# New approaches to formalizing security

Seminar report for Research Seminar in Cryptography

Pille Pullonen

Supervisor: Peeter Laud

December 12, 2013

## 1 Introduction

Recently the frameworks of abstract and constructive cryptography have been proposed by Maurer et al [MR11, Mau11, Mau10, Mau09]. These aim to redefine the common view of cryptographic security proofs and definitions. However, in a sense these ideas can be seen both as a protest and a generalisation of the current state of the art.

First, a protest because current definitions and proofs are built from the very low details of the computation model, such as the Turing machine, but often rely on the general understanding of these models rather than actual details. These make the corresponding results less formal than we could desire. Secondly, a generalisation because an important observation behind the abstract cryptography is that different specification of low level details may still lead to more or less equivalent high level results. Hence, maybe it is actually reasonable to enable well formalised cryptographic arguments on a higher abstraction level to achieve general results that apply for all lower level specifications.

This work is mostly based on the theory of abstract cryptography [MR11] and also uses the constructive cryptography [Mau10, Mau11] which is an application of abstract cryptography. This overview is intended to introduce these theories and offer some additional comments and insights. In addition, additional examples are given for the numerous definitions, that hopefully make understanding these ideas simpler.

Firstly, Section 2 recalls some of the main challenges of cryptographic protocol design and uses them as motivation for the abstract and constructive cryptography frameworks. Secondly, Section 3 gives an overview of the abstract cryptography definitions with several examples of the introduced concepts. Section 4 continues the discussion about abstract cryptography by introducing some examples of using this theory as well as highlighting some open questions. Following Section 5 gives a subcase of abstract cryptography called the constructive cryptography. Finally, Section 6 concludes this report.

## 2 Motivation

There are different security requirements and definitions for different systems, which is only natural because different use-cases call for very different solutions. However, one central question in cryptographic protocol design is the composability of proposed systems. Often it is easier to think of a system in an isolated, so called *stand-alone*, world where it is assumed that the participants just use the proposed protocol and do nothing else. Although analysing security in the stand-alone setting can give us some insight about the strengths of the protocol, it is usually insufficient model of the real life.

In a realistic world these same parties can be engaged in different secure or insecure protocols over time and at the same time with our protocol. This introduces different views of composability: *serial*, *parallel* and *general* composability. Ideally we require our proposed protocol to be secure independently of what else the participants might do. Especially, we say that a protocol is secure with respect to general composability if it remains secure even if other protocols are run serially or in parallel with this protocol. However, there are some rules for the composition. We commonly have a black-box approach, where one important step is that only the outputs and inputs and not the intermediate messages can be used outside the protocol.

Over time different formalisations for composability have been proposed: *universal composability* (UC) [Can01], *reactive simulatability* (RSIM) [PW00, PW01] and *inexhaustible interactive Turing machine* (IITM) [KT13].

Namely, UC and RSIM frameworks build up from the small details, for example modelling everything as state-transition machines, specifying means of communication and so on. From these specifications they reach the composition theorems for different cases, for example separately for synchronous and asynchronous communication. This can be seen as a motivating step for abstract cryptography because if analogous results can be obtained for different exact specifications then maybe this result could be obtained without the low level details.

Secondly, the motivation for constructive cryptography also begins from the general composition. Namely, the composition proofs are common for cryptographic protocols, but not for the primitives. However, building complex protocols and proving their security would be simpler if the primitives are known to be securely composable. Constructive cryptography mainly proposes to give composable security definitions also for cryptographic primitives. In addition, constructive cryptography also uses ideas from abstract cryptography.

Abstract cryptography is a general theory and previous theories as UC and RSIM can be seen as special cases of this theory. On the other hand, constructive cryptography is a more concrete theory and could also be used within UC or RSIM.

The second starting point for abstract cryptography is the understanding that the central adversaries that are commonly used in security analysis are not always sufficient. Namely, we are occasionally interested in analysing the cases where the parties are not allowed to collude, but they might independently behave maliciously. Security against a central adversary seems to be a stronger requirement, however sometimes we are interested in analysing the precise case of independently corrupted parties where the central adversary is unsuitable. For example, this allows for analysing special requirements of non-collusion and anonymity. The security proof for a security against a central adversary would require building a simulator that can jointly simulate the views of all corrupted parties to the adversary. On the other hand, security proofs for non-colluding *local* adversaries only contain local simulators that have to simulate the view of one corrupted party [CV12].

# 3 Overview of abstract cryptography

The main aim of abstract cryptography is to change the viewpoint of cryptographic definitions from highly technical to more general and higher level that is independent of the technical details. The idea is that each result should be defined at the highest possible abstraction level so that it holds for all lower levels.

Systems can be viewed at different abstraction levels, where the first level is the most abstract one and each next level adds more precise details. In the highest level each system is an algebraic object that can be described as a box with interfaces. Each lower level defines additional details such as exact notion of efficiency or physical aspects of the computation. For example, at the highest level we only have interfaces but no description about how to communicate over these interfaces. The protocols are introduced at a lower abstraction layer because they describe how the communication should works.

Constructive cryptography, an application of abstract cryptography, is a new look at cryptography definitions where security should be defined in terms of a scheme constructing a desired resource from a weaker resource. The main bonus from this approach is that the definitions are such that they are also composable by design [Mau11]. Specifics of constructive cryptography are introduced in Section 5.

## 3.1 Vocabulary

This section introduces the important definitions for the theory of abstract cryptography [MR11]. We start from the abstract concepts and gradually introduce how to use them for the purpose of cryptography.

### 3.1.1 Reductions and constructions

The paradigm of constructing more complex systems from simpler components is called *step-wise refinement*. Overall construction of a system can be viewed as a tree where each leaf is an initial component and each inner node is a *constructor* that fixes how its children are used to build a new component. In the top-down approach we say that the constructor *reduces* the components to its child components. On the contrary, in the bottom-up approach we say that the component is *realised from* its children using the given constructor.

We denote that set of components $S$ can be realised from a set of components $R$ using a constructor $\alpha$ as $R \overset{\alpha}{\to} S$. For example, an authenticated channel is reduced to two resources: insecure communication channel and a secret key. The corresponding constructor is the description of the converters that compute and verify the MAC. Formally we define these as follows.

**Definition 1** (Component set)**.** A component set is a set $\Omega$ with a parallel composition operation $||$.

**Definition 2** (Constructor set)**.** A constructor set $\Gamma$ is a set with a serial composition operation $\circ$, parallel composition operation $|$ and an identity element $\mathsf{id}$.

Note that we do not require any specific properties from these operations. The names *serial* or *parallel* composition operations are only used for giving some insight to the following usage of these operations.

**Definition 3** (Reduction)**.** A reduction for a component set $\Omega$ and constructor set $\Gamma$ is a subset of $\Omega \times \Gamma \times \Omega$. As noted previously, if $(R, \alpha, S)$ is in a reduction then we denote $R \overset{\alpha}{\to} S$.

For example, we can say that an authenticated channel can be reduced to a network channel and a secret key. Here, $R$ would be the network and the key, the constructor would be a message authentication algorithm and $S$ would be the resulting authenticated network channel.

An important observation is that anything can be a component or a constructor. In addition, reductions are defined over sets that are only specified by defined operations. Hence, reductions are an abstract concept. However, they are interesting because we can also define composability at this abstract level. We define different flavours of composability as follows.

