Initiator-Resilient Universally Composable Key Exchange

Dennis Hofheinz, Jörn Müller-Quade, and Rainer Steinwandt

IAKS, Arbeitsgruppe Systemsicherheit, Prof. Dr. Th. Beth,
Fakultät für Informatik, Universität Karlsruhe, 76 131 Karlsruhe, Germany
Email: {hofheinz,muellerq,steinwan}@ira.uka.de

Abstract. Key exchange protocols in the setting of *universal composability* are investigated. First we show that the ideal functionality $\mathcal{F}_{\mathrm{KE}}$ of [9] cannot be realized in the presence of adaptive adversaries, thereby disproving a claim in [9]. We proceed to propose a modification $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$, which is proven to be realizable by two natural protocols for key exchange. Furthermore, sufficient conditions for securely realizing this modified functionality are given. Two notions of key exchange are introduced that allow for security statements even when one party is corrupted. Two natural key exchange protocols are proven to fulfill the "weaker" of these notions, and a construction for deriving protocols that satisfy the "stronger" notion is given.

Keywords: formal cryptography, cryptographic protocols, universal composition, key exchange.

1 Introduction

It is generally agreed upon that providing only non-formal intuitive security statements about cryptographic schemes and protocols is not satisfying. Consequently, models have been developed which try to provide formally satisfying notions of *security* in various settings. The covered topics range from security notions for symmetric and asymmetric encryption schemes, over security concepts for signature schemes to security notions for arbitrary protocols.

We do not try to give a survey of all the work that has been done in this area, but it is worth pointing out that in the cryptographic research community much work can be traced back to a seminal paper of Goldwasser and Micali [16]. As already indicated by the title of the latter, formal approaches in this line of research are often well-suited to model probabilistic aspects of attacks, and attacks which make sophisticated use of the inner structure of messages. Despite some well-known proof methodologies, the typically encountered (reduction) proofs are "hand-made". On the other hand, in the security research community, much focus has been put on the use of term rewriting and formal proof systems. One particularly important model is due to Dolev and Yao [15]. Both the approach of the "crypto camp" and the approach of the "security camp" have clearly led to remarkable results. Unfortunately, at the moment there seems to be a clear gap between these two "camps". In research on protocol security, the situation is quite similar—two different models are used, and both of them have proven to be useful: The model of Canetti [6], e. g., allowed for interesting insights in the limitations of the composability of two-party computations [10], and the approach of Pfitzmann and Waidner [20] led to the development of a *universally compsable* *cryptographic library* [2], for instance. In fact, the latter work can be seen as a very interesting step towards closing the gap between the cryptographic and the security research community. Our contribution is formulated in the model of Canetti and deals with notions for the security of key exchange; it can be seen in the line of the works [4, 3, 21, 8, 9]. So far, we have not explored to what extent our results can be adapted to the setting of Pfitzmann and Waidner, where the problem of key exchange has been explored, e. g., by Steiner [22].

In [6], a very strict notion of security is given which guarantees *universal composability* of protocols. More specifically, that means that given any secure protocol π which utilizes an idealized version $\mathcal{F}$ of a protocol task (called an *ideal functionality*), another protocol τ which in turn securely realizes $\mathcal{F}$ can replace a polynomial number of instances of $\mathcal{F}$ in protocol π without compromising the overall security of π. Key exchange protocols in this setting were studied in [9]. However, as we will show in the following, the security notion of [9] cannot be fulfilled when considering *adaptive* adversaries, which may corrupt participants of the protocol at any time during the protocol execution. In this contribution we will therefore provide a slightly modified specification for key exchange realizable in the presence of adaptive adversaries. Furthermore, two natural key exchange protocols are proven secure in that sense. In fact, we investigate general sufficient conditions for key exchange protocols to be secure with respect to our notion.

In view of universal composability one must not restrict attention to the case where the "initiator" and the "responder" of a key exchange are uncorrupted and need to be protected against an adversary monitoring the communication channel "from the outside". To be able to employ a key exchange protocol within a more complicated protocol context it is necessary to specify the behavior of a key exchange protocol also for the case when the initiator or the responder are corrupted. In [9], in face of a corrupted initiator *or* responder, the adversary may freely choose the key which is to be the outcome of the key exchange protocol. Investigating, e. g., a Diffie-Hellman-like key exchange we observe that it is not obvious how the initiator could, if corrupted, let the adversary freely choose the key agreed upon. This leads to the natural question whether or not some known key exchange protocols may in fact realize something strictly stronger than a universally composable key exchange as described in [9] (resp., in Section 3 below). Specifically, it seems that a Diffie-Hellman-like key exchange is "initiator-resilient" in the sense that a corrupted initiator cannot force the outcome of the key exchange to be some specific key, which could then be known to some third party or be some "weak" key of an encryption functionality to be used after the key exchange.

