding of the word, but wasn't represented at the main cryptography conferences. Most people involved didn't even call themselves cryptographers. Had they accepted an oddly narrow and societally constructed view of what a cryptographer does?

Bellare and I wanted to figure out what this Kerberos really *was*. We read all the papers we could find, but they never came close to actually defining the goal. It was frustrating. At some point, I remember reading a paper on Kerberos by Bill Bryant.[7] It was, believe it or not, a four-act, two-person play. One character, Athena, was described as an up-and-coming system developer. The other character, Euripides, was a good-natured adversary. Repeatedly, Athena would refine her protocol and explain her thinking to Euripides. He would promptly attack the thing and send Athena back to work. After a tiresome number of such scenes, Euripides offers no more complaints and Kerberos is born.

I appreciated the author trying to communicate his idea in this nonstandard way. But the kind of iterative, no-assurance-except-so-and-so-didn't-break-it approach is exactly what the cryptographic community had learned to avoid. The play seemed to celebrate an approach I had thought one was supposed to be ashamed of.

By now convinced that IBM and its friends at the Open Software Foundation were spending gazillions of dollars on something utterly foundationless, I arranged to meet Jeff Schiller, a leading figure behind Kerberos, on one of my visits back to MIT. I explained that the problem Kerberos solved was without foundation and pointed out that even if one implemented Kerberos using a "semantically secure" encryption scheme, still it might be trivial to break. The underlying problem, I said, was that encryption was never the right tool for entity authentication. My recollection is that I was perfectly cordial and reasonable, but Schiller must have thought otherwise, as he apparently called up Micali just after we talked to ask what kind of lunatic he had trained.

In the end, Bellare and I found the problem addressed by Kerberos to be too complex to deal with at that time (we would come back to three-party key distribution a few years later). Happily, a paper had just come out that offered up a simpler scenario. Ray Bird and his colleagues at IBM considered two-party entity authentication, again in the symmetric (shared-key) setting.[8] They brought to our attention the idea of instances, sessions, and interleaving attacks. There weren't any definitions in their paper, but Bellare and I saw that one could now provide definitions and proofs.

The model we developed, shown in Figure 1, puts the adversary at center stage. It communicates with a set of *oracles*. These interact with the adversary—not, at least directly, with one another. Oracles reify instances, with oracle $\prod_i^t$ modeling instance t of party i. Oracles are stateful, each initialized to have whatever long-lived keys the corresponding party should hold. Each oracle computes messages according to the protocol under study.

The adversary communicates with its oracles using a repertoire of queries that model its real-world capabilities. Here's an example. A `Send` query, directed to an oracle, lets the adversary see how an instance will respond if sent some message. A `Reveal` query, again directed to an oracle, causes it to relinquish its session

key. A `Corrupt` query, now directed to all oracles corresponding to some party *i*, provides the adversary all state information in all those oracles. Finally, a `Test` query returns either the session key that an oracle holds or, alternatively, a random key drawn from some specified distribution.

To define key distribution, we had to formalize what it means for an oracle to come to have some session key with (just) some communication partner. Thus a notion of *partnering* is needed, a way to know which oracle is paired with which. One also needs a notion of *freshness* for the session key. A session key held by some oracle is *fresh* if the adversary hasn't learned it by trivial means—for example, using a `Reveal` query on the oracle's partner.

A protocol's security is measured by associating a real-valued *advantage* to any adversary. The advantage measures the adversary's ability to predict if the string returned by the `Test` query is a fresh session key or, instead, a randomly chosen key.

After defining mutual authentication and authenticated key exchange, we showed how to achieve these aims from simpler cryptographic tools. A theorem says that if the tools do their jobs well, then the high-level protocol does its job well.

I was excited about this paper. We took a complicated problem that a large community of people wanted solved and brought provable security to it in a completely practical way. But it would take years for the idea to catch on. Theorists thought the problem alien and uninteresting; practitioners felt the same way about cryptographic definitions and proof. Still, by now numerous papers have used the Bellare-Rogaway model to formalize and prove security for entity authentication and key distribution aims. Using this framework, real-world protocols, including Kerberos, SSH, and TLS, have been analyzed or refined.