**Definition 4** (Serially composable reduction)**.** A reduction $\longrightarrow$ is called serially composable if it is transitive for a composition of the constructors as

$$R \xrightarrow{\alpha} S \ \wedge \ S \xrightarrow{\beta} T \ \Rightarrow \ R \xrightarrow{\alpha\circ\beta} T$$

and there exists an identity constructor $\mathsf{id}$ that gives

$$R \xrightarrow{\mathsf{id}} R \ .$$

From these conditions we also know that in a serially composable reduction $\alpha \circ \mathsf{id} = \mathsf{id} \circ \alpha = \alpha$. In addition, we know that the serial composition must be associative as if we consider three reductions $R \xrightarrow{\alpha} S$, $S \xrightarrow{\beta} T$ and $T \xrightarrow{\gamma} U$ then we can use the transitivity to get $R \xrightarrow{\alpha\circ(\beta\circ\gamma)} U$ and $R \xrightarrow{(\alpha\circ\beta)\circ\gamma} U$. Therefore, $(\alpha \circ \beta) \circ \gamma = \alpha \circ (\beta \circ \gamma)$.

**Definition 5** (Context insensitive reduction)**.** The constructor set $\Gamma$ is called context insensitive, if

$$R \xrightarrow{\alpha} S \ \Rightarrow \ R||T \xrightarrow{\alpha|\mathsf{id}} S||T \ \wedge \ T||R \xrightarrow{\mathsf{id}|\alpha} T||S \ .$$

Context insensitivity can be thought of as a property that allows us to look at a reduction without analysing if there are other resources $T$ in parallel with the resources that we reduce. This can be further combined with another reduction based on $T$ to obtain parallel composability based on the transitivity relation.

**Definition 6** (Parallelly composable reduction)**.** A reduction $\longrightarrow$ is called parallelly composable if

$$R \xrightarrow{\alpha} S \;\wedge\; R' \xrightarrow{\alpha'} S' \;\Rightarrow\; R||R' \xrightarrow{\beta} S||S'$$

where $\beta = (\alpha|\mathsf{id}) \circ (\mathsf{id}|\alpha')$. We could also say $\beta = \alpha|\alpha'$.

A reduction is *parallelly composable* if it is transitive and context-insensitive. Finally, a reduction is *generally composable* if it is both serially and parallelly composable which is actually the same as serially composable and context insensitive.

Therefore, although we did not give any restrictions to the properties of operations on component or constructor sets, we have restrictions on these operations when we are interested in composability.

**Definition 7** (Sound reduction)**.** A reduction for a component set $\Omega$ and constructor set $\Gamma$ is sound for step-wise refinement if for every tree the following holds. At every node $v$ we have

$$S_{v_1}|| \dots ||S_{v_d} \xrightarrow{\alpha_v} S_v$$

meaning the local property of the tree where every node is constructed from its children, then for the root $r$ and leaves $l_1, \dots, l_k$ we have

$$S_{l_1}|| \dots ||S_{l_k} \xrightarrow{\alpha_r^*} S_r$$

where

$$a_v^* = \begin{cases} \mathsf{id} & \text{if } v \text{ is a leaf} \\ \alpha_v \circ (\alpha_{v_1}^*| \dots |\alpha_{v_d}^*) & \text{otherwise.} \end{cases}$$

General composability in terms of the construction (or reduction) tree means that the root is constructed from the leaves by the composition of the intermediate constructors. Hence, general composability means that step-wise refinement is meaningful. Moreover, a reduction is sound for step-wise refinement if and only if it is generally composable.

Hence, we know the requirements for composability and in the following we define how systems can be built from component and constructor sets.

### 3.1.2 Choice settings

An *n-choice setting* is a general setting where $n$ choices are somehow made from sets $(A_1, \dots, A_n)$ so that the $i$'th choice $a_i \in A_i$. Usually, we assume that each choice is made by a separate party but the framework is not limited to this assumption. Each tuple of choices results in an *effect*, which can be considered as an outcome of the choices.

**Definition 8** (Choice setting)**.** An $n$-choice setting $R$ is a function

$$R : A_1 \times A_2 \times \ldots \times A_n \to \Lambda$$

where $A_i$ are called the $i$'th choice domain and are arbitrary sets, and $\Lambda$ is called the effect space and is a set with an equivalence relation $\simeq$.

In the following we are working with the cases here the effect is actually a new choice setting, but in general in can be anything. We say that two choice settings are isomorphic if the same effects can be obtained from the choices.

**Definition 9** (Choice setting isomorphism)**.** Two $n$-choice settings $R$ and $S$ with choice domains $A_1, \ldots, A_n$ and $B_1, \ldots, B_n$ are isomorphic via a relation $\rho = \rho_1 \times \ldots \times \rho_n$ where $\rho_i$ is a $(A_i, B_i)$-relation, relative to the equivalence relation $\simeq$ on the effect space, if for any choices $(a_1, \ldots, a_n)$ and $(b_1, \ldots, b_n)$ where $a_i \in A_i$ and $b_i \in B_i$ we have

$$(\forall a_i \rho_i b_i) : R(a_1, \ldots, a_n) \simeq S(b_1, \ldots, b_n) \ .$$

The isomorphism is denoted by

$$R \overset{\rho}{\simeq} S \ .$$

The main implication of isomorphic choice settings is that the parties would not prefer one setting over another. A special case of isomorphic settings are a setting and an *extension* of the same system. In the extension each party can make the same choices as before and also some new choices that are equivalent to some previously available choices. For example, if the effect is defined by the parties either choosing an odd or an even number, the settings with choices $\{0, 1, 2, 3\}$ would be an extension of $\{0, 1\}$.

**Definition 10** ($n$-choice setting specification)**.** An *$n$-choice setting specification* (or $n$-specification) $\mathcal{R}$ is a set of $n$-choice settings and a tuple $(\hat{A}_1, \ldots, \hat{A}_n)$ such that for every choice setting $(A_1, \ldots, A_n) \in \mathcal{R}$ we have $\hat{A}_i \subseteq A_i$.

The set $\hat{A}_i$ is the *ith guaranteed choice domain* of $\mathcal{R}$ meaning that the choices in $\hat{A}_i$ are always available to the corresponding party and some other choices may be available. The intuition is that this defines the choices that we are interested in at a current abstraction level and, therefore, the guaranteed choice domain can be seen as a specification of the relevant properties of a system. For example, at some stage of the protocol the party could always have the possibility to make choices $\{0, 1\}$ but depending on other parties actions it could either have additional choices 3 or 4, giving the $\hat{A} = \{0, 1\}$ and $A = \{0, 1, 3\}$ or $A = \{0, 1, 4\}$.

A setting can be generalised by a more abstract setting where all the properties that hold for the abstract one must also hold for the concrete setting.