To make this intuition explicit, we first give a very straightforward and intuitive ideal functionality for initiator-resilient key exchange where even in case of a corrupted initiator, the key agreed upon is chosen at random. It turns out that this ideal functionality can be realized securely, although it might be considered "too restrictive", as two natural and "intuitively initiator-resilient" key exchange protocols can be shown not to realize this ideal functionality. Therefore, we also present a slightly more involved ideal functionality making use of a *non-information oracle*, as defined in [9].

In the new ideal functionalities introduced in this contribution, the adversary still has complete control over the outcome of the key exchange when the *responder* gets corrupted. Yet a close inspection of, e. g., a Diffie-Hellman-like key exchange protocol suggests that there exist key exchange protocols for which the influence each individual party has on the key is limited. It is an interesting open question if this additional property of certain key exchange protocols can be captured in an appropriate ideal functionality.

2 Preliminaries

To start, we shortly outline the framework for multi-party protocols defined in [6]. First of all, *parties* (denoted by P_1 through P_n) are modeled as *interactive Turing machines (ITMs)* (cf. [6]) and are supposed to run some (fixed) protocol π. There also is an *adversary* (denoted $\mathcal{A}$ and modeled as an ITM as well) carrying out attacks on protocol π. Therefore, $\mathcal{A}$ may corrupt parties (in which case it learns the party's current state and the contents of all its tapes, and controls its future actions), and intercept or, when assuming unauthenticated message transfer[1], also fake messages sent between parties. If $\mathcal{A}$ corrupts parties only *before* the actual protocol run of π takes place, $\mathcal{A}$ is called *non-adaptive*, otherwise $\mathcal{A}$ is said to be *adaptive*. The respective local inputs for protocol π are supplied by an *environment machine* (modeled as an ITM and denoted $\mathcal{Z}$), which may also read all outputs locally made by the parties and communicate with the adversary. Here we will only deal with environments guaranteeing a polynomial (in the security parameter) number of total steps all participating ITMs run. For more discussion on this issue, cf. [17].

The model we have just described is called the *real* model of computation. In contrast to this, the *ideal* model of computation is defined just like the real model, with the following exceptions: we have an additional ITM called the *ideal functionality* $\mathcal{F}$ and being able to send messages to and receive messages from the parties privately (i. e., without the adversary being able to even intercept these messages). The ideal functionality may not be corrupted by the adversary, yet may send messages to and receive messages from it. Furthermore, the parties $P_1, \ldots, P_n$ are replaced by *dummy parties* $\tilde{P}_1, \ldots, \tilde{P}_n$ which simply forward their respective inputs to $\mathcal{F}$ and take messages received from $\mathcal{F}$ as output. Finally, the adversary in the ideal model is called the *simulator* and denoted $\mathcal{S}$. The only means of attack the simulator has in the ideal model are those of corrupting parties, delaying or even suppressing messages sent from $\mathcal{F}$ to a party, and all actions that are explicitly specified in $\mathcal{F}$. However, $\mathcal{S}$ has no access to the contents of the messages sent from $\mathcal{F}$ to the dummy parties (except in the case the receiving party is corrupted) nor are there any messages actually sent between (uncorrupted) parties $\mathcal{S}$ could intercept. Intuitively, the ideal model of computation (or, more precisely, the ideal functionality $\mathcal{F}$ itself) should represent what we ideally expect a protocol to do. In fact, for a number of standard tasks, there are formulations as such ideal functionalities (see, e. g., [6]).

[1] In [9], the model for message transfer is called *unauthenticated*, even when each ordered pair (P_i, P_j) of parties is allowed to exchange *one* message in an authenticated manner (i. e., the adversary is unable to fake such a message).

To decide whether or not a given protocol π does what we would ideally expect some ideal functionality $\mathcal{F}$ to do, the framework of [6] uses a *simulatability*-based approach: at a time of its choice, $\mathcal{Z}$ may enter its halt state and leave output on its output tape. The random variable describing the first bit of $\mathcal{Z}$'s output will be denoted by $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)$ when $\mathcal{Z}$ is run on *security parameter* $k \in \mathbb{N}$ and initial input $z \in \{0,1\}^*$ (which may, in case of a non-uniform $\mathcal{Z}$, depend on k) in the real model of computation, and $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)$ when $\mathcal{Z}$ is run in the ideal model. Now if for any adversary $\mathcal{A}$ in the real model, there exists a simulator $\mathcal{S}$ in the ideal model such that for *any* environment $\mathcal{Z}$ and *any* initial input z, we have that

$$|\mathbf{P}(\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)=1) - \mathbf{P}(\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)=1)| \tag{1}$$

is a negligible[2] function in k, then protocol π is said to *securely realize* functionality $\mathcal{F}$.[3] Intuitively, this means that any attack carried out by adversary $\mathcal{A}$ in the real model can also be carried out in the idealized modeling with an ideal functionality by the simulator $\mathcal{S}$ (hence the name), such that no environment is able to tell the difference. Analogously to [6] we restrict to protocols which generate output if all messages are delivered and no party gets corrupted.