Message Authentication Codes

As a graduate student, I had heard of blockciphers like DES, the Data Encryption Standard, but I understood that cryptographic theory didn't deal with such objects. They were foundationless, and this was thought to be infectious: anything built from a confusion/diffusion primitive would again lack foundation.

Bellare and I believed that blockcipher-based constructions could be made rigorous. We decided to start with *message authentication codes* (MACs). These are short strings (usually 4 to 16 bytes) that you attach to a message so that a party with whom you share a secret key can verify the message's origin.

From a theory point of view, constructing MACs seemed to be a non-problem: starting with a one-way function, you could make a *pseudorandom function*,

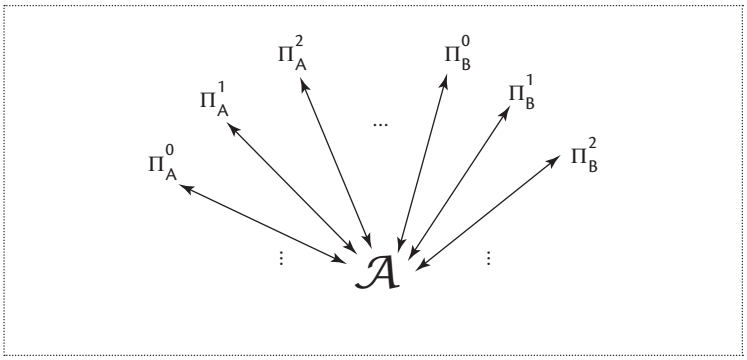

Figure 1. To define entity authentication and key distribution, an adversary is imagined to communicate with a collection of oracles, each of which models a particular instance of a particular entity.

which is always a good MAC. But that kind of answer is completely unresponsive to the practical task at hand. For one thing, it would lead to ludicrously inefficient MACs.

From a practical point of view, the problem would also seem to be solved: you can use the CBC-MAC (the cipher block chaining message authentication code) shown in Figure 2. Given a blockcipher E and a key K for it, you can process the input $M = M_1 \cdots M_m$ as shown to make a MAC T. The method is simple and widely standardized.

But there wasn't any proof that the method worked, and there were other problems, too. The CBC-MAC doesn't work if the messages have varying lengths. Also, the serial nature of the CBC-MAC precludes its use at very high speeds.

Bellare and I wanted to prove that the CBC-MAC of a secure blockcipher is secure (assuming all messages processed are the same length). But we thought, at first, that proving something like this would be impossible. Blockciphers are finite functions (the most popular one at the time, DES, maps 56 + 64 bits to 64 bits). In a finite function, no security parameter is present, so one can't formalize security in the way it had always been done. The notion of "polynomial time" becomes meaningless. There will always be an efficient adversary—indeed a constant-time one—that breaks the blockcipher. Correspondingly, there will always be a constant-time adversary that breaks the CBC-MAC.

Fortunately, we soon figured out that what's written in the last paragraph is rubbish. While the individual claims are correct, they simply don't support a conclusion of the inapplicability of provable security.

We started by formulating, without asymptotics, the security of a blockcipher E. Syntactically, a blockcipher E maps a k-bit key K and an n-bit plaintext X to an n-bit ciphertext $Y = E_K(X)$. Each K induces a permutation on n-bit strings. To measure security, an adversary $\mathcal{A}$