**Definition 11** (Abstraction)**.** Let $\mathcal{R}$ and $\mathcal{S}$ be $n$-specifications with guaranteed choice spaces $(\hat{A}_1, \ldots, \hat{A}_n)$ and $(\hat{B}_1, \ldots, \hat{B}_n)$. Let $\pi = (\pi_1, \ldots, \pi_n)$ be a function where $\pi_i : \hat{B}_i \to \hat{A}_i$. We say that $\mathcal{S}$ is a $\pi$ abstraction of $\mathcal{R}$ denoted by

$$\mathcal{R} \sqsubseteq^{\pi} \mathcal{S}$$

if for every $R \in \mathcal{R}$ there exists $S \in \mathcal{S}$ and relation $\rho = \rho_1 \times \ldots \times \rho_n$ such that $R \overset{\rho}{\simeq} S$ and $\pi_i^{-1} \subseteq \rho_i$.

The main idea is that if a party can make a specific guaranteed choice $b_i$ in the abstract setting then there is an equivalent choice $a_i = \pi_i(b_i)$ available in the concrete setting. Therefore, if we analyse the abstract setting and fix a very good choice then we can always make the equivalent choice in all exact versions of the concrete setting. In other words, the concrete setting has all the relevant properties of the abstract setting.

Abstraction is trivially a central concept of abstract cryptography because this relation allows us to show that one system is a more concrete version of a previous abstract concept and that all the results about the abstract object also hold for the new system.

The $n$-choice specifications together with a parallel composition form a *component set* and the abstraction function $\sqsubseteq^\pi$ is the corresponding reduction. Hence, each specification is a component and each $\pi$ will be a constructor. Therefore, the previous composition definitions can be used for specifications and the abstraction of a choice setting is a generally composable reduction (construction). In total, the abstraction $\sqsubseteq^\pi$ is a generally composable reduction if we consider the operations as follows.

**Definition 12** (Parallel composition of specifications)**.** Let $\mathcal{R}$ and $\mathcal{S}$ be two $n$-choice specifications. Then

$$\mathcal{R}||\mathcal{S} = \{R \times S : R \in \mathcal{R}, S \in \mathcal{S}\}$$

where

$$(R \times S)((a_1, b_1), \ldots, (a_n, b_n)) = R(a_1, \ldots, a_n)||S(b_1, \ldots, b_n) \ ,$$

if we assume that the effect space $\Lambda$ has an embedding $|| : \Lambda \times \Lambda \to \Lambda$.

Currently, there are no conditions for the embedding $|| : \Lambda \times \Lambda \to \Lambda$ other than it has to exist. Later we use the case where the effect is actually a new choice setting.

**Definition 13** (Parallel and sequential composition of the constructor set)**.** Let $\pi = (\pi_1, \ldots, \pi_n)$ and $\pi' = (\pi'_1, \ldots, \pi'_n)$ be two constructors. We define the compositions as operations on functions. Hence, serial composition is

$$\pi \circ \pi' = ((\pi_1 \circ \pi'_1), \ldots, (\pi_n \circ \pi'_n))$$

and the parallel composition is

$$\pi|\pi' = (\pi_1 \times \pi'_1, \ldots, \pi_n \times \pi'_n) \ .$$

The neutral element is defined as $\mathsf{id} = (\mathsf{id}, \ldots, \mathsf{id})$.

### 3.1.3 Systems

A system in the highest level of abstraction is an object with interfaces. The systems can be composed by connecting their interfaces and the composition is called *composition-order independent* if the order in which the systems are composed does not matter.

For example, resources, converters and distinguishers are systems which are interesting because they allow to model all systems needed for cryptography. However we can consider additional systems like games and adversaries.

*Resources* are something that provide a functionality, for example an insecure communication channel or an authenticated communication channel. A resource is a system with several interfaces, where one interface is commonly intended for one user. A *converter* is a system that transforms a resource. It can be thought of as an adapter that changes one interface of a resource to something else. The converter is always a system with two interfaces, one that connects to the resource and the other that connects to the user of the resource. For example, we can transform the network channel to an authenticated network channel by adding converters that compute and check authentication codes. An important example of a converter is a simulator, more specifically *local simulators* that are discussed later. An interesting implication of these definitions is that the resources and converters are the same in the two-party setting.

The third class of systems, *distinguishers*, are systems that connect to all interfaces of another system and have a one-bit output interface. They model the common cryptographic distinguishers or, for example, adversaries who have to distinguish between different setups. Distinguishers are used as a metric to measure the the similarity or dissimilarity of the resources. We say that systems are $\varepsilon$-similar for some class $\mathcal{D}$ of distinguishers if no distinguisher $D$ in this class has an advantage more than $\varepsilon$. Secure constructions are defined with respect to the indistinguishability of the construction and an ideal resource.

### 3.1.4 Cryptographic algebras

The previous concepts are formalised by the notion of *cryptographic algebras*.

**Definition 14** (Cryptographic algebra)**.** A cryptographic algebra $\langle \Phi, \Sigma, \approx \rangle$ for a set of interfaces $\mathcal{I}$ consists of a set of resources $\Phi$, a parallel composition operation $||$ and equivalence relation $\approx$ for the resources, a set of converters $\Sigma$, and a mapping from a converter, resource and an interface to a new resource $\Sigma \times \Phi \times \mathcal{I} \to \Phi$. The mapping is specified in the following.

The set of interfaces can be seen as a central concept here. In a sense, the interface set defines the set of intended users and adversaries in the system because each of them has an interface that it can use to connect to the system. For example, an interface set could contain three elements: an interface for user Alice, an interface for user Bob and an interface for the eavesdropper Eve.

**Definition 15** (Cryptographic algebra mapping)**.** The mapping $\Sigma \times \Phi \times \mathcal{I} \to \Phi$ means that we attach a converter to the specified interface of the resource. For a resource $R \in \Phi$, converter $\alpha \in \Sigma$ and interface $i \in \mathcal{I}$ the notation

$$\alpha^i R$$

means that the converter $\alpha$ is attached to interface $i$ of system $R$.

Here attaching no converter is the same as attaching a identity converter $\mathsf{id}$ and converters at different interfaces commute

$$\alpha^i \beta^j R = \beta^j \alpha^i R$$

(i.e. the time at which they are attached does not matter if they are at a different interface). In addition, similarity of the systems implies that adding the same converter does not make them more different

$$R \approx S \;\;\Rightarrow\;\; \alpha^i R \approx \alpha^i S$$

as well as adding an new resource in parallel

$$R \approx S \;\;\Rightarrow\;\; (R||T) \approx (S||T) \;\;\wedge\;\; (T||R) \approx (T||S) \;\; .$$

In addition, we only compose those resources in parallel that have the same set of interfaces.

**Definition 16** (Serial and parallel composition of the converters)**.** Let $\alpha$ and $\beta$ be two converters attached to the same interface of a system. Serial composition $\alpha\beta$ or $\alpha \circ \beta$ is defined as

$$(\alpha\beta)^i R = \alpha^i \beta^i R \;\; .$$

The serial composition is associative because function composition is associative. In addition $\mathsf{id} \circ \alpha = \alpha \circ \mathsf{id} = \alpha$.

Parallel composition is defined by

$$(\alpha||\beta)^i (R||S) = \alpha^i R || \beta^i S \;\; .$$

Differently from the serial composition $\mathsf{id}||\alpha \neq \alpha$. In addition, the composition is not well defined for $(\alpha||\beta)^i T$ if $T \neq R||S$.