Remark 1. In the framework of [6], the above definition of security is equivalent to the seemingly weaker requirement that there is a simulator $\mathcal{S}$ so that (1) is a negligible function in k for any environment $\mathcal{Z}$ and input z, and the special real-model *dummy adversary* $\tilde{\mathcal{A}}$, which follows explicit instructions from $\mathcal{Z}$.

Remark 2. The original modeling of [6] does not involve an explicit message sent to the ideal functionality upon party corruptions. Yet exactly this additional feature proved helpful in later works (e. g., [9, 11]) and in particular allows to formulate key exchange functionalities in a convenient way. This change does not affect the validity of the crucial *composition theorem* proven in [6].

Remark 3. In [6], the environment machine is modeled as a *non-uniform* ITM (i. e., as an ITM having input $z = z(k)$ dependent on the security parameter k). However, as the *composition theorem* of [6] remains valid when restricting to *uniform* environment machines (i. e., those with input not dependent on k, cf. [17]), it makes sense to alternatively consider only uniform environments where appropriate. In particular, all proofs given below hold for both uniform and non-uniform environments; alone the respective assumptions (i. e., the decisional Diffie-Hellman assumption) have to be considered with respect to the uniformity class in question.

3 Key Exchange

Now we are ready to show the ideal functionality $\mathcal{F}_{\text{KE}}$ from [9] (see also Figure 1) to be non-realizable if adversaries are allowed to corrupt adaptively. The key observation in our argument is that in the formulation of [9], the functionality $\mathcal{F}_{\text{KE}}$ determines the common key later handed to both participants right after the

[2] A function $f : \mathbb{N} \to \mathbb{R}$ is called *negligible*, if for any $c \in \mathbb{N}$, there is a $k_0 \in \mathbb{N}$ such that $|f(k)| < k^{-c}$ for all $k > k_0$.

[3] The formulation in [6] is slightly different, but equivalent to the one chosen here which allows to simplify our presentation.

respective initialization messages arrived. As the ideal-model adversary cannot delay or block these initialization messages, this happens right at the start of the protocol. Furthermore, if at this point in time, neither participant is corrupted, the session key is chosen uniformly by $\mathcal{F}_{\mathrm{KE}}$. Yet again, in the real model, at least one of the two participants should be able to influence the session key! (Although this is intuitively clear, showing it in our situation makes up the main part of the proof below.)

Functionality $\mathcal{F}_{\mathrm{KE}}$

$\mathcal{F}_{\mathrm{KE}}$ proceeds as follows, running on security parameter k, with parties $P_1, \ldots, P_n$ and an adversary $\mathcal{S}$.

1. Upon receiving a value `(Establish-session,`sid`,`P_i`,`P_j`,`$role$`)` from some party P_i, record the tuple $(sid, P_i, P_j, role)$ and send this tuple to the adversary. In addition, if there already is a recorded tuple $(sid, P_j, P_i, role')$ (either with $role' \neq role$ or $role' = role$) then proceed as follows:
   (a) If P_i and P_j are uncorrupted then choose $\kappa \overset{R}{\leftarrow} \{0,1\}^k$, send `(key,`$sid$`,`$\kappa$`)` to P_i and P_j, send `(key,`sid`,`P_i`,`P_j`)` to the adversary, and halt.
   (b) If either P_i or P_j is corrupted, then send a message `(Choose-value,`sid`,` P_i`,`P_j`)` to the adversary; receive a value κ from the adversary, send `(key,`sid`,`κ`)` to P_i and P_j, and halt.
2. Upon corruption of either P_i or P_j, proceed as follows. If the session key is not yet sent (i. e., it was not yet written on the outgoing communication tape), then provide $\mathcal{S}$ with the session key. Otherwise provide no information to $\mathcal{S}$.

Fig. 1. The ideal functionality $\mathcal{F}_{\mathrm{KE}}$ from [9].

More specifically, we show that corrupting a party right after it received input and then running the protocol completely through this corrupted party causes the output of the other party to allow for distinguishing real from ideal. In the ideal model, this output is chosen uniformly by $\mathcal{F}_{\mathrm{KE}}$, whereas in the real model, it results from a factually executed key exchange.[4] The proof of Theorem 14 in [9] simply does not take into account such malicious behavior of a party that was corrupted *after* it received its initialization message. This applies in particular to the reduction in the proof of [9, Theorem 14] of an environment $\mathcal{Z}$ (together with an adversary $\mathcal{A}$) to an attacker $\mathcal{A}'$ in the model of SK-security. Generalizing this kind of argument[5] seems possible, thereby excluding the realizability of, e. g., the functionality $\mathcal{F}_{\mathrm{SFE}}$ (as formulated in [6]) for secure function evalution even in the $\mathcal{F}_{\mathrm{CRS}}$-hybrid model.[6] Here, $\mathcal{F}_{\mathrm{CRS}}$ denotes the *common reference string functionality* as used in [11].

[4] Recently we learned that our argument is very similar to the one of [13] presented against the bit commitment functionality $\mathcal{F}_{\mathrm{COM}}$ from [6, 7].