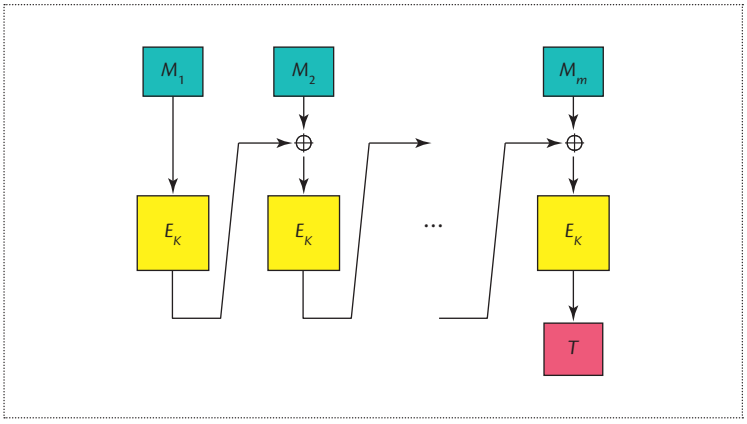

Figure 2. Constructing a message authentication code (MAC). Given a blockcipher E and a key K for it, you can process the input $M = M_1 \cdots M_m$ as shown to create the MAC T.

is provided an oracle that responds to each query X either with $Y = E_K(X)$ or, alternatively, with $\pi(X)$, for a random permutation π on n-bit strings. The adversary, after interacting with one of these two oracles, outputs a bit to indicate its guess as to which. The *advantage* $\mathbf{Adv}^{\mathrm{prp}}_{E}(\mathcal{A})$ garnered by $\mathcal{A}$ in attacking E is the probability that its prediction is correct (by convention, linearly rescaled to land in $[0, 1]$ or $[-1, 1]$). In the viewpoint of practice-oriented provable security, one is now done defining a blockcipher's security: the association of an adversary to a real number measuring its efficacy *is* a definition of security. The notion of blockcipher security just described is security in the sense of a *pseudorandom permutation*, or PRP.

Similarly, we can define the quality of a MAC. Syntactically, a MAC F takes a key K and a message M and returns, deterministically, an n-bit string T. An adversary $\mathcal{B}$ is given access to an oracle. The oracle is initialized with a random key K. Thereafter, it applies the MAC, keyed by K, to each message asked. In the end, the adversary outputs a pair (M,T). The adversary is said to *forge* if M was never asked of the oracle yet T is the right MAC for it (with respect to the hidden key K). The adversary's advantage $\mathbf{Adv}^{\mathrm{mac}}_{F}(\mathcal{B})$ is the probability that it forges.

Bellare, Joe Kilian, and I proved the security of the CBC-MAC.[9] The proof seemed hard in 1994, but subsequent ideas have made it easy. Let F be the CBC-MAC of an n-bit blockcipher. Suppose you have an adversary $\mathcal{B}$ that computes F on q strings, each of them mn-bits, and then, sometimes, it forges a message of this same length. Then you can use $\mathcal{B}$ to break the blockcipher E. Specifically, you can turn $\mathcal{B}$ into an adversary $\mathcal{A}$ where $\mathbf{Adv}^{\mathrm{prp}}_{E}(\mathcal{A}) \geq \mathbf{Adv}^{\mathrm{mac}}_{F}(\mathcal{B}) - 2m^2q^2/2^n$. The efficiency of $\mathcal{A}$ is close to the efficiency of $\mathcal{B}$.

The great thing about having an explicit formula like this is that it lets you figure out what your results mean for practice. If, for example, $\mathcal{B}$ attacks the CBC-MAC of AES (a 128-bit blockcipher) using a billion messages of 1 Kbyte each, then it will have a chance of forgery that's at most 2^{-55} more than the insecurity of AES. That's a useful thing to be able to say.

Still, the real significance of our work wasn't that it analyzed the CBC-MAC but that it helped bring symmetric cryptography into the provable-security fold. Our paper, and others we wrote around the same time, demonstrated that blockciphers, and other finite functions, make a perfectly good starting point for doing reductions. What's more, reductions could serve as a tool for understanding or improving cryptographic practice.

Interlude

The original approach to provable-security cryptography aimed to study abstract security relationships among asymptotically defined objects. One would make claims like "a one-way function implies a pseudorandom generator." The choice of what to use as a starting point and what to reach as an endpoint was based mostly on aesthetic considerations. Later, many problems that fell under the provable-security scalpel were quite speculative or imaginative. This is fun and good, but it's also good to develop tools and techniques likely to be deployed.