**Definition 17** (Distinguisher)**.** A distinguisher $D$ is a mapping from the set of resources $\Phi$ to binary distributions

$$D : R \mapsto DR \;\; .$$

A distinguisher $D$ emulating a converter $\alpha$ induces a new distinguisher

$$(D\alpha^i) : R \mapsto D(\alpha^i R) \;\; .$$

Analogously, for a distinguisher emulating a resource

$$D[\cdot||S] : R \mapsto D(R||S) \;\; .$$

**Definition 18** (Distance between resource systems)**.** We say that the distance of resources $R$ and $S$ is $\varepsilon$ for some class $\mathcal{D}$ of distinguishers, denoted by $R \approx_\varepsilon S$ if $\Delta^{\mathcal{D}}(R,S) \leq \varepsilon$ where $\Delta^{\mathcal{D}}(R,S)$ is the maximal among the distinguishing advantages

$$\Delta^D(R,S) = |\Pr(DR = 1) - \Pr(DS = 1)|$$

of the distinguishers $D \in \mathcal{D}$ where $\Pr(DR = 1)$ marks the probability that the distinguisher outputs 1 when connecting to $R$.

We often only consider the cases of $R \approx_0 S$ where we write $R \approx S$, meaning that these resources are perfectly indistinguishable for the the class of distinguishers that we are interested in. We use $\Delta^{\mathcal{D}}(R, S)$ to denote the maximal advantage that a distinguisher $D \in \mathcal{D}$ has between distinguishing systems $R$ and $S$. The advantages of each distinguisher $D$ satisfy the triangle inequality as

$$\Delta^D(R, S) \leq \Delta^D(R, T) + \Delta^D(T, S) \ .$$

**Definition 19** (Compatibility with a cryptographic algebra)**.** A distinguisher class $\mathcal{D}$ is compatible with a cryptographic algebra $\langle \Phi, \Sigma, \approx \rangle$ if the following conditions hold. For all $D \in \mathcal{D}$

$$R \approx S \ \Rightarrow \ DR = DS \ .$$

In previous terms we have that $R \approx S$ implies $\Delta^{\mathcal{D}}(R, S) = 0$. Moreover, attaching a converter $\alpha \in \Sigma$ or another resource $T \in \Phi$ in parallel induces a distinguisher $D \in \mathcal{D}$ in the same class.

A cryptographic algebra using a class $\mathcal{D}$ of distinguishers instead of the equivalence relation is denoted by $\langle \Phi, \Sigma, \mathcal{D} \rangle$. The distinguisher class has to be compatible with the initial cryptographic algebra.

### 3.1.5 Putting the notions together

Resources are actually a concrete example of choice settings where the choices are converters. Hence, choice settings are an abstraction of the resources. The corresponding effect is the resulting resource that we obtain by applying the converters to the initial resource. Hence, we can also use the general notion of composition for the resources where the abstraction relation $\sqsubseteq$ is the corresponding reduction or constructor. In addition, we can describe equivalent resources as isomorphic choice settings.

If a resource is a choice setting then we can also consider the resource specifications. The idea is that the resource specifications are collections of resources that offer somehow similar functionality. In the following this functionality is defined by the possible converters that can be applied to the resource. Mainly we consider filters that specify that some access is not allowed to the resource for honest participants.

**Definition 20** (Full and guaranteed choice space)**.** For each interface $i$ there exists a *full* choice space $\Sigma$ (set of all possible converters) and a *guaranteed* choice space

$$\Sigma\phi_i = \{\alpha\phi_i : \alpha \in \Sigma\}$$

for some filter $\phi_i \in \Sigma$ such that for each setting in the specification and for each choice space $A_i$ we have $\Sigma\phi_i \subseteq A_i \subseteq \Sigma$.

As the names indicate, the guaranteed choice space is something that has to be available for each resource in the specification. On the other hand, full choice space contains all possible means of accessing the resource.

**Definition 21** (Filtered specifications)**.** For converters $\phi_i$ let $\phi = (\phi_1, \dots, \phi_n)$, then filtered specification $\mathcal{R}_\phi$ denotes the $n$-specification as a set of all resources $R$ where the choice spaces (which are converter spaces) $A_i$ follow the restriction $\Sigma\phi_i \subseteq A_i \subseteq \Sigma$.

Here, $\phi$ works as a filter that restricts the access to a resource $R$, moreover, $\phi_i$ is a filter that restricts access to the interface $i$. Guaranteed choices $\Sigma\phi_i$ are those that are allowed by the filter and every user can always perform them in this system. Not guaranteed choices only become available if a party removes the filter $\phi_i$ and choices in the full set that are not guaranteed can be thought of as an adversarial behaviour. Hence, the space $A_i$ actually contains the set $\Sigma\phi_i$ of honest parties possible actions and some set of adversarial actions. Therefore, specifying $A_i$ actually describes potential adversarial behaviour as well as the parties who can be considered corrupted.

For example, the specification could be a set of resources that correspond to secret keys. Different resources here correspond to different key generation protocols. We can assume that the filtered access to a party gives it a possibility to learn the value of the key. Therefore, learning the key is a guaranteed choice. However, some key generation protocols might be such that a malicious participant can affect the generation so that the outcome is not uniform. Therefore, the full choice space of the key generation resources would also allow for unfiltered access that influences the value of the generated key. It could be that we want to say this set of key generation resources is abstracted by a resource that generates a uniformly random string. For this abstraction relation we only consider the filtered access to the specification.

Finally, the abstraction of the choice settings is defined through a mapping on the guaranteed choice spaces. Hence, for an abstraction of systems we require a mapping on the filtered specifications.

**Definition 22** (Abstraction of filtered specifications)**.** For $\mathcal{R}_\phi$ and $\mathcal{S}_\psi$ we define a mapping $\Sigma\psi_i \to \Sigma\phi_i$ as $\alpha\psi_i \mapsto \alpha\pi_i\phi_i$. Let $\pi = (\pi_1, \dots, \pi_n)$ and denote

$$\mathcal{R}_\phi \sqsubseteq^\pi \mathcal{S}_\psi$$

if the following condition holds. For every $R' \in \mathcal{R}_\phi$ there exists $S' \in \mathcal{S}_\psi$ and a relation $\rho = \rho_1 \times \dots \rho_n$ between the converter spaces $A_1 \times \dots \times A_n$ of $R'$ and $B_1 \times \dots \times B_n$ of $S'$ where $A_i$ and $B_i$ contain converters for interface $i$ such that

$$R' \overset{\rho}{\simeq} S' \text{ and } \{(\gamma\pi_i\phi_i, \gamma\psi_i) : \gamma \in \Sigma\} \subseteq \rho_i \ .$$

In words we say that each $R'$ is isomorphic to $S'$ with relation $\rho$.

This is analogous to Definition 11 where we considered the isomorphism of the choices. Here actually $\gamma$ represents the choice that the user makes, but the choice that is effective in the choice setting $\mathcal{R}$ or $\mathcal{S}$ is changed by the protocol and the filters. Therefore, the final choices are actually $\gamma\pi_i\phi_i$ and $\gamma\psi_i$ respectively. The other way to think about this is that $\pi_i\phi_i$ somehow the same as $\psi_i$. Hence, analogously to the basic choice setting abstraction $\pi$ plays a role in specifying the isomorphism for the guaranteed choice spaces because $\rho$ always has to contain these subsets. The isomorphisms for different $R$ and $S$ may differ for the part defined for the not guaranteed choices.