[5] i. e., giving input to an uncorrupted party P_i and then, after that party forwarded its input to the ideal functionality in the ideal model, corrupting P_i and forcing it to take part in a real protocol run with *different* input

[6] After acceptance of this paper, a revised version [12] of [11] was published in the IACR ePrint archive; this revision addresses attacks like this by changing the ideal model. In the changed model, the ideal-model adversary is in charge of delivery of input messages forwarded by the dummy parties to the ideal functionality.

To overcome such problems, we introduce a modified key exchange functionality and prove two common protocols to be secure realizations hereof. In fact, these protocols are very similar to the ones considered in [3] for key exchange. However, our key exchange functionality differs from $\mathcal{F}_{\mathrm{KE}}$ in several aspects: first, the common key may be chosen by the ideal-model adversary if at the *end* of a simulated protocol run, anyone of the participants is corrupted. Moreover, to exclude complications conditional on the order and roles in which parties are asked to perform a key exchange, we define a family $\{\mathcal{F}_{\mathrm{KE}}^{(i,j)}\}_{P_i,P_j}$ of ideal functionalities indexed by the parties involved and thus implicitly fixing the respective roles they take in the key exchange. We remark that there is also a subtlety regarding the distribution from which the common keys are picked. As with $\mathcal{F}_{\mathrm{KE}}$ from [9], we demand random k-bit strings (where k is the security parameter) for keys. On the other hand, the "raw" output resulting from a, say, Diffie-Hellman-like key exchange may be computationally distinguishable from random k-bit strings, even under the decisional Diffie-Hellman assumption. In the case of Diffie-Hellman-like key exchange protocols, we therefore follow the approach in [21, Section 5.2.2] and use a family of pair-wise independent hash functions to pass from random group elements to random bitstrings.

Proposition 1. *Presuming authenticated links and no further set-up assumptions, $\mathcal{F}_{\mathrm{KE}}$ from [9] cannot be securely realized by any two-party protocol π terminating in strict polynomial time if adversarial corruption is adaptive.*

Proof. Assume that π securely realizes $\mathcal{F}_{\mathrm{KE}}$. Let $m(k)$ be a polynomial bounding the total number of messages sent between parties while performing π. Furthermore, let's fix two distinct parties P_i and P_j. To cover ideal-model adversaries $\mathcal{S}$ which do not guarantee timely delivery of the common key, we introduce the following environment $\mathcal{Z}_1$ (expecting to be run with the *dummy adversary* $\tilde{\mathcal{A}}$ in the real model):

1. Activate P_i with `(Establish-session,`sid`,`P_i`,`P_j`,initiator)`.
2. Activate P_j with `(Establish-session,`sid`,`P_j`,`P_i`,responder)`.
3. Advise the adversary to deliver all messages between P_i and P_j, but at most $m(k)$ messages in total.
4. If P_i or P_j outputs a key, call it α, resp. β; if both P_i and P_j output keys, output 1, else output 0.

Since π is terminating, in the real model $\mathcal{Z}_1$ always outputs 1. Moreover, as π securely realizes $\mathcal{F}_{\mathrm{KE}}$, $\mathcal{Z}_1$ must also output 1 in the ideal model in all but a negligible fraction of runs. That means we may assume that in a "normal" protocol run of π, the ideal-model adversary eventually delivers output to the parties (except in a negligible fraction of runs). A similar argument shows that π must guarantee matching keys (i. e., $\alpha = \beta$) in all but a negligible fraction of runs. To see this, we only need to modify $\mathcal{Z}_1$ in its fourth step, so that it outputs 1 exactly if $\alpha = \beta$. Now we are ready to formalize the attack discussed at the beginning of this section. Consider the following environment $\mathcal{Z}_2$, which also expects to communicate with the *dummy adversary* $\tilde{\mathcal{A}}$ in the real model:

1. Pick randomly $(b, \bar{b}) \in \{(i, j), (j, i)\}$.
2. Activate P_i with `(Establish-session,`sid`,`P_i`,`P_j`,initiator)`.

3. Activate P_j with `(Establish-session,`sid`,`P_j`,`P_i`,responder)`.
4. Instruct the adversary to corrupt P_b and to discard all messages possibly waiting to be delivered from P_b to $P_{\bar{b}}$.
5. Perform protocol π in the role of P_b, therefore send and receive messages through the corrupted "relay" P_b; let the adversary deliver all messages between P_b and $P_{\bar{b}}$.
6. Compare the output value of $P_{\bar{b}}$ with the local result of the key exchange protocol performed with $P_{\bar{b}}$ over P_b; if both match, output 1; otherwise output 0.

Now in the real model, the adversary $\tilde{\mathcal{A}}$ will follow precisely $\mathcal{Z}_2$'s instructions; consequently, a "normal" run of protocol π will take place between $P_{\bar{b}}$ (which expects to talk to P_b) and $\mathcal{Z}_2$. As $\alpha = \beta$ with overwhelming probability, the probability for $\mathcal{Z}_2$ to output 1 in the real model will be at most negligibly away from 1.