Practice-oriented provable security would therefore take a different tack. We would be guided by practical considerations about what problems to address. This can't be done with only one's imagination as a guide. We would study the concrete security relationships among what were often finitary objects. The efficiency of reductions would be emphasized. We would try to build useful objects out of whatever cryptographic practice brought to the table.

The traditional provable-security ethos tended to nudge theorists away from useful questions—even, on occasion, getting things backward from a practical point of view. Consider the problem of creating a secure PRP out of a one-way function (for example, multiplication of large prime numbers). It's a fascinating thing to try to do. But if you actually need a one-way function for cryptographic practice, as UNIX systems did with the "`crypt`" library call, the natural approach is to make one out of a blockcipher—the exact *opposite* of the problem theorists had studied.

Authenticated Encryption

The first formalization for encryption, due to Goldwasser and Micali (GM), was in the public-key setting.[3] More from a lack of interest than for technical reasons, it would take 15 years for GM's paper to get adapted to the shared-key setting. This happened in a 1997 paper by Bellare, Anand Desai, Eron Jokipii, and me (BDJR).[10]

In BDJR, encryption is performed by a probabilistic algorithm $\mathcal{E}$ that maps a key K and a plaintext M to a ciphertext C. Decryption deterministically recovers M from K and C. To formalize security, an adversary $\mathcal{A}$ is given access to one of two types of oracles. Both begin by choosing a random key K. A *real* encryption oracle then responds to each query M by encrypting it, returning a ciphertext $C \xleftarrow{\$} \mathcal{E}(K, M)$. The dollar sign is a reminder that the encryption process depends on random coins. In contrast, a *fake* encryption oracle responds to each query M by encrypting the same number of zero bits: $C \xleftarrow{\$} \mathcal{E}(K, 0^{|M|})$. Adversary $\mathcal{A}$'s *advantage* is a real number that measures how well it can distinguish the two types of oracles. Formally, it's the probability that $\mathcal{A}$ answers 1 given the real encryption oracle minus the probability that $\mathcal{A}$ answers 1 given the fake encryption oracle.

There are a lot of ideas implicit in a definition like the one just given. Let me identify two. First, we understood that, to be secure, encryption had to be probabilistic. GM had explained that a good encryption algorithm couldn't be deterministic, because it would, in that case, reveal repetitions in plaintexts. Second, we understood that encryption deals only with privacy. Other tools should handle other aims, like authenticity. If someone told me in the mid-1990s that they had designed a symmetric encryption scheme that also guaranteed authenticity, I might have advised them not to call it encryption, because achieving authenticity wasn't what "encryption" ought to do.

Writing BDJR, we didn't make a conscious choice on either matter: we didn't see any alternative. Ludwik Fleck, an early contributor to the sociology of science, wrote that "Once a structurally complete and closed system of opinions consisting of many details and relations has been formed, it offers enduring resistance to anything that contradicts it."[11] One form of resistance is invisibility. Fleck explains that "What does not fit into the system remains unseen."

If the goal of a notion of symmetric encryption was to create an abstraction boundary useful for applied cryptography, then the BDJR definition falls short. Here are three reasons why.

First, users of encryption often assume it provides a lot more than privacy. They assume it provides authenticity, invalid ciphertexts being *rejected*. The designers of Kerberos, for example, implicitly assumed this. Experience makes clear that users will misuse any tool that provides less than they expect. Second, users of encryption often need to authenticate stuff that isn't being encrypted. A typical example is a message header in a networking protocol. The header should be authenticated, so that routing information can be verified by the receiver, but it can't be encrypted, as intermediate routers need to use the header and don't have the key. Such considerations suggest that, when encrypting a payload, one should authenticate something else on the side, and bind together the two.

