Constructive Cryptography – A New Paradigm for Security Definitions and Proofs*

Ueli Maurer

Department of Computer Science
ETH Zurich
CH-8092 Zurich, Switzerland
maurer@inf.ethz.ch

Abstract. Constructive cryptography, an application of abstract cryptography proposed by Maurer and Renner, is a new paradigm for defining the security of cryptographic schemes such as symmetric encryption, message authentication codes, public-key encryption, key-agreement protocols, and digital signature schemes, and for proving the security of protocols making use of such schemes. Such a cryptographic scheme can be seen (and defined) as constructing a certain resource (e.g. a channel or key) with certain security properties from another (weaker) such resource. For example, a secure encryption scheme constructs a secure channel from an authenticated channel and a secret key.

The term "construct", which is defined by the use of a simulator, is composable in the sense that a protocol obtained by the composition of several secure constructive steps is itself secure. This is in contrast to both the traditional, game-based security definitions for cryptographic schemes and the attack-based security definitions used in formal-methods based security research, which are generally not composable.

Constructive cryptography allows to take a new look at cryptography and the design of cryptographic protocols. One can give explicit meaning to various types of game-based security notions of confidentiality, integrity, and malleability, one can design key agreement, secure communication, certification, and other protocols in a modular and composable manner, and one can separate the understanding of what cryptography achieves from the technical security definitions and proofs, which is useful for didactic purposes and protocol design.

1 Introduction and Motivation

1.1 Modularity in Constructive Disciplines

A central paradigm in any constructive discipline is the decomposition of a complex system into simpler component systems or modules, which each may consist of yet simpler modules, and so on. This paradigm is useful only if the composition of modules is well-defined and preserves the relevant properties of the

* This paper is an extended abstract accompanying the author's invited talk at TOSCA 2011. The author is supported by the Swiss National Science Foundation.

modules. For example, in software design, the composition operation must preserve correctness of the modules, i.e., correctness should be defined in a way that a system consisting of correct modules is itself correct.

The goal of constructive cryptography is to see cryptography as a constructive discipline, in a well-defined sense. The design of a cryptographic protocol involves several mechanisms (e.g. encryption, message authentication, etc.), each of which is proven secure in isolation (as a module). The security of the composed protocol then follows from a general composition theorem.

However, this approach requires the security of cryptographic schemes to be defined in a suitable manner. Indeed, for the traditional, game-based cryptographic security definitions, as explained below, the composition property is unclear. In contrast, security definitions stated in the proposed constructive manner are intrinsically composable and capture what one really wants to say about a concrete cryptographic system.

1.2 Traditional Security Definitions in Cryptography

In a traditional definition of security of a cryptographic scheme, one usually defines a game that characterizes the capabilities of a (hypothetical) adversary. A cryptographic scheme is defined to be secure if no computationally feasible strategy allows the adversary to win the game with non-negligible probability (or advantage), for reasonable notions of feasible and negligible. The notion of "feasible" is hard-wired into the definition and is defined as some form of polynomial time. Similarly, negligible is defined in a specific manner such that, roughly speaking, feasible times negligible is still negligible. Such definitions are therefore necessarily asymptotic.

For example, the security of a message authentication code (MAC) is defined as follows. Roughly speaking, without access to the secret key, no adversary restricted to feasible computation can win the following game with non-negligible probability. The adversary has access to a MAC-oracle (with the secret key embedded) and can ask for the generation and/or verification of MACs for arbitrary messages. The game is won if the adversary can generate a fresh message (not asked to the oracle before) as well as a correct MAC for it.

While this definition sounds reasonable and strong and naturally captures intuition, one may still ask why it should be the right definition. Indeed, for many cryptographic primitives, several different security definitions have been proposed. For example, a strengthened version of the security of a MAC requires that it even be infeasible to generate another valid MAC for the same message (assuming the MAC scheme is probabilistic). It is generally not clear when such a stronger definition is required.

For encryption, the definitions include security against chosen-plaintext attacks (CPA) or against chosen-ciphertext attacks (CCA), and for different types of distinguishing games. In addition, various integrity definitions for encryption have been defined. Several authors have investigated which combinations of such notions are strong enough in a given context.

1.3 Constructive Cryptography

The question of which definition is adequate should be answered relative to a specification of what is supposed to be achieved by the application of the scheme, under specified assumptions. Hence the constructive approach proposed here.

Constructive cryptography is a new paradigm in which the security definition of cryptographic schemes is radically different. For example, a MAC is defined to be secure if it *constructs* an authenticated communication channel from an insecure communication channel and a secret key, for a well-defined, simulation-based notion of "construct" and for well-defined definitions of an insecure and an authenticated channel[1]. Similarly, a symmetric encryption scheme is defined to be secure if it constructs a secure communication channel from an authenticated communication channel and a secret key.

The general composition theorem (Theorem 1) of this theory implies that the combination of a secure MAC and a secure encryption scheme constructs a secure channel from an insecure channel and two secret keys. By another composition step, the two keys can be constructed from a single secret key using a pseudo-random generator.

The security of public-key cryptosystems, key agreement protocols, and digital signature schemes can be seen similarly in the constructive cryptography paradigm, but this will not be discussed in this paper (but see [21]). The emphasis of this paper is on the general theory, less on the discussion of specific cryptographic schemes, which are discussed in [22,20] and in future papers.

1.4 Outline of the Paper

In Section 2 we discuss the well-known one-time pad from a new perspective, leading to the constructive cryptography viewpoint explained in Section 3. In Section 4 we briefly explain the idea of [18] to introduce abstraction layers in cryptography. In particular, the theory of abstract systems is introduced. In Section 5, the theory of constructive cryptography is explained. Related work is discussed in Section 6, and several research directions that can be addressed in the framework of constructive cryptography are mentioned in the concluding Section 7.

2 Motivating Example: The One-Time Pad

In this section we discuss the one-time pad (OTP), the best-known provably secure cryptosystem, as a motivating example. We recall the traditional (information-theoretic) security proof for OTP-encryption and explain two intrinsic problems with this proof. First, despite the proof, the OTP can be argued

[1] While using ideas of the frameworks of Canetti [4] (universal composability, called UC) and of Backes, Pfitzmann and Waidner [24,2] (reactive simulatability), which formalized the so-called "ideal-world real-world" paradigm, constructive cryptography is significantly different, as discussed in Section 6.

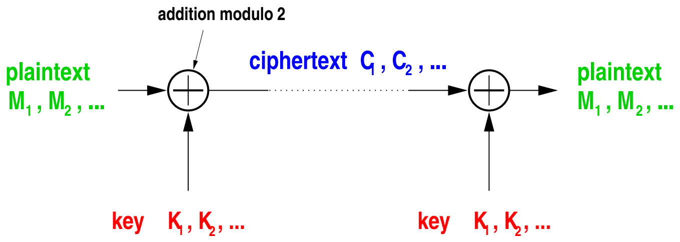

Fig. 1. The one-time pad (OTP) encryption scheme. The plaintext, the key and the ciphertext are bit strings of equal length, say n bits. The key is a uniformly random bit-string that is used only once. Encryption [decryption] consists of XORing (i.e., adding bit-wise modulo 2) the key to the plaintext [ciphertext]. The adversary is assumed to have access to the communication channel (i.e., to the ciphertext) but has no (a priori) information about the key.

to be completely insecure, for two different reasons. Second, the security proof cannot be carried over to a more practical setting where the (long) secret key is replaced by a pseudo-random key generated from a short secret key. Intuitively, an information-theoretic security proof should carry over and become a proof of computational security. The constructive approach to defining security solves both problems of standard security definitions.