On the other hand, in the ideal model, the session key which will be output by the uncorrupted initiator $P_{\bar{b}}$ at the end of the simulated run of π (we'll call this key κ here) is fixed by $\mathcal{F}_{\mathrm{KE}}$ right after step 3, so at a time when neither initiator nor responder is corrupted. Consequently, κ is picked uniformly out of $\{0,1\}^k$ by $\mathcal{F}_{\mathrm{KE}}$. (Of course, in step 4 the simulator is allowed to corrupt P_j and thereby may get to know κ, but it is not able to *influence* κ.) For mimicking the real model, $\mathcal{S}$ must now be able to convince $\mathcal{Z}_2$ that the session key explicitly negotiated in step 5 is exactly κ. In other words, either $\mathcal{Z}_2$ succeeds in distinguishing the real from the ideal model, or π offers the initiator as well as the responder the possibility of "provoking" any output value κ. In case $\mathcal{Z}_2$ is *not* a successful distinguisher, we will construct from $\mathcal{Z}_2$ an environment $\mathcal{Z}_3$ which *must* be successful in distinguishing real from ideal. The reasoning will be as follows: if $\mathcal{S}$ can provoke keys "at wish" both in the roles of initiator and responder, then there must be a contradiction when $\mathcal{S}$ performs a key exchange *with itself* (more precisely, with a simulation of itself). This situation is illustrated in Figure 2.

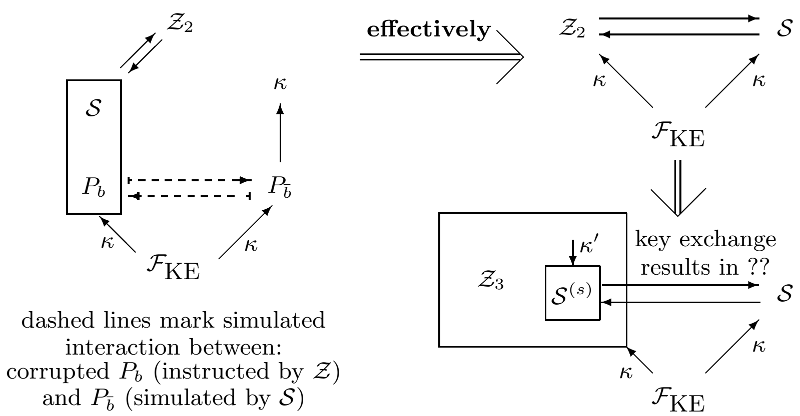

Fig. 2. The environments $\mathcal{Z}_2$ and $\mathcal{Z}_3$ from the proof of Proposition 1 in the ideal model.

Specifically, consider an environment $\mathcal{Z}_3$, which is a modification of $\mathcal{Z}_2$. Namely, we modify $\mathcal{Z}_2$ only from the fifth step on, in which $\mathcal{Z}_2$ performs protocol π in the role of P_b with $P_{\bar{b}}$. Instead of playing the role of an "honest" P_b with uniformly selected random tape, $\mathcal{Z}_3$ internally keeps a simulation of a *complete* ideal model, including simulated dummy parties $P_1^{(s)}, \ldots, P_n^{(s)}$, a simulated ideal functionality $\mathcal{F}_{\mathrm{KE}}^{(s)}$, and a simulation $\mathcal{S}^{(s)}$ of the simulator $\mathcal{S}$ itself. However, the role of the environment in $\mathcal{Z}_3$'s simulation is taken by a simulation $\mathcal{Z}_2^{(s)}$ of $\mathcal{Z}_2$ which in its first step selects b to be the $\bar{b}$ of $\mathcal{Z}_3$ and vice versa. (To avoid confusion, with b and $\bar{b}$, we mean in the following $\mathcal{Z}_3$'s choices of these variables.) The idea of this is to let $\mathcal{Z}_2^{(s)}$ corrupt $P_{\bar{b}}$ in the simulation and to let $\mathcal{S}^{(s)}$ perform a simulated run of π in the role of P_b with the non-simulated $P_{\bar{b}}$ (whose role is taken by $\mathcal{S}$ if we are in the ideal model). Therefore, all messages sent from $P_{\bar{b}}$ are forwarded to $P_b^{(s)}$ and vice versa. Finally, $\mathcal{Z}_3$ outputs 1 exactly if the local output of $P_{\bar{b}}$ matches that of $P_b^{(s)}$. (Again, if either of them does not generate output after $m(k)$ delivered messages, $\mathcal{Z}_3$ halts with output 0.)

In the real model, since we assumed $\mathcal{Z}_2$ not to be successful in distinguishing the real from the ideal model, $\mathcal{S}^{(s)}$ must be "successful" in performing a key exchange with a non-corrupted party $P_{\bar{b}}$ which yields as output exactly the key generated by the (simulated) ideal functionality $\mathcal{F}_{\mathrm{KE}}^{(s)}$. As in $\mathcal{Z}_3$'s simulation, the latter output is eventually delivered to $P_b^{(s)}$, $\mathcal{Z}_3$ will output 1 with overwhelming probability in the real model.