Third, the use of random bits is problematic. Implementers and protocol designers routinely fail to provide adequately random bits. Wrong advice about initialization vectors (IVs) is common—for example, it has often been claimed, wrongly, that a counter will work for the IV of CBC mode. Generating an adequately good approximation of random bits can be costly or impossible—and it usually requires cryptography. People routinely get it wrong. In practice, trading randomness for a *nonce*—something, like a counter, that's used at most once for a given key—is a big win.

We can reformulate encryption in a way that reflects all three concerns, a line of work that my colleagues and I carried out between 2000 and 2006. Let me sketch what's now a standard notion for symmetric encryption—*authenticated encryption* (AE) or, to be more explicit, *authenticated encryption with associated data* (AEAD).

We begin with syntax. Encryption is now a deterministic and stateless mechanism that takes a key K, a nonce N, associated data A, and a plaintext M. It returns a ciphertext $C = \mathcal{E}(K,N,A,M)$. Decryption, again deterministic and stateless, transforms a key K, nonce N, associated data A, and a ciphertext C to either a string-valued plaintext $M = \mathcal{D}(K,N,A,C)$ or else $\perp$ (bottom), an indication of invalidity.

A security definition captures the intent that encryption should protect the privacy of M and the authenticity of N, A, and C. To formalize this, the adversary is presented with a pair of oracles. There are two possibilities. The first is to give the adversary "real" encryption and decryption oracles. The oracles are initialized with a random key K. Then, when the adversary asks an encryption query of (N,A,M), it gets the ciphertext $C \leftarrow \mathcal{E}(K,N,A,M)$. For simplicity, assume $|C| = |M| + \tau$ for some constant τ, the *ciphertext expansion*. When the adversary asks the decryption oracle (N,A,C), it gets plaintext $M \leftarrow \mathcal{D}(K,N,A,C)$. This value might be $\perp$, to indicate invalidity.

Alternatively, the adversary might be given "fake" encryption and decryption oracles. The former, presented with (N,A,M), returns $|M| + \tau$ random bits (where $|M|$ denotes the length of M). The later, presented (N,A,C), returns $\perp$.

To keep the adversary from trivially winning and to ensure that nonces have their intended semantics, we add in the restriction that the adversary may not repeat a nonce N in any encryption query, nor may it ask a decryption query of (N,A,C) after having asked an encryption query (N,A,M) that returned C.

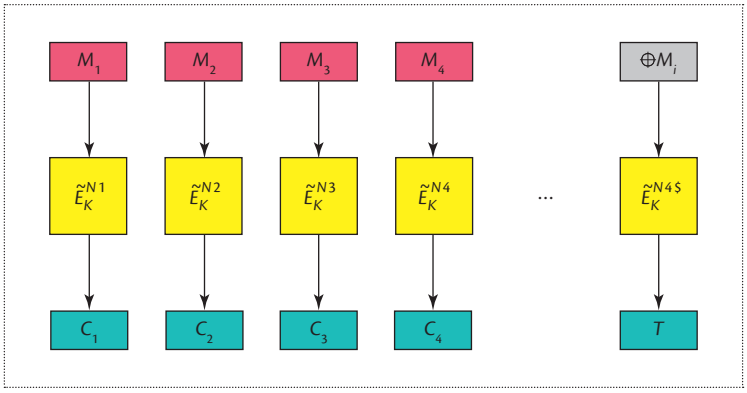

Figure 3. OCB mode authenticated encryption using a tweakable blockcipher $\tilde{E}$. Using the key K and nonce N, plaintext $M = M_1 \cdots M_m$ is encrypted to a ciphertext $C = C_1 \cdots C_m$ T that ensures both privacy and authenticity.