The main intuition is that if for each setting in $\mathcal{R}_\phi$ there exists a setting in $\mathcal{S}_\psi$ where the choice spaces are isomorphic then by definition the choices result in the same effect. Hence, all the guaranteed effects of $\mathcal{R}_\phi$ are achievable in $\mathcal{S}_\psi$ and $\mathcal{S}_\psi$ is an abstraction of $\mathcal{R}_\phi$.

From the previous result we know that abstraction is a generally composable relation, hence, resources are generally composable.

## 3.2 Main results of abstract cryptography

The first important fact from the previous introduction is that abstraction $\sqsubseteq$ is a universally composable reduction. Hence, also the abstraction of resources is generally composable. Therefore, if we build our protocols using the composition of resources and changing them using the converters then we get a generally composable system.

The main result of abstract cryptography is the fact that abstraction relation can be showed using only local ongoing simulations. Ongoing simulator is such that it translates between two interfaces in an online manner by going through messages sequentially. This opposes to a simulation of the view where we could prepare the messages in any order independently of their occurrence in the protocol.

The main theorem of abstract cryptography is stated as follows.

**Theorem 1.** *Let $\langle \Phi, \Sigma, \approx \rangle$ be a cryptographic algebra and for any interface $i \in \mathcal{I}$ let $\phi_i, \psi_i, \pi_i, \sigma_i \in \Sigma$ be converters. Then*

$$\forall R \in \mathcal{R}\ \exists S \in \mathcal{S}\ \forall \mathcal{P} \subseteq \mathcal{I},\ :\ \pi_\mathcal{P}\phi_\mathcal{P}R \approx \sigma_{\overline{\mathcal{P}}}\psi_\mathcal{P}S \ \Rightarrow\ \mathcal{R}_\phi \sqsubseteq^\pi \mathcal{S}_\psi\ .$$

The precondition means that for a fixed subset $\mathcal{P}$ of the interfaces we apply the specified filters $\phi$ and $\psi$ and the abstraction relation $\pi$. These are all also used in the definition of the abstraction as either the filters $\phi$ and $\psi$ of the abstraction function $\pi$. Actually $\pi$ can be thought of as the protocol converter because this corresponds to the protocol that constructs $S$ from $R$.

For the other interfaces $\overline{\mathcal{P}}$ we apply a converter $\sigma$ to the unfiltered interface on the side of $S$. This $\sigma$ can be thought of as a simulator that makes the unfiltered interface of $S$ indistinguishable from the equivalent unfiltered interface of $R$. However, each corresponding $\sigma_i$ is a local simulator that makes one interface of $S$ indistinguishable from the interface of $R$. Here local means that it is a simulator for exactly one party and not a collection of (colliding) parties as usual for jointly corrupt parties.

# 4 Applying abstract cryptography

This section gives more motivation to the abstract cryptography framework as well as some connection between this and some previously known results. This section both summarises facts from the theory and adds some new comments as well as points out the limitations of the current state of the theory.

## 4.1 Two party case

Some intuition for the general abstraction theorem (Theorem 1) can be found in a specific case for two party setting where the sets $\mathcal{R} = \{R\}$ and $\mathcal{S} = \{S\}$ only contain a single resource. For simplicity also assume that the access is not filtered or that the filter is the same as an identity converter. In this case we can write out all the sets $\mathcal{P}$ in the theorem statement for the interface set $\mathcal{I} = \{1, 2\}$ as follows.

$$\begin{aligned} \pi_1\pi_2 R &\approx S \\ \pi_1 R &\approx \sigma_2 S \\ \pi_2 R &\approx \sigma_1 S \\ R &\approx \sigma_1\sigma_2 S \ . \end{aligned}$$

By definition, if these hold then $R \sqsubseteq^{\pi} S$. However, the two party case is special as we can also say that $R \sqsubseteq^{\pi} S$ implies these four conditions, therefore they are equivalent [MR11]. In addition, if we consider the universal composability [Can01] for the two party case then we only require the first three of the equivalences to conclude that the construction is as secure as $S$. The last equivalence is not used in universal composability because that framework does not consider the case where all parties are corrupted. However, it is important for abstract cryptography, especially in case these parties are corrupted independently. Hence, we can say that in this case abstraction is more general than universal composability.

The fact that we have an equivalence between $R \sqsubseteq^{\pi} S$ and these four equations gives rise to interesting impossibility results for the two party case. We need the equivalence because there results actually use the direction that the relation $R \sqsubseteq^{\pi} S$ implies the four conditions.

These results are centred around a plain communication channel $C$ which is like an identity resource, meaning that $\alpha^1\beta^2 C = \alpha\beta$. In addition, as mentioned earlier, in the two party setting each resource can be considered a converter because it has exactly two interfaces.

The main result for the two party case is that if $S\gamma S \not\approx S$ for all converters $\gamma \in \Sigma$ then there does not exist a protocol $\pi = (\pi_1, \pi_2)$ such that $C \sqsubseteq^{\pi} S$. Meaning that the resource $S$ can not be constructed from only the plain network channel. The two main implications of this theorem are that neither composable commitments nor delayed communication channels can be constructed from just the plain network channel. For the corresponding proofs see [MR11].

As a side note, the article [MR11] also claims the following. *An unleakable (uncoercible) communication channel can not be constructed from a secret key and an authenticated network channel.* However, this is shown for the Alice-Bob-Eve setting that we will only discuss for the constructive cryptography.

## 4.2 Three or more parties and local simulators

Firstly, we would like to have some connection between more general abstract cryptography and, for example, universal composability cases. However, in more general cases we can not easily compare UC and abstract cryptography frameworks because in such case UC only relies on the monolithic simulators. By a monolithic or a central simulator we mean that one simulator simulates the view of all corrupted parties, this is the commonly used approach. In a local simulator each simulator only simulates the view of one party. In a multi-party setting we can see a bigger difference between local and central adversaries and that in common settings we need monolithic simulators.

Abstract cryptography does not commonly consider any adversary, but just parties who may but do not have to follow the protocol. This works well for set-ups where there are different goals for the parties, but they are not corrupted by the same adversary. The case where a central adversary is not enough to model the setup was first introduced in [ASV08]. Abstract cryptography was the first general framework to start using this idea, but, for example, local adversaries are now also studied in the universal composability framework [CV12].

The idea of the local simulators goes hand in hand with the idea of local adversaries because a local simulator is sufficient to show simulatability for a local adversary. Therefore, the abstraction theorem has a lot of sense for the local adversaries case. On the other hand, we commonly need monolithic simulators for colluding adversaries because their views of the protocol have some joint element.

In case of a multi-party setting with collusion the abstraction theorem Theorem 1 is not very strong, as not all protocols are locally simulatable and still require monolithic simulators if parties are corrupted and collaborate. Hence, local simulator condition is possibly too strong and we would require other means of proving abstraction. Currently, the theory of abstract cryptography does not have good examples of cases other than two party case discussed before or a limited three party setting with Alice-Bob-Eve. Therefore, it is not very clear what is the power of the main theorem.

A hint about handling this case is given later for static corruption in constructive cryptography in Section 5.4.1. The idea is that we only consider non-colluding parties and model the collusion via special resources where we can still apply local simulators because the corrupted interfaces are joined into one interface for the adversary. Therefore, it will be the local simulator for given adversary, however, we really just hide the monolithic simulator under the name of local in this case.