In the ideal model, either the protocol fails (i. e., either $\mathcal{S}$ or $\mathcal{S}^{(s)}$ does not deliver an output message from the ideal functionality to an uncorrupted party), or the local outputs of $P_b^{(s)}$ and $P_{\bar{b}}$ are *distinct* with overwhelming probability. (Note that $\mathcal{F}_{\mathrm{KE}}$ and the simulated $\mathcal{F}_{\mathrm{KE}}^{(s)}$ have independent random tapes from which they pick their respective output values.) In any case, $\mathcal{Z}_3$ outputs 0 in all but a negligible fraction of runs in the ideal model, thereby distinguishing the real from the ideal model. $\square$

Now we present a family $\{\mathcal{F}_{\mathrm{KE}}^{(i,j)}\}_{P_i,P_j}$ of functionalities intended to capture the requirements for key exchange.[7] More specifically, the functionality $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$ (presented in Figure 3) is aimed at modeling a key exchange between the parties P_i and P_j. This functionality is derived from the functionality $\mathcal{F}_{\mathrm{KE}}$ from [9], yet differs from it in several important aspects, see the discussion above. In the case of authenticated communication, we will show two common protocols to be securely realizing our key exchange functionality. (For unauthenticated communication in the sense of [9], one can use, e. g., an existentially unforgeable signature scheme to implement authenticated links.) Therefore, we start with

Definition 1. *A protocol $\pi^{(i,j)}$, parametrized by indices of two parties P_i and P_j, will be called a* universally composable key exchange protocol*, provided that*

[7] Here and from now on, we assume pairs of parties over which families of functionalities or protocols are indexed *not* to be of the form (i, i), i. e., we assume the participating parties to be distinct.

Functionality $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$

$\mathcal{F}_{\mathrm{KE}}^{(i,j)}$ proceeds as follows, running on security parameter k, with parties $P_1, \dots, P_n$ and an adversary $\mathcal{S}$.

1. Wait to receive values (`ready`,*sid*) from the parties P_i and P_j and from the adversary $\mathcal{S}$. When receiving (`ready`,*sid*) from either P_i or P_j, forward this message (including the sender identity) to $\mathcal{S}$.
2. After having received values (`ready`,*sid*) from P_i, P_j, and $\mathcal{S}$ (in any order), proceed as follows:
   (a) If both P_i and P_j are uncorrupted, choose $\kappa \xleftarrow{R} \{0,1\}^k$.
   (b) If at least one of the parties P_i and P_j is corrupted, send a message (`choose-key`,*sid*) to the adversary $\mathcal{S}$. Upon receiving an answer (`key`,*sid*,κ) from $\mathcal{S}$, extract the value of κ from it.

   Once κ is set, send (`key`,*sid*,κ) to P_i and P_j, and send (`key`,*sid*) to the adversary $\mathcal{S}$. Then halt.

Fig. 3. The modified key exchange functionality $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$

- *it has the same interface as $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$ (with respect to communication between $\mathcal{Z}$ and the parties),*
- *it involves communication only between P_i and P_j,*
- *once a party generates output, it immediately erases all internal information,*
- *when all messages between P_i and P_j are delivered, and neither P_i nor P_j gets corrupted, $\pi^{(i,j)}$ guarantees common output (i. e., matching keys) computationally indistinguishable from random k-bit strings, even when all the communication between P_i and P_j is made public,*
- *at the time the first party (either P_i or P_j) generates output, the other party has erased all protocol information other than the output unless it is corrupted; furthermore, at this point, the protocol involves only one more fixed "acknowledgment" message sent from the party which generated output to the one which did not yet do so.*

It should be remarked that the last of the requirements in Definition 1 can be interpreted as a special case of the `ack` property defined in [9], whereas the requirement for keys indistinguishable from random k-bit strings can be seen as a variant of SK-security (see [9]).

Example 1. Protocol $\textsc{dh}_{\mathcal{G},\mathcal{H}}^{(i,j)}$ (presented in Figure 4) is a variant of the common Diffie-Hellman key exchange protocol, derived from the protocol SIG-DH of [9]. At this $\mathcal{G} = \{G_\mu\}_\mu$ is a family of cyclic groups of prime order (explicitly given by a generator g) with hard decisional Diffie-Hellman (DDH) problem (cf., e. g., [5, Section 2]). For each group $\langle g \rangle \in \mathcal{G}$ we denote by $\mathcal{H}_{\langle g \rangle} = \{H_{\langle g \rangle,\nu}\}_\nu$ a family of pair-wise independent hash functions that is used to pass from group elements g^{xy} to bitstrings $H_{\langle g \rangle,\nu}(g^{xy})$ where $x, y \in \{1, \dots, |\langle g \rangle| - 1\}$: as the ideal functionality $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$ chooses the key as a random *bitstring* κ, we follow the approach in [21, Section 5.3.2] (see also [22]) and assume the parameters to be chosen such that the decisional Diffie-Hellman problem and the entropy smoothing theorem (cf., e. g., [19, Chapter 8]) imply the computational indistinguishability of the distributions $\{(g, g^x, g^y, \nu, H_{\langle g \rangle,\nu}(g^{xy}))\}_k$ and $\{(g, g^x, g^y, \nu, \kappa)\}_k$—with κ a random bitstring of length equal to the output length of $\mathcal{H}_{\langle g \rangle}$ and k the security param-