2.1 The Traditional Security Proof for the One-Time Pad

The OTP encryption scheme (see Figure 1) can be proved to be information-theoretically secure, i.e., unbreakable even for a computationally unbounded adversary. How can one formulate and prove this claim?

The usual way to argue about the security of OTP-encryption is to show that the plaintext (also called the message) and the ciphertext are statistically independent. This means that the ciphertext gives no information about the plaintext, independent of the available computing power. In other words, if one has to guess the plaintext when given the ciphertext, one can just as well ignore the ciphertext.

Proposition 1. *In one-time pad encryption, the plaintext and the ciphertext are statistically independent (for any plaintext distribution).*

Proof. Let M, C, and K denote random variables corresponding to the plaintext, the ciphertext, and the key, respectively. Because the key K is uniformly distributed, so is the ciphertext C for every choice $M = m$ of the plaintext message. In other words, $\mathsf{P}_{C|M=m}$ is the uniform distribution for all m, and hence so is P_C. Thus

$$\mathsf{P}_{CM}(c,m) \;=\; \mathsf{P}_{C|M}(c,m) \cdot \mathsf{P}_M(m) \;=\; \mathsf{P}_C(c) \cdot \mathsf{P}_M(m),$$

for all m and c, which is the definition of M and C being statistically independent. □

In the following two subsections we discuss two reasons why this is not really the statement about the OTP one should want to prove.

2.2 Two "Security Problems" of the One-Time Pad

Despite the above security proof, the OTP appears to be very "insecure", at least for two independent reasons.

First, an adversary with access to the communication channel can modify the message in a controlled manner, even though she gets no information about the transmitted message. The adversary can XOR an arbitrary offset δ to the transmitted message, simply by XORing δ to the ciphertext. If OTP-encryption were used in a banking application, an attacker could for instance change the bank account number of a money transfer.

Second, suppose that the OTP is used to encrypt one of the two messages "Yes" and "No" encoded, say, in 8-bit ASCII. Since the two messages have different lengths, the ciphertext leaks complete information about the message, in sharp contradiction to the above claim that the ciphertext is independent of the plaintext message.

What is a reasonable answer to the above two criticisms, apparently suggesting that the security proof for the OTP is useless in practice? The right answer is that we have to declare explicitly what we assume to be available and what we claim to achieve. First, we need to assume that the channel over which the ciphertext is transmitted is an *authenticated* (but not necessarily confidential) channel, preventing an adversary from modifying the ciphertext. Second, we need to state explicitly that the resulting secure channel actually *leaks the message length* (but not more). Then the constructive security proof can be interpreted as stating that OTP-encryption securely constructs a secure length-leaking channel from an authenticated channel and a shared secret key.

2.3 Computational Security: Additive Stream Ciphers

To illustrate the second short-coming of the above security statement (Proposition 1), we consider a variation of OTP-encryption used in practice, where the truly random key is replaced by a keystream sequence that is generated deterministically from a short secret key by a so-called pseudo-random generator. This is called an *additive stream cipher* (see Figure 2).

How should one define the security of such a stream cipher? Obviously, because the entire keystream is generated from a short secret key, the system cannot be information-theoretically secure. One could (theoretically) break the system by trying all possible keys until the correct one is identified as the only key that results in meaningful plaintext after decryption[2]. Hence we must find a way to formulate what it means for this system to be *computationally secure*.

[2] This requires a sufficient amount of ciphertext and assumes that the plaintext is redundant, i.e., that one can distinguish between meaningful and meaningless messages.

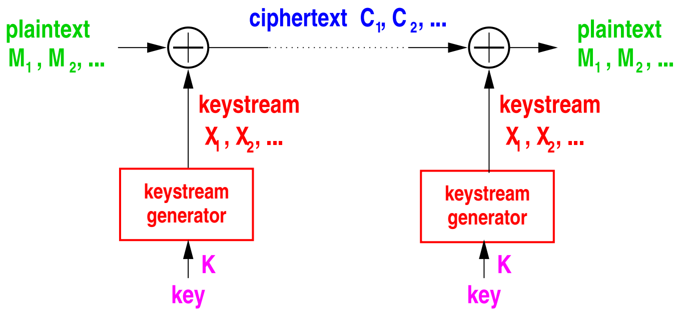

Fig. 2. Additive stream cipher. It is obtained from OTP-encryption by replacing the random key by the output of a pseudo-random generator (also called keystream generator in this context) applied to the secret key K. This system can be used for many consecutive messages if sender and receiver use consecutive portions of the keystream.

Our goal is that the computational security of the additive stream cipher follows from the (information-theoretic) security proof of the OTP and the assumption that the keystream generator is a cryptographically secure pseudo-random generator. What we need are two things: a computational version of "securely construct" and a composition theorem that allows to combine such statements. Here the two statements are that a pseudo-random generator (computationally) securely constructs a long secret key from a short secret key, and the previously explained constructive statement for the OTP.

3 Explaining the Security of the One-Time Pad in Constructive Cryptography

3.1 The Basic Approach

In the constructive cryptography approach, a cryptographic mechanism (e.g. the one-time pad) is interpreted as a method for constructing a certain *ideal resource* from a certain *real resource*. This is in the spirit of [4,24,2]. One argues that an adversary could, in an ideal world where the ideal resource is available, achieve anything that he could achieve in the real world where the real resource is given. This argument involves, as a thought experiment, a simulator system which transforms the ideal resource into the real-world system consisting of the real resource and the protocol engines (called converters).

This paradigm will be explained at an abstract level in Section 5. In the following we discuss the one-time pad example. What will be important to note is that we consider two types of systems: resources and converters.

3.2 The One-Time Pad Example

We need to define the real resource and the ideal resource for the one-time pad example. We refer to Figure 3.

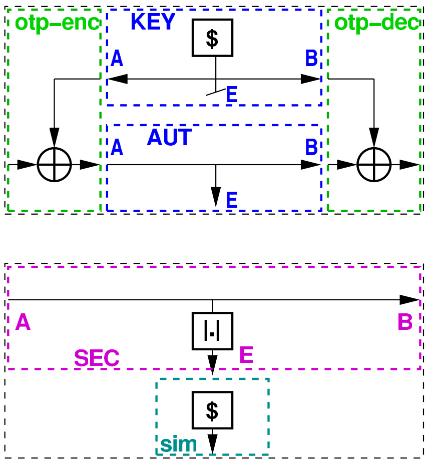

Fig. 3. The one-time pad (OTP) as an example of the constructive cryptography approach. The OTP constructs a secure channel SEC (the ideal resource) from an authenticated channel AUT and a secret key KEY (the real resource). There exists a simulator system sim which, when attached to interface E of the ideal resource SEC, makes it equivalent to the real resource with the protocol engines otp-enc and otp-dec attached. Equation (1) is an alternative (algebraic) way of describing the systems and stating their equivalence.

The ideal resource is a secure channel (see Figure 3, bottom), denoted as SEC (or also as $A \bullet\!\!\longrightarrow\!\!\bullet B$ in the so-called $\bullet$-calculus; see [21,22]). A secure channel SEC (for sending a single message from A to B) is a system with three interfaces, for a sender A, a receiver B, and an adversary E. SEC takes an input (from some message space) at interface A, outputs the message at interface B, and outputs the length of the message (indicated by $|\cdot|$) at interface E.