## 4.3 Computational models

The common models for security are information theoretic, statistical or computational that correspond to different abilities of the adversary. The adversary can either be unbounded or restricted to doing feasible or efficient computations. Honest parties are traditionally limited to efficient or feasible. Moreover, feasible and efficient are usually synonymous, both meaning polynomial time. In abstract cryptography these two notions are also separated, but not connected to any specific complexity class, like polynomial.

Systems are implemented by algorithms that in turn are described by being either *efficient* or *feasible*. All efficient systems are also feasible, but not vice versa. Feasibility is defined in the terms of feasible resources $\Phi^f$, converters $\Sigma^f$ and distinguishers $\mathcal{D}^f$. Efficient notion is defined by the efficient converters $\Sigma^e$.

**Definition 23** (Closed feasibility notion)**.** A feasibility notion is said to be closed if $\langle \Phi^f, \Sigma^f, \approx \rangle$ is a cryptographic algebra and the distinguisher class $\mathcal{D}^f$ is compatible with this algebra.

**Definition 24** (Compatibility of efficiency and feasibility)**.** An efficiency notion is said to be compatible with a feasibility notion if $\Sigma^e \subseteq \Sigma^f$ and it is closed under composition $\Sigma^e \circ \Sigma^e \subseteq \Sigma^e$.

We can define the classes for information theoretic and computational model as cryptographic algebras. For example, computational security can be described as an algebra of feasible systems $(\Phi^f, \Sigma^f, \mathcal{D}^f)$ where $\pi_i \in \Sigma^e$ are the protocol converters. Which means that the adversary has to perform feasible computations, but the participants have to be efficient.

The information theoretic setting can be fixed as an unbounded cryptographic algebra $(\Phi^u, \Sigma^u, \mathcal{D}^u)$. However, in this case also the protocols that we use can be unbounded. Therefore, analogously to the computational security we can consider efficient information theoretic security with protocol converters $\pi_i \in \Sigma^e$. Efficient information theoretic security implies computational security.

What exactly we mean by efficient of feasible can be determined at a lower level. Depending on the context we can, for example, either use polynomial or logarithmic time as a meaning of an efficient algorithm.

## 4.4 Additions to the general framework

The initial framework of abstract cryptography has been extended to allow for more fine grain modelling of the resources and converters [DGHM13]. Especially, it concentrates on the memory usage. For example, stateless converters that have no memory between queries from the outer interface that connects to the user. In addition, we could model an arbitrary bound on the memory by a memory resource that has that amount of memory available. To quantify a memory usage of a simulator they use a stateless converter and the corresponding memory resource. Hence, we can define *memory-aware reducibility* where a system can be securely constructed from the other when the adversary has only limited amount of memory. The specifics of this approach are connected with the indifferentiability framework [MRH04] that is not discussed as part of this work. However, an important implication of this addition is that abstract cryptography can be used also by specifying more restricted requirements than just efficiency.

# 5 Constructive cryptography

In a sense, all of the previous is independent of cryptography and could be used for any systems. Constructive cryptography gives it more meaning in the cryptology, especially for security definitions and secure constructions.

The main difference from the abstract cryptography is that in practice we usually consider only certain classes of adversarial behaviour and we are interested in secure construction with respect to these classes. On the other hand, the abstraction relation required indistinguishability for all cases. Still, constructive cryptography uses the abstraction ideas because it does not build all resources from low level details but uses the high level systems. Constructive cryptography was laid out in [Mau10, Mau11] and has been used to show security for confidentiality and integrity of symmetric key encryption [MRT12], anonymity preserving public key encryption [KMO$^+$13] and key agreement [MTC13]. This section does not go into details about all aspects of constructive cryptography, but rather tries to give an overview of some aspects that are of interest either to get more intuition about constructive and abstract cryptography or to point out some open questions or difficulties in using these theories.

Actually, the main idea that cryptographic schemes can be described in terms of transformations of the network channel originates from [MS94, MS96]. This can be seen as a starting point for constructive cryptography.

## 5.1 Security definitions

In the UC [Can01] and RSIM [PW00, PW01] the security is defined in a *ideal world versus real world* comparison. Nonetheless, these concepts are used more for cryptographic protocols and especially in multi-party computation. On the other hand, for cryptographic primitives we usually have several security goals and definitions and we use game based approach to prove their security. Nevertheless, these proofs do not give guarantees about composability which we would actually require.

The main idea of constructive cryptography is to give all security definitions in terms of *ideal resource* that a given *real resource* satisfies. Especially, constructive cryptography so far [Mau11, MRT12, KMO$^+$13, MTC13] targets cryptographic primitives. The real resource in turn is described by some other resources and the converters applied to the initial resources. This definition makes sure that the requirements of the initial resources, that we need in the construction, correspond to all the preconditions, and the ideal resource captures all properties achieved by the construction. Actually, in previous terms, we want to show that the ideal resource is an abstraction of the real resource, but the difference is that in this case we are working with one exact resource and not a class of resources. Hence, we really show that the ideal and real resources behave similarly or, in more common terms, that they are indistinguishable for some class of distinguishers. Therefore, constructive cryptography can be seen as a meeting point for the abstract cryptography framework and classical cryptography security definitions.

The approach taken by the constructive cryptography has an interesting difference compared to UC or RSIM framework ideal functionalities. It is common to state things like the ideal functionality of a commitment protocol. However, in constructive cryptography, the ideal resource is not defined for the protocol, but for the construction, where the protocol is applied on another resource. Therefore, there is no ideal resource for the commitment, but an ideal resource corresponding to the commitment protocol used, for example, over an authenticated channel or plain network channel.

In a traditional setting, the security of a scheme is usually defined by an adversary in a security game. The scheme is considered secure if no feasible adversary wins the game with non-negligible probability. However, in many cases we have different security goals and hence, different games for one scheme. The specific version of the scheme that is used is chosen with respect to what is supposed to be achieved by using this scheme. This gives a natural rise to the constructive cryptography. Hence, instead of thinking alternatingly between attack scenarios and desired functionality we only focus on the functionality that the scheme offers. The notion of an adversary does not occur in the constructive cryptography approach. However, game-based definitions can be explained and reimplemented in constructive cryptography [Mau11] and even adapted, for example, see [MRT12, KMO$^+$13] for the usage of games.

In addition, by defining the construction we also define the context. Therefore there are no *stand-alone* protocols in this world and in a sense we obtain composability by design.

## 5.2 Common setting with Alice, Bob and Eve

The parties in the system would be the intended users and also network eavesdroppers. The parties can either behave honestly or dishonestly. On the other hand, the eavesdroppers could, but do not have to be present. The users case is used as defined in the main theorem by either using a protocol converter that is defined by the abstraction or in case they are corrupted, we use the simulator $\sigma$. However, for eavesdroppers we use the simulator $\sigma$ if they are present and the corresponding protocol converter is $\perp$ denoting that this party is not present and the interface is blocked.

Consider a common case of honest Alice and Bob and a network eavesdropper Eve. This is a common setting for many cryptographic primitives, for example, symmetric encryption and message authentication codes. Alice and Bob are honest parties and Eve is an intruder in the network.