Protocol $\text{DH}^{(i,j)}_{\mathcal{G},\mathcal{H}}$

These are instructions for two parties P_i and P_j to carry out a key exchange. Prior to acting upon these instructions, each of the parties waits for an initial `(ready,`sid`)` input.

1. Dependent on the security parameter k, party P_i chooses a group $\langle g \rangle \in \mathcal{G}$ along with a generator g. Then P_i chooses $x \stackrel{R}{\leftarrow} \{1, \ldots, |\langle g \rangle| - 1\}$, calculates $\alpha = g^x$ and sends $(sid, \mathrm{D}(g), \alpha)$ to P_j, where $\mathrm{D}(g)$ is a description of $\langle g \rangle$ which also specifies the generator g.
2. Upon receiving from P_i a message $(sid, \mathrm{D}(g), \alpha)$ with $\mathrm{D}(g)$ being acceptable for the current security parameter, P_j chooses $y \stackrel{R}{\leftarrow} \{1, \ldots, |\langle g \rangle| - 1\}$ and a random index ν into the family $\mathcal{H}_{\langle g \rangle}$. Then P_j calculates $\beta = g^y$, $\gamma = \alpha^y$, and $\kappa = H_{\langle g \rangle, \nu}(\gamma)$, sends (sid, β, ν) to P_i, and erases all local information but κ.
3. Upon receipt of (sid, β, ν) from P_j, P_i calculates $\gamma = \beta^x$ and $\kappa = H_{\langle g \rangle, \nu}(\gamma)$, then erases all local information but κ and sends `(`sid`,done)` to P_j. Party P_i then outputs `(key,`sid`,`κ`)`, erases κ and halts.
4. Upon receipt of `(`sid`,done)` from P_i, party P_j outputs `(key,`sid`,`κ`)`, erases κ and halts.

Fig. 4. Protocol $\text{DH}^{(i,j)}_{\mathcal{G},\mathcal{H}}$

eter. We assume that for a group $G \in \mathcal{G}$ associated with security parameter k, the output length of $\mathcal{H}_G$ is exactly k.

While the protocol SIG-DH in [9] assumes that a suitable group description along with a group generator is provided to the protocol participants as 'initial information', in the protocol $\text{DH}^{(i,j)}_{\mathcal{G},\mathcal{H}}$ in Figure 4 a description $\mathrm{D}(g)$ of a suitable group, including the specification of a generator g, is explicitly transmitted within the protocol. To avoid incorrect choices by a (corrupted) initiator of the key exchange, we assume that for given security parameter and description $\mathrm{D}(g)$ one can verify in strict polynomial time whether $\langle g \rangle \in \mathcal{G}$ holds and $\langle g \rangle$ is acceptable for the security parameter, thus implying difficulty of the DDH problem in $\langle g \rangle$.[8] Having in mind practical proposals like IKEv2 [18] or JFKi, JFKr [1] where the agreement on the specific group is a relevant issue, this slightly more complicated formulation seems acceptable. From the formal point of view, this modeling also avoids the set-up assumption that implicitly is made when using "globally available" parameters (which depend on the security parameter).

From the construction it is clear that protocol $\text{DH}^{(i,j)}_{\mathcal{G},\mathcal{H}}$ fulfills the requirements for a universally composable key exchange protocol—note here that the use of signed messages is not necessary, as we assume authenticated communication.

Example 2. As another example, take a look at protocol $\text{PKKE}^{(i,j)}_{\text{PK}}$ in Figure 5, where $\text{PK} = (\mathtt{K}, \mathtt{E}, \mathtt{D})$ is a semantically secure public-key encryption scheme (see [16]). By the semantic security of PK, an eavesdropped encryption of the secret key κ reveals no information about κ to a polynomially bounded adversary, and thus protocol $\text{PKKE}^{(i,j)}_{\text{PK}}$ satisfies all the requirements of Definition 1 and can be called a universally composable key exchange protocol.

[8] As an example in which "wrong" parameter choices can be detected easily, $\langle g \rangle$ could be computed canonically from k, so there would be only *one* $\langle g \rangle \in \mathcal{G}$ associated with each k. Of course, hardness of the DDH problem then requires specific assumptions.