Systems like SEC, AUT, otp-enc, and those discussed below can be made precise in any suitable language, but this is beyond the scope of this paper. As mathematical objects, they are random systems (see [14] or [17]). One also has to specify aspects like whether the adversary E has the capability of deleting a message, and whether the channel allows to send only one or several consecutive messages. For simplicity, the reader can think of our channels as allowing the transmission of a single message, but the claims also hold for channels allowing to send several messages.

The real resource (see Figure 3, top) consists of an authenticated channel, denoted as AUT (or also as $A \bullet\!\!\longrightarrow B$, see [21,22]), and, in parallel, a shared secret key, denoted as KEY (or also as $A \bullet\!\!=\!\!=\!\!\bullet B$). This system can be written as AUT||KEY[3]. In contrast to a secure channel SEC, an authenticated channel AUT

[3] The parallel composition for resources, denoted || is neither commutative nor (necessarily) associative. It is postulated in an abstract sense in Section 4, and it must

is one in which E can also learn the message sent by A but, like for a secure channel, E cannot modify the message output to B.

The protocol specifies protocol engines (later called *converter systems*) for A and B, namely otp-enc and otp-dec, respectively, which they attach to the system AUT||KEY, resulting in a system we can write as

$$\mathsf{otp\text{-}dec}^{\mathrm{B}}\ \mathsf{otp\text{-}enc}^{\mathrm{A}}\ (\mathsf{KEY}||\mathsf{AUT}).$$

Here the superscripts (e.g. B in $\mathsf{otp\text{-}dec}^{\mathrm{B}}$) specifies that the converter otp-dec is attached to interface B of the resource system AUT||KEY (see Section 4).

How can we argue that OTP encryption constructs a (length-leaking) secure channel? After all, in the real world (i.e., when OTP encryption is used) the adversary receives something, namely the ciphertext, while she receives nothing in the ideal world. How can we resolve this apparent problem?

3.3 The Simulator

Since the ciphertext is statistically independent of the message, we can argue that the adversary is not better off when the ideal resource SEC is available compared to when OTP encryption is used. If the ideal resource were used, he could *simulate* the ciphertext (he would receive in the real world, see Figure 3, top) by himself. More precisely, if a simulator sim generating a random ciphertext (of appropriate length[4]) is attached to the adversary interface of the ideal system, then the resulting system (see Figure 3, bottom) is equivalent (i.e., identical) to the system in the real world (OTP encryption). Written as an equation of systems, we have

$$\mathsf{otp\text{-}dec}^{\mathrm{B}}\ \mathsf{otp\text{-}enc}^{\mathrm{A}}\ (\mathsf{KEY}||\mathsf{AUT}) \quad \equiv \quad \mathsf{sim}^{\mathrm{E}}\ \mathsf{SEC}. \tag{1}$$

(see the top and bottom systems in Figure 3). Here equivalent means that the entire input-output behavior is exactly the same. In other words, both systems (at the top and the bottom of Figure 3) behave identically. Each system takes an input at interface A, outputs this message at interface B, and outputs a random string of the same length at interface E. Equivalent behavior means that the two systems behave identically if plugged into any possible environment or application. This means that from the adversary's viewpoint, the ideal world is just as good as the real world because anything that could happen in the real world could identically happen in the ideal world. We remark that this simulation paradigm has a deeper justification in the abstract cryptography framework [18], where simulators are not part of a security definition but only part of the *proof* of a security statement.

be defined concretely for a specific instantiation of the system concept. Here is is understood naturally as the asynchronous parallel composition.

[4] The length must be given as an input to the simulator, hence it must be "leaked" by the secure channel.

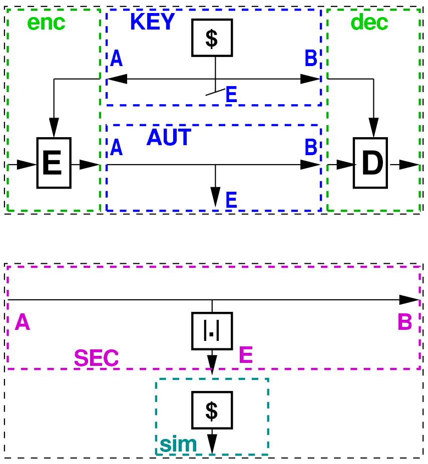

Fig. 4. Explaining symmetric encryption in constructive cryptography. The only change compared to the one-time pad example is that the two protocol systems (later called converters) otp-enc and otp-dec are replaced by enc and dec which implement the encryption algorithm E and decryption algorithm D, respectively. Equation (1) is replaced by (2).

3.4 Computational Security

It is well-known that the one-time pad, and any other information-theoretically secure encryption scheme, is impractical because, roughly speaking, the key must be as long as the total message length to be sent[5]. In practice one therefore strives for security (only) against computationally bounded adversaries, which is called *computational security*. Computational hardness is usually phrased in an asymptotic sense, and the bound on the adversary's computing power is usually assumed to be polynomial in the security parameter[6].

Figure 4 shows a setting where the OTP encryption scheme is replaced by a computationally secure encryption scheme, for example a secure stream cipher as shown in Figure 2. One can use the same simulator for the ideal system (generating a random ciphertext of appropriate length). However, the real resource with encryption/decryption and the ideal resource with simulator are not identical anymore. All we want to argue is that they are *essentially similar* (or, alternatively stated, very close) for an adversary with bounded computing power. More generally, we want to argue that they behave essentially identically in any realistic application context, where realistic means that only feasible computations occur. Written as an equation of systems, the requirement can be stated as

[5] This is true if no additional source of information (e.g. a satellite with noisy channels to the parties) is available [13].

[6] However, as explained in [18], this is not essential; what is relevant is a composition property of the feasibility notion.

$$\mathsf{dec}^{\mathrm{B}}\ \mathsf{enc}^{\mathrm{A}}\ (\mathsf{KEY}||\mathsf{AUT}) \quad \approx \quad \mathsf{sim}^{\mathrm{E}}\ \mathsf{SEC}, \tag{2}$$

where the notion of system being approximately equivalent needs to be defined.

3.5 Distinguishers and Computational Indistinguishability

The notion of two systems S and T "being similar" or "behaving essentially identically" in any context is captured by the concept of a *distinguisher*. A distinguisher D for two systems of the same type is a system (or algorithm) that can access ("play with") a system of this type and outputs a bit. The distinguisher's advantage is the difference of probability of outputting 1 in the two cases, where D is connected to S and where D is connected to T, respectively.

Two systems S and T are ε-similar (or ε-close) if no distinguisher (in a certain class of distinguishers) has a distinguishing advantage of more than ε. If we consider the class of all *feasible* (e.g. polynomial time) distinguishers and ε is very small (negligible), then this means that in any computationally efficient environment or application (which can be considered as the distinguisher) the two systems behave essentially the same. The probability that a difference can be observed is at most ε.

This leads to a constructive security definition of an encryption scheme; it is secure if it *constructs*, in the sense of computational indistinguishability (eq. (2)), a secure channel from an authenticated channel and a secret key.

3.6 One-Time Pad Encryption over an Insecure Channel

While, as explained above, encryption is naturally used over an authenticated channel, one may ask what happens when it is used over an insecure channel. Indeed, some protocols (e.g. SSL) can be seen as first transforming an insecure channel into a confidential channel (without or with limited authenticity) and then using a MAC to transform the confidential channel into a secure channel (see Section 5.2). Such a confidential channel is characterized by some form of malleability (see [22]).

Specifically, it is quite easy to see that if one uses OTP encryption over an insecure (rather than authenticated) channel, then the resulting channel is an XOR-malleable channel allowing the adversary E to XOR an arbitrary offset δ to the transmitted message. However, the message remains confidential (except for its length), and the adversary cannot modify the message in any other way. For example, he can not swap the first and the second half of the message (even though it might seem that swapping the ciphertext halves might result in swapping the plaintext halves).