**Definition 25** (Secure construction for Alice, Bob and Eve)**.** We say that a protocol $\pi = (\pi_1, \pi_2)$ securely constructs resource $S$ from resource $R$ within $\varepsilon$ denoted by

$$R \xrightarrow{\pi,\varepsilon} S$$

if we have availability

$$\Delta^{\mathcal{D}}(\pi_1^A \pi_2^B \perp^E R, \perp^E S) \leq \varepsilon$$

meaning that the construction is secure if no eavesdroppers are present and we have security in presence of the eavesdroppers as

$$\exists \sigma \in \Sigma : \Delta^{\mathcal{D}}(\pi_1^A \pi_2^B R, \sigma^E S) \leq \varepsilon \ .$$

Where we have a cryptographic algebra $\langle \Phi, \Sigma, \approx \rangle$ and protocol interfaces $\mathcal{I} = \{A, B, E\}$ respectively for Alice, Bob and Eve. In addition we have a distinguisher class $\mathcal{D}$.

If the distinguisher class $\mathcal{D}$ is compatible with the cryptographic algebra then the secure construction is generally composable. This is a more concrete result than the general composability of the abstraction. The main implication of composability of the construction is that constructive security definitions are also further composable. Therefore we can use step-wise refinement also here to build larger functionalities from known blocks so that the final construction remains secure.

This is actually a restriction of the Theorem 1 to this specific case where we consider the specifications that only have one resource in them. By the general definition we should consider all $2^3$ subsets of the interface set, but they reduce to the two analysed cases because of the restricted behaviour of the parties.

## 5.3 One time pad example

This section considers an example of constructive cryptography for the Alice, Bob and Eve setting. For this we use the one time pad (OTP) example from [Mau11]. In OTP the ciphertext $c$ is defined by $k \oplus m$ where $m$ is the message and $k$ is a secret key. In addition, the bit-length of $m$ is the same as the bit-length of $k$.

The OTP is well known for being an unconditionally secure cryptosystem where we can prove that the ciphertext and the plaintext are statistically independent. However, for most practical purposes OTP is not usable. The first problem is that the adversary can modify the ciphertext and knows how this is going to change the plaintext. For example, the adversary can pick a bit-string $a$ and modify the ciphertext as $a \oplus c$ which means that after decryption, the plaintext becomes $m \oplus a$ instead of $m$. Secondly, the ciphertext in this case actually leaks the message length. We can overcome the first issue by using an authenticated channel, but we always have to take into account that the ciphertext leaks something about the length of the plaintext.

These limitations can of course be taken into account when using OTP, but for discovering them we need to analyse the behaviour of OTP. The idea of constructive cryptography security definitions would is that such limitations should be directly pointed out in the security definition. Hence, from the definition, it should be clear what the proposed scheme offers and under which conditions. In a sense, the definition, given by the ideal functionality, should be all that we actually know about the system for using it.

The ideal functionality that we aim at is a secure channel $\mathsf{SEC}$ for sending one message from $A$ to $B$ that leaks the length of the message to an eavesdropper $E$. Note that it is defined for sending just one message because we can not reuse the key for one-time pad. The ideal resource and a simulator are illustrated by Figure 1.

The real resource consists of an authenticated network channel and a secret key shared by $A$ and $B$ in parallel, giving $\mathsf{AUT}||\mathsf{KEY}$. Here they both have an interface for $A$, $B$ and $E$, but the interface to $E$ is blocked for $\mathsf{KEY}$. The OTP protocol is defined by two converters: encryption $\mathsf{otp\text{-}enc}$ and decryption $\mathsf{otp\text{-}dec}$. The real resource is illustrated

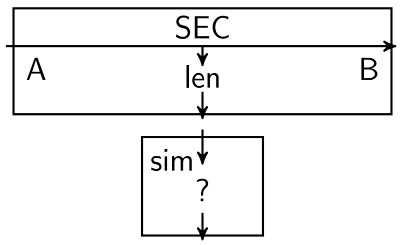

Figure 1: The ideal resource for a secure network channel and the simulator to make it equivalent with the real construction. For the availability case we would have $\perp$ instead of the simulator.

by Figure 2. If we consider the parties' names as interfaces, then we have

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

which is just a shorter way to denote the construction from Figure 2.

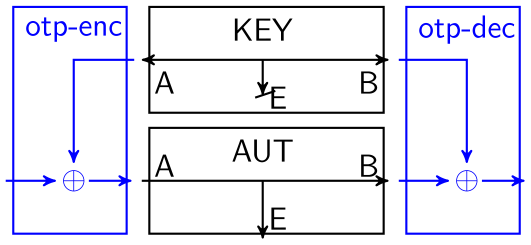

Figure 2: The real resource for OTP from the one time pad, an authenticated network channel and a secret key.

The corresponding security definition becomes

$$\mathsf{otp\text{-}dec}^B \mathsf{otp\text{-}enc}^A (\mathsf{AUT}||\mathsf{KEY}) = \mathsf{sim}^E \mathsf{SEC} \ .$$

To prove the security we in turn have to find a suitable converter sim. For OTP this is easy, as sim gets the message length as an input from SEC and generates a random bit-string of this length for $E$. This output, as well as the eavesdroppers output in the real resource, are both uniformly random and of the same length, therefore indistinguishable.

Secondly, we are also interested in the availability property of the OTP, which holds, as we clearly have

$$\mathsf{otp\text{-}dec}^B \mathsf{otp\text{-}enc}^A \perp^E (\mathsf{AUT}||\mathsf{KEY}) = \perp^E \mathsf{SEC} \ .$$

Although we have the condition that SEC allows to send only one message we should also keep in mind that the construction is secure as long as we only send one message on AUT.

## 5.4 General cases of constructive cryptography

At current state, the constructive cryptography is used for showing the security of only cryptographic primitives and not more general protocols. The focus for primitives is important because the ideal world specification based approaches are anyway used in the cryptographic protocol design. In addition, primitives generally involve a fixed set of parties, whereas the Alice-Bob-Eve setting is sufficient for most cases. However, this section raises some questions about the applicability of constructive cryptography in the case of more general sets of participants.

We can consider different security definitions for general multi-party protocols. The differences contain the classical aspects as, for example, static and adaptive corruption, passive or active adversaries. What we have to show to prove the security of a construction is to consider all combinations of corrupted parties and show that for all of them the ideal and real resource behave the same.

Secure general construction can be seen as a special case of abstraction where $\mathcal{R} = \{R\}$ and $\mathcal{S} = \{S\}$ and the existence of this construction $\pi$ implies $\mathcal{R} \sqsubseteq^{\pi} \mathcal{S}$. However, as we already saw for the Alice-Bob-Eve setting there are cases where we do not need to consider all the possibilities. The main trouble with defining different classes also means separate definitions of secure construction. Finally, it means that each of these constructions has to be proven composable and the gain from abstractive cryptography ideas is not clear.

For example, it seems that the secure construction definition of Alice-Bob-Eve easily generalises to semi-honest security of a multi-party protocol if we add players beside Alice and Bob. However, the main problem is that in case of a central adversary the local simulation may be too strong requirement and we have to have means to look at corrupted parties jointly.