One of the advantages of regarding AEAD as a single conceptual primitive is that one can now try to optimize efficiency characteristics of a scheme for it while insisting on a proof with good bounds. Efficiency has been a key consideration in AEAD schemes like GCM[12] and OCB.[13] The latter, pictured in Figure 3 without its AD, uses a *tweakable blockcipher* (TBC). TBCs, formalized by Moses Liskov, Ronald Rivest and David Wagner,[14] are like conventional blockciphers except for the inclusion of the tweak (shown as a superscript). Each tweak T names a permutation $\tilde{E}_K^T$ that's essentially independent of the others. For OCB, the TBC is realized from a conventional blockcipher, typically AES, in such a way that the overhead is tiny. In the end, OCB using AES executes at nearly the same speed as counter mode (CTR) with AES, a basic blockcipher mode that provides no authenticity.

AEAD has taken off, supplanting classical modes of operation like CBC and replacing the conventional understanding of what a symmetric encryption scheme *should* deliver. Numerous AEAD schemes have been devised, and several have been standardized. TLS version 1.3 will allow only AEAD, which should put an end to the long sequence of cryptographic attacks on the record protocol of SSL/TLS. A competition for AEAD schemes, CAESAR, drew 57 submissions from around the world. At the time of this writing, there remain 15 third-round candidates. The competition should conclude in 2018.

AEAD continues to evolve. Recent work on *online* and *misuse-resistant* AE highlight the unsettledness that pervades any active field. The first aims to achieve best-possible security for a single, constant-memory pass over the inputs. The second aims to achieve strong security even if nonces get reused. There remains contention around how to define these ideas, and how expensive achieving them needs be.

The success of AEAD doesn't just stem from improving speed: it stems from finding an abstraction boundary that's easier to correctly use. To a significant extent, cryptography has been a search for desirable abstraction boundaries. And it's not mathematical or scientific truth that helps you find good abstraction boundaries. They are drawn by those who work in a field to advance it in a chosen direction.

Further Evidence

I've been using anecdotes from my own research to illustrate how one branch of cryptography took the form that it did more as a matter of human agency than scientific necessity. There are many other ways to support this claim. Let me mention two: regional differences and the inessential partitioning of the field.

First, geography correlates with the style of cryptography a person will do. While the provable security tradition has had an enormous influence on cryptography in the US, it dominates far less in Europe. Symmetric cryptanalysis never really caught on among US academics but has flourished in Europe, Israel, and Japan. The character of cryptography in the US flows mostly from the theory community, but in Europe, cryptography has been more strongly shaped by combinatorics, coding theory, and electrical engineering. Institutional traditions and the interests of funding agencies have been formative in shaping what is done where. This argues for constructionism.

Second, consider the peculiar fact that cryptography-based privacy research is mostly done outside the mainstream cryptographic community. Artifacts like Tor aren't studied by people in the crypto community so much as those in the privacy-enhancing technologies (PETs) community. Different people, conferences, and sensibilities are involved. Yet there's no good reason why this should be so, and it's easy to imagine a different timeline in which the crypto and PETs communities are one, privacy playing a much larger role in defining crypto's course.

Platonism versus Formalism

In the philosophy of mathematics, the distinction between Platonism and formalism mirrors the divide between realism and constructionism.

The Platonist views mathematical truth as something "out there." The mathematicians' job is to find this truth and communicate it to colleagues. In contrast, the formalist sees math as the invention of humans. We fashion axioms and definitions and explore the world that we thereby create. What is "out there" is that which we put.

Philip Davis and Reuben Hersh say that "the typical working mathematician is a Platonist on weekdays and a formalist on Sundays. That is, when he is doing

mathematics he is convinced that he is dealing with an objective reality whose properties he is attempting to determine. But then, when challenged to give a philosophical account of this reality, he finds it easiest to pretend that he does not believe in it after all." [15]

For those doing rigorous work in cryptography, the claim above rings true (apart from the suggestion that we don't work on weekends!). In proving a theorem, and even in devising a definition or scheme, a cryptographer feels more a seeker of truth than the inventor of artifice. But this feeling is probably wrong, and perhaps it's wrong in an ironic way, for in cryptography, artifice is actually a goal.