4 An Abstract Theory of Systems

This section is based on abstract cryptography [18] and we refer to that paper for more details.

4.1 Abstraction in Cryptography

Before we explain constructive cryptography at a general abstract level, we first recall the discussion in [18] (also presented in [15]) of the role of abstraction in cryptography.

In every mathematical discipline one tries to identify the key concepts and to formalize them in an abstract manner. Abstraction means to eliminate irrelevant details from consideration, thereby focusing only on the relevant aspects of a problem or context. The purpose of abstraction is to provide, at the same time, simpler definitions, higher generality of results, simpler proofs, improved elegance, better didactic suitability, and, perhaps most importantly, new insights.

In many contexts, the highest achievable level of abstraction, once identified, appears natural and stable. For example, the natural mathematical concepts of a relation, a function, a graph, a group, a field, or a vector space capture exactly, in a minimal manner, the relevant notions. In contrast, in cryptography definitions, theorems, and proofs are generally highly technical and have substantial complexity due to the technical artifacts of the particular model (e.g. defining the computational model via Turing machines and communication via tapes, using asymptotic definitions of systems and protocols, defining efficiency as polynomial-time, using a particular adversarial model, etc.).

In view of this, it seems desirable to state cryptographic definitions in an abstract manner and to leave the technical aspects to a lower level of abstraction. This allows for simplicity (e.g. non-asymptotic definitions and proofs) and generality (e.g. simultaneous treatment of different security notions, like information-theoretic and computational).

4.2 Levels of Abstraction: Bottom-Up vs. Top-Down

The traditional approach in theoretical computer science, and more specifically in complexity theory and cryptography, is bottom-up. One first defines (at a low level) a computational model (e.g. a Turing machine or a circuit), based on which one defines the concept of an algorithm for the model and a communication model (e.g. based on tapes). One then defines a complexity notion for an algorithm (e.g. the number of steps), and then a notion of efficiency (e.g. polynomial in some parameter(s), in an asymptotic sense). Finally, based on all these notions, one defines the security of a cryptosystem, typically as the infeasibility for the adversary to win a specific game.

The paradigm shift proposed in [18] is to use a top-down approach. In order to state definitions and develop a theory, one starts from the other end, the highest possible level of abstraction, and proceeds downwards, introducing in each new lower level only the minimal necessary specializations of that level necessary for expressing what one wants to capture.

It is important to point out that theorems proved at a certain (high) level of abstraction are completely precise (as they are mathematical theorems). This is true without instantiations of the lower levels, which is exactly the point of abstraction. The definitions and theorems are inherited by the lower levels, provided (of course) that the lower levels satisfy the postulated properties or axioms of the higher levels. It is hence strictly more desirable to prove theorems at higher levels of abstraction; nothing is lost by doing this[7].

This paper deals only with high levels of abstraction. Modeling systems at a concrete level (with potentially tedious details), except for the examples given, is left to other papers (e.g. [22,20]).

4.3 Resources and Converters

At the highest level of abstraction, a *system* is an abstract object with *interfaces* by which it interacts with its environment and with other systems. Interfaces are labeled with elements of a label set. Two systems can be composed into a single system by connecting one interface from each system.

Specific types of systems, for example discrete systems as defined in the random system framework [14], are defined at the next lower level of abstraction (see [18]).

In this paper we consider only two special types of systems: resource systems and converter systems. In the statements we make about systems composed of resources and converters, we also consider distinguishers as a third type of system.

An *$\mathcal{I}$-resource system* (or simply *$\mathcal{I}$-resource*), usually denoted by capital letters (e.g. R or S) or by a symbol like SEC in the examples discussed earlier, is a system with interface label set $\mathcal{I}$ (e.g. $\mathcal{I} = \{1, \ldots, n\}$ or $\mathcal{I} = \{A, B, E\}$). Typically (but not only) one can think of each interface being accessible to one party. A *converter system* (or simply converter), usually denoted by a Greek letter (e.g. α or π) or by a symbol like $\mathsf{otp\text{-}enc}$, is a system with two interfaces, where one interface is designated as the *outside* interface and the other as the *inside* interface[8]. The inside interface of a converter α can be connected to interface $i \in \mathcal{I}$ of a resource system R; the outside interface of α serves as the new interface i of the combined system, which is again a resource system and is denoted $\alpha^i R$[9].

4.4 Cryptographic Algebras

We now formalize the notions introduced above.

Definition 1. A *cryptographic algebra* $\langle \Phi, \Sigma \rangle$ for an interface set $\mathcal{I}$ consists of a set Φ of $\mathcal{I}$-resources with a parallel composition operation $\|$,[10] a set Σ of

[7] For example, in algebra one proves theorems about groups, without having to mention examples of groups, and the point of doing so is that such a theorem applies to *every* group, i.e., every structure satisfying the group axioms.

[8] One can think of such a system as a protocol engine converting or transforming an interface of a resource into an interface with a different behavior.

[9] A system composed of a resource and converters has a star-shaped topology, with a resource in the center and a (possibly empty) chain of converter systems attached to each interface. The resulting system is again a resource with the same interface set.

[10] For every $i \in \mathcal{I}$, the i-interface of $R\|S$ can be thought of as consisting of the two i-interfaces of R and S merged into a single interface, by some addressing mechanism that is not (yet) of interest at this level of abstraction.

converters, and a mapping $\Sigma \times \Phi \times \mathcal{I} \mapsto \Phi$ defining the resource obtained when converter α is attached to interface i of resource R, denoted as $\alpha^i R$, such that

(i) Converter application at different interfaces commutes:
$$\alpha^i \beta^j R = \beta^j \alpha^i R$$
for all $i \neq j$, $R \in \Phi$, and $\alpha, \beta \in \Sigma$.

(ii) Attaching no converter is defined as a special *neutral* converter $\mathbf{1} \in \Sigma$ satisfying $\mathbf{1}^i R = R$ for all $i \in \mathcal{I}$ and $R \in \Phi$.

The commutativity condition of the above definition is a special case of composition-order independence [18], the statement that the order in which systems are composed does not matter[11].

One can naturally define serial and parallel composition operations on the converter set Σ as follows. Serial composition: $\alpha\beta$ (or $\alpha \circ \beta$) is defined by
$$(\alpha\beta)^i R := \alpha^i \beta^i R$$
for all i and R. This composition operation is associative because function composition is: $(\alpha\beta)\gamma = \alpha(\beta\gamma)$. Note that $\mathbf{1}\alpha = \alpha\mathbf{1} = \alpha$. Parallel composition: $\alpha\|\beta$ is defined by
$$(\alpha\|\beta)^i (R\|S) := \alpha^i R \,\|\, \beta^i S$$
for all i and $R, S \in \Phi$[12].

We also consider a pseudo-metric[13] on the space Φ of resources to measure the similarity or dissimilarity of resources. As discussed in Section 4.5, the metric is usually defined as the best distinguishing advantage for a certain class of distinguishers, but it is useful to consider an abstract pseudo-metric. Two special and natural properties of a pseudo-metric are captured in the following definition. They state that the pseudo-metric is non-expanding in the sense that $d(R, S)$ does not increase if one puts a resource T in parallel to R and S,[14] or if one connects a converter to the same interface of R and S.