However, currently most of the possible cases have not been explored and it could be that there is a good way to handle different cases in a unified manner. The interesting questions about general constructive cryptography include the modelling of traditional concepts, like central adversary, or building the resources that are more complicated than one network channel. Following two subsections explore the areas of central adversaries and resource construction. It is up to the following works in this field to introduce other cases.

### 5.4.1 Static corruption and central adversary

Static corruption with a central adversary for constructive cryptography is defined analogously to the model used in reactive simulatability[PW00]. This was introduced for constructive cryptography in [KMO$^{+}$13]. This is quite a different case from the general approach so far because so far we have not really considered adversary or the corruption concept. However, static corruption with a central adversary is quite commonly used in cryptography.

For a resource $R$ we define special versions of it depending on which parties are corrupted. The new resources are denoted by $R_{\mathcal{C}}$, where $\mathcal{C}$ is the set of corrupted parties. It can be said that a resource in this setting actually means a class of resources. In addition, the adversary will be like the eavesdropper and connects to the same $E$-interface. The resource $R_{\mathcal{C}}$ is defined so that the interface $i$ of each corrupted party is joined with the eavesdroppers interface $E$. Therefore, the adversary or the former eavesdropper has all the control over these interfaces. However, different approaches on handling the corruption could be considered. For example, when we consider passive corruption, then the adversary still has to follow the protocol. However, differently from the honest parties the adversary can see more inside the protocols than just inputs and outputs. For the active case, the adversary is not limited in its actions.

**Definition 26** (Secure construction for static corruption)**.** The resource $S$ is said to be securely constructed from $R$ by protocol $\pi$ with respect to static corruption if the secure construction definition holds pairwise between $R_{\mathcal{C}}$ and $S_{\mathcal{C}}$ for all sets $\mathcal{C}$ of the corrupted parties.

As a side note, currently the case of adaptive central adversary has not been explored for the constructive cryptography.

### 5.4.2 Network resources

A central concept of constructive cryptography is the resource. Among them we are mostly interested in the resources that allow for communication, therefore, network channels. The basic resource is the plain network channel from which we can build authenticated or secure channels by adding cryptographic keys. However, key as a resource corresponds more to a key generation and sharing than just the key value. These are concepts that are easy to understand, but we need to make them somehow precise to work with them.

However, the difficulty is that we follow the abstract cryptography and we do not want to describe these resources in terms of low level details. The main tool for describing resources in constructive cryptography so far is the secure channel calculus [MS94, MS96]. However, describing network setups for more than two parties, for example as in [KMO$^+$13], requires extending the initial calculus and can become tricky. In addition, a closer look at the anonymous broadcast channel construction in [KMO$^+$13] reveals that we can actually combine resources that do not provide an interface to all participants. It is not necessary to specify the resources in secure channel calculus but currently this is the only used way.

In addition, we usually consider two classes of resources. The real network resource and the ideal desired resource. The previous constructions are used for the real resource. Specifying the ideal resource is an additional challenge when using constructive cryptography. For example, what would be the ideal network resource corresponding to a combination of an authenticated network and a zero-knowledge protocol.

## 6 Conclusion

This report introduced the ideas of abstract and constructive cryptography that both propose interesting ideas for approaching cryptographic security proofs. These ideas are special as they propose the view of local adversaries as opposed to commonly used central adversaries. Current main results in these fields are focused on the cases where it is natural to consider local adversaries, however it currently lacks tools to work with central adversaries.

Both frameworks have been mainly studied for limited cases of either two-party protocols or classical Alice-Bob-Eve setting. It is somewhat unclear how much modifications the initial ideas require in order to use them in a more general setting. Hopefully these directions will be given in the follow up work on these fields. The main open question is the exact usability of the abstraction relation in a general multi-party setting or the benefit of these ideas in the presence of a central adversary.

## References

[ASV08] Joël Alwen, Abhi Shelat, and Ivan Visconti. Collusion-free protocols in the mediated model. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 497–514. Springer, 2008.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. *FOCS '01 Proceedings of the 42nd IEEE symposium on Foundations of Computer Science*, 016(016), 2001.

[CV12] Ran Canetti and Margarita Vald. Universally composable security with local adversaries. In *Proceedings of the 8th international conference on Security and Cryptography for Networks*, SCN'12, pages 281–301, Berlin, Heidelberg, 2012. Springer-Verlag.

[DGHM13] Grgory Demay, Peter Gai, Martin Hirt, and Ueli Maurer. Resource-restricted indifferentiability. In Thomas Johansson and PhongQ. Nguyen, editors, *Advances in Cryptology EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 664–683. Springer Berlin Heidelberg, 2013.

[KMO$^+$13] Markulf Kohlweiss, Ueli Maurer, Cristina Onete, Bjrn Tackmann, and Daniele Venturi. Anonymity-preserving public-key encryption: A constructive approach. Cryptology ePrint Archive, Report 2013/238, 2013.

[KT13] Ralf Kuesters and Max Tuengerthal. The iitm model: a simple and expressive model for universal composability. Cryptology ePrint Archive, Report 2013/025, 2013.

[Mau09] Ueli M. Maurer. Abstraction in cryptography. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, page 465. Springer, 2009.

[Mau10] Ueli Maurer. Constructive cryptography  a primer. In Radu Sion, editor, *Financial Cryptography and Data Security*, volume 6052 of *Lecture Notes in Computer Science*, pages 1–1. Springer Berlin Heidelberg, 2010.

[Mau11] Ueli Maurer. Constructive cryptography – a new paradigm for security definitions and proofs. In S. Moedersheim and C. Palamidessi, editors, *Theory of Security and Applications (TOSCA 2011)*, volume 6993 of *Lecture Notes in Computer Science*, pages 33–56. Springer-Verlag, April 2011.

[MR11] Ueli Maurer and Renato Renner. Abstract cryptography. In Bernard Chazelle, editor, *The Second Symposium in Innovations in Computer Science, ICS 2011*, pages 1–21. Tsinghua University Press, January 2011.

[MRH04] Ueli Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *Theory of Cryptography*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer Berlin Heidelberg, 2004.

[MRT12] Ueli M. Maurer, Andreas Redlinger, and Bjrn Tackmann. Confidentiality and integrity: A constructive perspective. In *Theory of Cryptography*, volume 7194 of *Lecture Notes in Computer Science*, pages 209–229. Springer, 2012.

[MS94] Ueli M. Maurer and Pierre E. Schmid. A calculus for secure channel establishment in open networks. In Dieter Gollmann, editor, *ESORICS*, volume 875 of *Lecture Notes in Computer Science*, pages 175–192. Springer, 1994.

[MS96] Ueli M Maurer and Pierre E Schmid. A calculus for security bootstrapping in distributed systems. *Journal Of Computer Security*, 4:55–80, 1996.

[MTC13] Ueli Maurer, Björn Tackmann, and Sandro Coretti. Key exchange with unilateral authentication: Composable security definition and modular protocol design. Cryptology ePrint Archive, Report 2013/555, 2013.

[PW00] Birgit Pfitzmann and Michael Waidner. Composition and integrity preservation of secure reactive systems. In *Proceedings of the 7th ACM conference on Computer and communications security*, CCS '00, pages 245–254, New York, NY, USA, 2000. ACM.

[PW01] Birgit Pfitzmann and Michael Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, SP '01, pages 184–, Washington, DC, USA, 2001. IEEE Computer Society.