Recognizing the extent to which a field is constructed can be liberating. When you internalize that a belief like "secure encryption must be probabilistic" is pure construction, it opens the door to many interesting research directions. When disciplinary assumptions lose some of their authority, the resulting freedom creates an atmosphere where invention can thrive.

Recognizing cryptographic work as a social construction allowed Bellare and me to overcome the two-hat predicament. It was predicated on a wrong belief: that you couldn't meld the needs of practice with the essential characteristics of theory. But you could. We could ditch the two hats in favor of one.

In the end, the cryptographic community isn't following some predestined journey of discovery; we are engaged in an ongoing dialectic. That dialectic continually refines, reformulates, and answers anew the most basic question of any field: what are our basic aims, and how are we going to get there? ■

Acknowledgments

Ideas expressed here were heavily influenced by my years working with Mihir Bellare, who also provided key comments. Thanks to Dan Boneh, Kenny Paterson, and Nigel P. Smart for putting together this special issue. Thanks to the anonymous referees for their useful comments, including the mention of Platonism vs. formalism and the reference to Davis and Hersh. A longer version of this article was written to accompany an invited talk at Eurocrypt 2009. I have received support from NSF grants CNS 0904380, 1228828, and 1314885. All opinions expressed here are strictly my own, not necessarily those of the NSF.

References

1. A. Pickering, *Constructing Quarks: A Sociological History of Particle Physics*, Univ. Chicago Press, 1984.
2. O. Goldreich, "On Post-Modern Cryptography," Nov. 2006 (revised Feb. 2012); www.wisdom.weizmann.ac.il/~oded/on-pmc1.html.
3. S. Goldwasser and S. Micali, "Probabilistic Encryption," *J. Computer and System Sciences*, vol. 28, no. 2, 1984, pp. 270–299 (earlier version in STOC 1982).
4. R. Needham and M. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," *Comm. ACM*, vol. 21, no. 12, 1978, pp. 993–999.
5. D. Dolev and A.C. Yao, "On the Security of Public Key Protocols," *IEEE Trans. Information Theory*, vol. 29, no. 2, 1983, pp. 198–207.
6. M. Burrows, M. Abadi, and R. Needham, "A Logic of Authentication," *Proc. Royal Society of London, Series A*, Mathematical and Physical Sciences, vol. 426, no. 1871, 1989, pp. 233–271.
7. B. Bryant, "Designing an Authentication System: A Dialogue in Four Scenes," MIT, 1988; web.mit.edu/kerberos/dialogue.html.
8. R. Bird et al., "Systematic Design of Two-Party Authentication Protocols," *Advances in Cryptology* (CRYPTO 91), 1991, pp. 44–61.
9. M. Bellare, J. Kilian, and P. Rogaway, "The Security of the Cipher Block Chaining Message Authentication Code," *J. Computer and System Sciences*, vol. 61, no. 3, pp. 362–399.
10. M. Bellare et al., "A Concrete Security Treatment of Symmetric Encryption," *Proc. 38th Ann. Symp. Foundations of Computer Science* (FOCS 97), 1997, pp. 394–403.
11. L. Fleck, *Genesis and Development of a Scientific Fact*, Univ. Chicago Press, eds., T. Trenn and R. Merton, 1981 (translated by F. Bradley and T. Trenn, originally published in German, 1935).
12. M. Dworkin, "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC," NIST Special Publication 800-39D, Nov. 2007.
13. T. Krovetz and P. Rogaway, "The OCB Authenticated-Encryption Algorithm," RFC 7253, Internet Research Task Force, May 2014.
14. M. Liskov, R. Rivest, and D. Wagner, "Tweakable Block Ciphers," *J. Cryptology*, vol. 24, no. 3, 2011, pp. 588–613.
15. P.J. Davis and R. Hersh, *The Mathematical Experience*, Mariner Books, 1998.

Phillip Rogaway is a professor in the Department of Computer Science at the University of California, Davis. His research focuses on obtaining provably good solutions to protocol problems of genuine utility. Rogaway received a PhD from MIT. Contact him at rogaway@cs.ucdavis.edu.