Definition 2. A pseudo-metric d on Φ is *compatible* with the cryptographic algebra $\langle \Phi, \Sigma \rangle$ if
$$d(R\|R', S\|S') \leq d(R, S) + d(R', S') \qquad (3)$$
for all $R, R', S, S' \in \Phi$, and
$$d(\alpha^i R, \alpha^i S) \;\leq\; d(R, S) \qquad (4)$$
for all $i \in \mathcal{I}$, $R, S \in \Phi$ and $\alpha \in \Sigma$.

[11] This is what one has in mind when drawing a figure of a system composed of subsystems connected by lines, as the drawing does not preserve information about the order in which the various parts were drawn.

[12] Note that $(\alpha\|\beta)^i T$ need not be explicitly defined if T is not of a the form $T = R\|S$.

[13] A pseudo-metric on a set S is a function $d : S \times S \to \mathbf{R}^+$ such that $d(a, a) = 0$, $d(a, b) = d(b, a)$, and $d(a, c) \leq d(a, b) + d(b, c)$.

[14] Note that (3) is equivalent to $d(R\|T, S\|T) \leq d(R, S)$ and $d(T\|R, T\|S) \leq d(R, S)$.

4.5 Distinguisher-Based Pseudo-Metrics

A distinguisher D (for n-interface resources) can be defined as a system with $n+1$ interfaces, where n interfaces connect to the interfaces of a resource R and the other (outside) interface outputs a bit.

The typical pseudo-metrics in cryptography are distinguisher-based metrics, i.e., the distance between two resource systems is the best advantage a distinguisher in a certain class $\mathcal{D}$ of distinguishers can achieve:

$$d(R,S) = \Delta^{\mathcal{D}}(R,S) := \sup_{D\in\mathcal{D}} \Delta^{D}(R,S),$$

where $\Delta^{D}(R,S)$ is the advantage of D in distinguishing R and S.

The class $\mathcal{D}$ is, for example, either the set of *all* distinguishers (information-theoretic security) or the set of all *feasible* distinguishers (computational security). In the first case, one can distinguish between perfect security (the distance is 0) or statistical security (the distance is very small).

A distinguisher D emulating (internally) a converter $\alpha \in \Sigma$ at interface i induces a new distinguisher, denoted $D\alpha^i$, defined by

$$\Delta^{D\alpha^i}(R,S) = \Delta^{D}(\alpha^i R, \alpha^i S).$$

Similarly, a distinguisher D emulating a resource $T \in \Phi$ in parallel induces a new distinguisher, denoted $D[\cdot\|T]$, defined by

$$\Delta^{D[\cdot\|T]}(R,S) = \Delta^{D}(R\|T, S\|T).$$

$D[T\|\cdot]$ is defined analogously.

An important property of a distinguisher class $\mathcal{D}$ is that it is closed under the emulation of a converter, in the sense that $\mathcal{D}\Sigma^i \subseteq \mathcal{D}$, where $\mathcal{D}\Sigma^i = \{D\alpha^i | D \in \mathcal{D}, \alpha \in \Sigma\}$. In other words, a converter can be absorbed into a distinguisher without extending the distinguisher class: For $D \in \mathcal{D}$ and $\alpha \in \Sigma$, we also have $D\alpha^i \in \mathcal{D}$.

Similarly, $\mathcal{D}$ should also be closed under emulation of a resource, in the sense that $\mathcal{D}[\cdot\|\Phi] \subseteq \mathcal{D}$ and $\mathcal{D}[\Phi\|\cdot] \subseteq \mathcal{D}$, where for example $\mathcal{D}[\Phi\|\cdot] = \{D[T\|\cdot] : D \in \mathcal{D}, T \in \Phi\}$.

Lemma 1. *For a distinguisher class $\mathcal{D}$ for resources in Φ, the pseudo-metric $\Delta^{\mathcal{D}}$ is compatible with the cryptographic algebra $\langle\Phi,\Sigma\rangle$ if*

$$\mathcal{D}\Sigma^i \subseteq \mathcal{D}, \qquad \mathcal{D}[\cdot\|\Phi] \subseteq \mathcal{D}, \qquad \textit{and} \qquad \mathcal{D}[\Phi\|\cdot] \subseteq \mathcal{D}.$$

Proof. Since $\mathcal{D}\alpha^i \subseteq \mathcal{D}\Sigma^i \subseteq \mathcal{D}$ we have

$$\Delta^{\mathcal{D}}(\alpha^i R, \alpha^i S) = \Delta^{\mathcal{D}\alpha^i}(R,S) \le \Delta^{\mathcal{D}}(R,S),$$

which is (4). Similarly, since $\mathcal{D}[\cdot\|T] \subseteq \mathcal{D}[\cdot\|\Phi] \subseteq \mathcal{D}$ we have

$$\Delta^{\mathcal{D}}(R\|T, S\|T) = \Delta^{\mathcal{D}[\cdot\|T]}(R,S) \le \Delta^{\mathcal{D}}(R,S).$$

As mentioned, this inequality together with the dual inequality $\Delta^{\mathcal{D}}(T\|R, T\|S) \le \Delta^{\mathcal{D}}(R,S)$ implies (3). $\square$

If one considers the class of feasible distinguishers and converters, then one needs a feasibility notion for which the conditions of Lemma 1 are satisfied. Since one can repeatedly absorb converters into the class, the class needs to be defined asymptotically. For example, polynomial-time notions of feasibility, if properly defined, have this composition property. In fact, this can, in retrospect, be seen as a reason for working with polynomial-time notions in cryptography and complexity theory. However, as we have illustrated (see also [18]), an abstract treatment is possible, is simpler, and implies the corresponding statements for any composable feasibility notion, not just (a specific form of) polynomial-time.

In such an asymptotic definition of feasibility, the systems are actually asymptotic families of systems (indexed by some parameter) and the distance is a function of the parameter rather then a number. This view is compatible with the abstract view presented above. One can define a negligibility notion, which is a subset of the functions $\mathbf{N} \to \mathbf{R}^+$, containing the constant 0-function, and closed under multiplication with a feasible function. We do not expand further on these lower-level issues and refer to [18].

5 Constructive Cryptography for the Alice-Bob-Eve Setting

A standard cryptographic setup consists of two honest parties, Alice and Bob, connected by a certain communication resource (e.g., an insecure channel) that may be partially controlled by an adversary, Eve. This setup corresponds to a special case of abstract cryptography [18], where the interface set is $\mathcal{I} = \{A, B, E\}$, where A and B are assumed to be honest and E is assumed to be the adversary, and where one considers single resources rather than resource sets (called specification in [18]). The term *constructive cryptography* [16] here refers to the application of abstract cryptography to defining classical cryptographic primitives in a constructive way. We refer also to [22,19].

5.1 Defining "Secure Construction"

In the following, we consider resources with interface set $\mathcal{I} = \{A, B, E\}$, where E is the adversary interface. We will define what it means to "securely construct" a resource S from a resource R using converters π_1 for A and π_2 for B. Recall the discussion in Section 3.

Since the party E is only introduced as a thought experiment to define security, it can not be assumed to be present. A resource can be modeled as having two modes, one mode when E is not present and one mode when E is present and makes potential use of its assumed power. There are several ways to model such a two-mode resource. A natural one is to assume a special converter $\perp$ which, when attached to the E-interface, puts the resource into the "no adversary" mode. In other words, the two modes of a resource R are $\perp^{\mathsf{E}} R$ and R (where, in the latter, E has direct access to R).

Concretely, the converter $\perp$ can for example be thought of as shielding the interface to the outside (i.e., to a distinguisher) and setting some kind of flag at its inside interface which tells the resource to behave in a certain restricted mode (corresponding to E not being present). A resource could for example be (but does not have to be) modeled as taking as an initial input a "cheating bit" c at interface E. If the bit is set to $c = 1$, then the interface provides additional functionality (like allowing E to read and/or modify a message) compared to the case $c = 0$, which corresponds to E not being present. This can be interpreted as follows: An adversary E sets the bit to 1 in order to acquire additional power. In this view, $\perp$ is a converter which sets $c = 0$ (at the inside interface) and shields the outside interface. For example, an insecure channel can be modeled by specifying that if $c = 0$, then the message(s) input by A are delivered to B, whereas if $c = 1$, then E receives a message sent by A and can replace it by an arbitrary message, which is delivered to B. Similarly, a secure channel with deletion feature can be modeled by specifying that if $c = 0$, then the message(s) input by A are delivered to B, whereas if $c = 1$, then E learns the message length and can input a bit indicating whether the message should be delivered to B or deleted.

The two modes must be considered separately, resulting in two conditions which can be called availability (E not present) and security (E present).

Definition 3. Consider a cryptographic algebra $\langle \Phi, \Sigma \rangle$ for interface set $\mathcal{I} = \{A, B, E\}$ and a pseudo-metric d on Φ. For resources R and S we say that protocol (π_1, π_2) for $\pi_1, \pi_2 \in \Sigma$ *(securely) constructs* S from R, within ε, denoted

$$R \overset{(\pi_1,\pi_2,\varepsilon)}{\longrightarrow} S,$$

if the following two conditions (availability and security) are satisfied:

1. $d(\pi_1^A \pi_2^B \perp^E R, \perp^E S) \leq \varepsilon$
2. $\exists \sigma \in \Sigma : \quad d(\pi_1^A \pi_2^B R,\ \sigma^E S) \leq \varepsilon.$

In the one-time pad example (Section 3), only the second condition (security) was discussed. The first condition (availability) was not mentioned, but of course it is necessary, and indeed satisfied for encryption (in particular the one-time pad) as explained in Section 3. Otherwise a secure channel (with deletion feature for E) could be constructed from a communication channel that never delivers a message to B, which obviously would not make sense. Here we do not discuss the first condition further.

5.2 Composability

A notion of "construction" must be composable to be useful. For example, if a certain scheme constructs a secret key from an authenticated communication channel (e.g. a key agreement protocol like a variant of the Diffie-Hellman protocol [9]), then composing such key generation with encryption (and another authenticated channel) should construct a secure channel. Similarly, since a secure MAC constructs an authenticated channel from a secret key and an insecure channel, and encryption constructs a secure channel from an authenticated channel and (another) secret key, the combination of MAC and encryption should construct a secure channel from an insecure channel and two secret keys (see [22]). This corresponds to the well-known encrypt-then-MAC paradigm. Note that the order (encrypt-then-MAC) in which converters are applied to the message is reversed compared to the order in which transformations are applied. We refer to [22] for a discussion of this and of the dual MAC-then-encrypt paradigm.

The above are two examples of serial composability. We also need parallel composability, as described in the following theorem. Composability was defined abstractly in [18], and here we consider a special case.

Theorem 1. *The construction notion* $R \overset{(\pi_1,\pi_2,\varepsilon)}{\longrightarrow} S$ *as defined in Definition 3 is generally composable if the pseudo-metric* d *is compatible with the cryptographic algebra, i.e., we have:*

$$\begin{array}{lllll} \textit{(i)} & R \overset{(\pi_1,\pi_2,\varepsilon)}{\longrightarrow} S \ \wedge \ S \overset{(\pi_1',\pi_2',\varepsilon')}{\longrightarrow} T & \Longrightarrow & R \overset{(\pi_1'\pi_1,\pi_2'\pi_2,\varepsilon+\varepsilon')}{\longrightarrow} T; \\ \textit{(ii)} & R \overset{(\pi_1,\pi_2,\varepsilon)}{\longrightarrow} S \ \wedge \ R' \overset{(\pi_1',\pi_2',\varepsilon')}{\longrightarrow} S' & \Longrightarrow & R\|R' \overset{(\pi_1'\|\pi_1,\pi_2'\|\pi_2,\varepsilon+\varepsilon')}{\longrightarrow} S\|S'; \\ \textit{(iii)} & R \overset{(\mathbf{1},\mathbf{1},0)}{\longrightarrow} R. \end{array}$$

Conditions 2 and 3 together imply

$$R \overset{(\pi_1,\pi_2,\varepsilon)}{\longrightarrow} S \quad \Longrightarrow \quad R\|T \overset{(\pi_1\|\mathbf{1},\pi_2\|\mathbf{1},\varepsilon)}{\longrightarrow} S\|T,$$

which can be interpreted as *context-insensitivity* in the sense that resources (e.g. T) available in parallel (a context) does not invalidate a construction statement. (The analogous statement with T on the left side also holds.) Note that, for example, if Φ and Σ are the sets of feasibly implementable resources and converters, respectively, then the context-insensitivity does not hold with respect to, say, a factoring oracle T (if factoring is computationally hard). But such a T would not be in Φ.

Proof. The theorem makes six statements that can be proved independently: For each of (*i*) to (*iii*) we have to prove conditions 1 and 2 of Definition 3. Here we only prove one of these statements.

To prove condition 2 claimed by (*i*), assume that $R \overset{(\pi_1,\pi_2,\varepsilon)}{\longrightarrow} S$ and $S \overset{(\pi_1',\pi_2',\varepsilon')}{\longrightarrow} T$ are satisfied. By definition, this implies that

$$d(\pi_1^A\pi_2^B R,\ \sigma^E S) \leq \varepsilon$$

for some $\sigma \in \Sigma$, as well as

$$d(\pi_1'^A\pi_2'^B S,\ \sigma'^E T) \leq \varepsilon'$$

for some $\sigma' \in \Sigma$. We need to prove that there exists a simulator $\hat{\sigma}$ such that

$$d((\pi_1'\pi_1)^A(\pi_2'\pi_2)^B R,\ \hat{\sigma}^E T) \ \leq \ \varepsilon + \varepsilon'. \tag{5}$$

We prove that this holds for $\hat{\sigma} = \sigma\sigma'$. Commutativity of converter application at different interfaces implies

$$(\pi_1'\pi_1)^A(\pi_2'\pi_2)^B R \;=\; \pi_1'^A\pi_1^A\pi_2'^B\pi_2^B R \;=\; \pi_1'^A\pi_2'^B\pi_1^A\pi_2^B R.$$

Using (4), namely the compatibility of d, we obtain

$$\begin{aligned} d((\pi_1'\pi_1)^A(\pi_2'\pi_2)^B R,\ \pi_1'^A\pi_2'^B\sigma^E S) &= d(\pi_1'^A\pi_2'^B\pi_1^A\pi_2^B R,\ \pi_1'^A\pi_2'^B\sigma^E S) \\ &\le d(\pi_1^A\pi_2^B R,\ \sigma^E S) \\ &\le \varepsilon. \end{aligned}$$

Using (4) we also obtain

$$d(\sigma^E\pi_1'^A\pi_2'^B S,\ \sigma^E\sigma'^E T) \;\le\; d(\pi_1'^A\pi_2'^B S,\ \sigma'^E T) \;\le\; \varepsilon'.$$

Using $\sigma^E\pi_1'^A\pi_2'^B S \;=\; \pi_1'^A\pi_2'^B\sigma^E S$ (commutativity) and $\hat{\sigma} = \sigma\sigma'$,[15] we thus have

$$d(\pi_1'^A\pi_2'^B\sigma^E S,\ \hat{\sigma}^E T) \;\le\; \varepsilon'.$$

The triangle inequality now yields (5):

$$\begin{aligned} &d((\pi_1'\pi_1)^A(\pi_2'\pi_2)^B R,\ \hat{\sigma}^E T) \\ &\quad\le\; d((\pi_1'\pi_1)^A(\pi_2'\pi_2)^B R,\ \pi_1'^A\pi_2'^B\sigma^E S) + d(\pi_1'^A\pi_2'^B\sigma^E S,\ \hat{\sigma}^E T) \\ &\quad\le\; \varepsilon + \varepsilon'. \end{aligned}$$

□

We note that this proof of composition holds in complete generality, independently of how one defines technical aspects at lower layers. It would, for example, even hold for systems connected via an analog communication mechanism, and it holds for any composable notion of feasibility and any model of computation.

6 History and Related Work

The idea that cryptographic schemes can be described as transformations of channel security properties, in a constructive sense, was first proposed in the author's lecture notes in the early 90's and was published in [21]. This approach to explaining cryptography, and protocols combining the use of several cryptographic schemes, as channel transformations, also making trust assumptions explicit, has been called the •-calculus (see for example [21,22,23]). Constructive cryptography as discussed here can be seen as defining the semantics of transformations in the •-calculus, which was originally used with only an intuitive semantics, mainly for didactic purposes. We do not elaborate further on the •-calculus.

The so-called "ideal-world real-world" paradigm in cryptography emerged in the context of secure multi-party computation which was (and is) understood

[15] Note that the order of composing protocols and simulators is different. The reason can be seen by drawing a figure.

as the emulation of an (ideal) trusted party that computes a certain function or specification, in a real world where only communication channels are available to the parties. It was introduced formally in the frameworks of Canetti [4] (universal composability, called UC) and of Backes, Pfitzmann and Waidner [24,2] (reactive simulatability), based on the simulation paradigm.

In these frameworks, the basic idea is to consider an ideal system (often called a functionality) capturing the goal one wants to securely realize, when given a complete asynchronous network and possibly a set-up, using a protocol specifying what the honest parties have to do. The definition of what it means to "securely realize" involves an adversary who can corrupt certain parties, and captures the idea that whatever the adversary can achieve in the real world he could also achieve in the ideal world. This is made precise by means of a so-called *simulator*, an ingenious concept introduced by Goldwasser, Micali, and Rackoff [10] to define zero-knowledge protocols.

The frameworks [4,24,2] are technically quite specific and have several drawbacks and limitations explained in the abstract cryptography framework [18]. This framework gives a new, direct semantics to the "ideal-world real-world" paradigm based on the concept of a resource isomorphism. In this theory, simulators are only a proof technique, not part of the (security) definition. Actually, in this framework there is not necessarily a central adversary, and simulators are local (as opposed to monolithic).

Constructive cryptography was developed in parallel with abstract cryptography [18], which is joint work with Renato Renner, from which we borrow substantially, including the use of the abstract theory of systems. Constructive cryptography can be seen both as a fragment and as an application of abstract cryptography. Basic ideas of constructive cryptography were also described in [19].

In view of the "ideal-world real-world" paradigm, constructive cryptography appears natural. Nevertheless, previous works on the security of cryptographic schemes work with game-based definitions. We refer to [20] for a discussion of the game-based approach in view of constructive cryptography. Some papers have defined ideal functionalities for certain secure communication primitives. Shoup [25] investigated functionalities for secure key exchange. Bellare et al. [3] propose a modular approach to designing authentication and key exchange protocols, but their approach mixes the game-based and the "ideal-world real-world" approach. This is also true for [6]. Canetti and Krawczyk [7] propose ideal functionalities in the UC-framework for key exchange and secure channels, but their approach is quite technical and involves apparently unnecessary artifacts (e.g. a so-called non-information oracle). These works, and many more, can now be reexamined in the spirit of constructive cryptography.

Many works that define ideal functionalities to explain traditional cryptography actually define functionalities for the cryptographic schemes rather than the channels or keys. For example, Canetti [5] defines an ideal functionality for a digital signature scheme. In constructive cryptography, this appears surprising, as a digital signature scheme corresponds to a pair of *converters*, not a *resource*.

This illustrates that constructive cryptography, while using ideas of previous frameworks, is intrinsically different, in addition to being phrased at a more abstract and general level.

There exists a vast literature on applying formal methods to the design and the analysis of security protocols, and some of them deal with the composition of protocols (e.g., see [8,11,12,23,26] and the references therein). It is beyond the scope of this extended abstract to give a detailed comparison between these approaches and our approach, but we mention a few intrinsic differences.

The most important difference between works in the formal-methods community and the cryptography community is in the *kind* of (mathematical) statement one makes, not in the manner how these statements and the proofs are formalized. We explain three major aspects in which the statements made in the two approaches differ.

First, security is often defined as the absence of a certain class of *attacks*, i.e., the inexistence of a trace in the class of considered attack traces. A major question is whether the class of attack traces captures all relevant attacks. More precisely, the question is what the absence of attack traces of a certain type really means. For example, it should allow one to conclude the security of a protocol that makes use of the given protocol as a subprotocol. We refer to [11] for a discussion of work on protocol composition. In contrast, security in constructive cryptography is defined by the specification of the ideal resource, which characterizes completely what a certain entity (e.g. the adversary E) can do. The notion of attack does not exist in this view, as everything relevant is said by the resource specification. As a consequence, in constructive cryptography, composition follows by a general composition theorem, that is, by design of the kind of statement one makes.

Second, most works applying formal methods make use of a specific idealized model of cryptography, typically the Dolev-Yao model, which abstracts away many relevant aspects. For example, it does not capture that the message length could leak. We refer to [1] for a more detailed discussion of what statements derived for the Dolev-Yao model can actually mean.

Third, realistic statements in cryptography are generally of a probability-theoretic nature, for example stating a bound on the probability that the adversary can break a system or on the advantage in distinguishing two systems. Most works on formal methods do not consider probabilities.

Work based on formal methods comes with the promise of being completely rigorous. Of course, the precision of security statements is important, no matter what kind of statement one makes, and formal methods of various types can provide tools to achieve precision and proof automation. But rigor and precision does not require a statement to be made in a particular formal language. We point out that in this paper, only the statements of Sections 4 and 5 are rigorous, while Sections 2 and 3 are intentionally stated in a less formal language. However, discrete systems and their composition can be formalized in the language of random systems [14] (see also [17]). An interesting open question is how formal methods can be applied to derive statements in constructive cryptography.

On the other hand, since in our approach protocols can be decomposed into elementary steps that are quite easy to prove, the security proof of a complex protocol can actually be simple enough that it may not be necessary to apply formal methods to capture a statement and its proof. We believe that precision and simplicity are not contradictory and that one should always strive for simplicity, without sacrificing rigor.

7 Conclusions and Future Work

Based on [18], we have considered the common cryptographic setting with two honest parties Alice and Bob and an adversary Eve. We have proposed to model cryptography in this setting as a constructive discipline in which resources (e.g. channels) are constructed from resources (e.g. channels and keys), by a cryptographic scheme that corresponds to a pair of converters (constituting the protocol engines) for Alice and Bob.

While the constructive approach could in principle also be phrased in an existing framework such as the UC-framework [4],[16] we present our theory at an abstract level where only the relevant aspects need to be considered, resulting in simplicity and minimality of the arguments. For example, a feasibility notion must, abstractly speaking, only be such that the feasible distinguisher class is closed under emulation of feasible resources and converters. It is possible, but not necessary, to define feasibility as some form of polynomial-time. Asymptotics, Turing machines, and other artifacts that are an integral part of any conventional security definition in cryptography, are not necessary at this abstract level, without loss of precision.

Our approach illustrates the importance of distinguishing between two system types, converters and resources. Cryptographic schemes correspond to converters, not resources. For example, in our view, secrecy is an attribute of a channel, not necessarily a property of an encryption scheme. The (often considered) security goal of achieving secrecy and authenticity means to construct a resource (channel) with these two attributes, not to design an encryption scheme that has two properties which can be called secrecy and integrity. This construction can be divided into two composable steps (see [22]), in two alternative ways: authenticate-then-encrypt and encrypt-the-authenticate.

Constructive cryptography makes precise the intuitive high-level understanding of what cryptography and cryptographic protocols achieve (e.g. constructing a secure channel), separating it from rather involved mathematical security definitions and proofs. One can understand cryptography without understanding security definitions. This separation of concerns suggests a new approach in teaching where the high-level understanding of cryptography can be treated precisely, without need for understanding cryptographic definitions (e.g. simulators), in a

[16] Note that abstract cryptography [18] is more general and can *not* be phrased in the UC-framework, as it not only considers a single (central) adversary but models more generally parties with conflicting goals.

general self-contained course on information security, while security definitions and proofs can be treated in a specialized course on cryptography.

Many relevant topics could not be discussed in this paper and are deferred to subsequent papers. Three of them are addressed below.

First, symmetric encryption is the only cryptographic primitive that was actually explained in this paper as a constructive step. Other cryptographic primitives such as message authentication codes, public-key encryption, key-agreement protocols, and digital signature schemes, can also be described as constructive steps, in the spirit described in [21], leading to constructive security definitions for these schemes.

Second, the previous game-based security definitions can be explained in the context of constructive cryptography, giving them a (constructive) semantics. Usually one of the known game-based definitions, if lifted to a more abstract level where one does not necessarily have to talk about asymptotics, polynomial-time, etc., is equivalent to a given constructive security definition. For the case of symmetric encryption, the relation between constructive and game-based definitions is investigated in [20].

Third, we can now design a complete protocol constructing a secure communication channel between two entities A and B from a secret key shared by them and an insecure network (which is a resource involving many entities, including A and B). In addition to the two constructive steps of applying message authentication and encryption (see [22]), the roles of addresses (sender and receiver) and of sequence numbers and/or nonces must also be made explicit as constructive steps. For a protocol designed in this manner, attacks (e.g. replay attacks, reflection attacks, etc.) sometimes successful against conventionally designed protocols, can be excluded by design and need therefore not even be considered. In a certain sense, the constructive paradigm is type-safe, avoiding unexpected effects when different protocol components are combined. Future protocol suites for key management, secure communication, certification, etc., can be designed (and hence be proven secure) in the constructive cryptography paradigm.

Acknowledgements. Constructive cryptography was developed in parallel with abstract cryptography [18]. The collaboration with Renato Renner was essential. I have had numerous very helpful discussions and an on-going collaboration with Björn Tackmann who also provided continuous motivation. Many current and former group members have provided useful feedback in the context of the courses I taught at ETH, and with several colleagues I have had many useful discussions. These people include Divesh Aggarwal, Kfir Barhum, David Basin, Sandro Coretti, Grégory Demay, Martin Hirt, Dennis Hofheinz, Thomas Holenstein, Christoph Lucas, Sebastian Mödersheim, Krzysztof Pietrzak, Bartosz Przydatek, Dominik Raub, Pavel Raykov, Christoph Sprenger, Stefano Tessaro, Zuzana Trubini(-Beerliova), Luca Viganò, Stefan Wolf, and Vassilis Zikas.

References

1. Backes, M., Hofheinz, D., Unruh, D.: CoSP: A general framework for computational soundness proofs. In: ACM Conference on Computer and Communications Security, pp. 66–78 (2009)
2. Backes, M., Pfitzmann, B., Waidner, M.: A general composition theorem for secure reactive systems. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 336–354. Springer, Heidelberg (2004)
3. Bellare, M., Canetti, R., Krawczyk, H.: A modular approach to the design and analysis of authentication and key exchange protocols. In: Proc. 30th Annual Symposium on the Theory of Computing (STOC), pp. 419–428. ACM, New York (1998)
4. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS, pp. 136–145 (2001)
5. Canetti, R.: Universally composable signature, certification, and authentication. In: 17th IEEE Computer Security Foundations Workshop (CSF 2004), p. 219 (2004)
6. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
7. Canetti, R., Krawczyk, H.: Universally composable notions of key exchange and secure channels. In: Knudsen, L. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 337–351. Springer, Heidelberg (2002)
8. Cortier, V., Delaune, S.: Safely composing security protocols. Formal Methods in System Design 34(1), 1–36 (2009)
9. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory 22, 644–654 (1976)
10. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. 18(1), 186–208 (1989)
11. Gross, T., Mödersheim, S.: Vertical protocol composition. In: 24th IEEE Computer Security Foundations Workshop (CSF 2011) (to appear, 2011)
12. Guttman, J.D., Thayer, F.J.: Protocol Independence through Disjoint Encryption. In: Computer Security Foundations Workshop, pp. 24–34 (2000)
13. Maurer, U.: Secret key agreement by public discussion from common information. IEEE Transactions on Information Theory 39(3), 733–742 (1993)
14. Maurer, U.: Indistinguishability of random systems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 110–132. Springer, Heidelberg (2002)
15. Maurer, U.: Abstraction in cryptography. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, p. 465. Springer, Heidelberg (2009)
16. Maurer, U.: Constructive cryptography - a primer. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, p. 1. Springer, Heidelberg (2010)
17. Maurer, U., Pietrzak, K., Renner, R.: Indistinguishability amplification. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 130–149. Springer, Heidelberg (2007)
18. Maurer, U., Renner, R.: Abstract cryptography. In: The Second Symposium in Innovations in Computer Science, ICS 2011, pp. 1–21. Tsinghua University Press, Beijing (January 2011)
19. Maurer, U., Renner, R., Wolf, S.: Unbreakable keys from random noise. In: Tuyls, P., et al. (eds.) Security with Noisy Data, pp. 21–44. Springer, Heidelberg (2007)
20. Maurer, U., Rüedlinger, A., Tackmann, B.: Confidentiality and integrity revisited (manuscript in preparation)

21. Maurer, U., Schmid, P.E.: A calculus for security bootstrapping in distributed systems. Journal of Computer Security 4(1), 55–80 (1996); appeared also In: Gollmann, D. (ed.) ESORICS 1994. LNCS, vol. 875, pp. 175–192. Springer, Heidelberg (1994)
22. Maurer, U., Tackmann, B.: On the soundness of authenticate-then-encrypt. In: ACM Conference on Computer and Communications Security, pp. 505–515 (2010)
23. Mödersheim, S., Viganò, L.: Secure pseudonymous channels. In: Backes, M., Ning, P. (eds.) ESORICS 2009. LNCS, vol. 5789, pp. 337–354. Springer, Heidelberg (2009)
24. Pfitzmann, B., Waidner, M.: Composition and integrity preservation of secure reactive systems. In: ACM Conference on Computer and Communications Security, pp. 245–254 (2000)
25. Shoup, V.: On formal models for secure key exchange. IBM Research report, no. RZ 3120 (April 1999)
26. Sprenger, C., Basin, D.A.: Developing security protocols by refinement. In: ACM Conference on Computer and Communications Security, pp. 361–374 (2010)