Protocol $\text{PKKE}^{(i,j)}_{\text{PK}}$

These are instructions for two parties P_i and P_j to carry out a key exchange. Prior to acting upon these instructions, each of the parties waits for an initial (`ready`,sid) input.

1. Party P_i generates a key pair (d,e) via $(d,e) \leftarrow \mathtt{K}(k)$ and sends the public key e in form of the message (sid,e) to P_j while locally storing the corresponding private key d.
2. Upon receiving a message (sid,e) from P_i, party P_j first chooses a random k-bit string κ, then computes κ's encryption with respect to the public key e via $c \leftarrow \mathtt{E}(e,\kappa)$, sends ($sid$,$c$) to P_i, and erases all local information except the key κ.
3. Upon receiving (sid,c) from P_j, party P_i computes the decryption κ of c via $\kappa \leftarrow \mathtt{D}(d,c)$, then erases all local information but κ, sends (sid,`done`) to party P_j, outputs (`key`,sid,κ), erases κ and halts.
4. Upon receiving (sid,`done`) from P_i, party P_j outputs (`key`,sid,κ), erases κ and halts.

Fig. 5. Protocol $\text{PKKE}^{(i,j)}_{\text{PK}}$

Proposition 2. *Suppose we are in a model with authenticated links and trusted erasures. Assume further that for two fixed different parties P_i and P_j, protocol $\pi^{(i,j)}$ is a* universally composable key exchange protocol *as defined above. Then $\pi^{(i,j)}$ securely realizes functionality $\mathcal{F}^{(i,j)}_{\text{KE}}$ with respect to adaptive adversaries.*

Proof. The proof can be found in Appendix B. □

4 A Stronger Notion of Key Exchange

The description of $\mathcal{F}_{\text{KE}}$ as well as the one of $\mathcal{F}^{(i,j)}_{\text{KE}}$ allows the adversary to freely *choose* the session key if at least one participating party is corrupted. This also holds for the 'relaxed' key exchange functionality $\mathcal{F}^{\mathcal{N}}_{\text{RKE}}$ from [9], and one may ask whether this "worst case modeling" of corrupted parties is indeed justified. E. g., in the protocol $\text{PKKE}^{(i,j)}_{\text{PK}}$ the consequences of corrupting P_i and of corrupting P_j are intuitively quite different: a malicious party P_j has complete control over the resulting key κ, and it can, e. g., choose a value for κ that has been chosen earlier by some "outsider" P_a. If κ is later used to encrypt messages sent by P_i, then the "outsider" P_a will be able to read all these messages without any communication between P_j and P_a taking place during or after the key exchange of P_i and P_j. For doing so, P_a does not even have to eavesdrop the communication between P_i and P_j during the key exchange. On the other hand, a corrupted P_i is not able to influence an honest choice of κ performed by P_j.

In the Diffie-Hellman protocol $\text{DH}^{(i,j)}_{\mathcal{G},\mathcal{H}}$ a similar "asymmetry" exists, but this property is not reflected in the definitions of the mentioned key exchange functionalities, either. As in some situations an additional security guarantee as provided by $\text{DH}^{(i,j)}_{\mathcal{G},\mathcal{H}}$ and $\text{PKKE}^{(i,j)}_{\text{PK}}$ may be desirable, in the sequel we want to put this observation on firmer grounds. The functionality $\mathcal{F}^{(i,j)}_{\text{KE+}}$, which we introduce for this purpose, is certainly not completely satisfactory from a conceptual point of view. Nevertheless, we think it gives ample evidence for the possibility to provide non-trivial (formal) security guarantees even if a participant in a key exchange is corrupted: A natural modification of $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$ might be the one presented in Figure 6. The functionality $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$ guarantees random keys even when P_i gets corrupted. However, as soon as P_j is corrupted, the adversary may freely determine the common key κ as with $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$.

Functionality $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$

$\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$ proceeds as follows, running on security parameter k, with parties $P_1, \ldots, P_n$ and an adversary $\mathcal{S}$.

1. Wait to receive values (`ready`,sid) from the parties P_i and P_j and from the adversary $\mathcal{S}$. When receiving (`ready`,sid) from either P_i or P_j, forward this message (including the sender identity) to $\mathcal{S}$.
2. After having received values (`ready`,sid) from P_i, P_j, and $\mathcal{S}$ (in any order), proceed as follows:
   (a) If P_j is *not* corrupted, choose κ uniformly from $\{0,1\}^k$.
   (b) If P_j *is* corrupted, send a message (`choose-key`,sid) to the adversary $\mathcal{S}$. Upon receiving an answer (`key`,sid,κ) from $\mathcal{S}$, extract the value of κ from it.

   Once κ is set, send (`key`,sid,κ) to P_i and P_j, and send (`key`,sid) to the adversary. Then halt.

Fig. 6. The key exchange functionality $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$

Remark 4. Unfortunately, neither protocol $\mathrm{DH}_{\mathcal{G},\mathcal{H}}^{(i,j)}$ nor protocol $\mathrm{PKKE}_{\mathrm{PK}}^{(i,j)}$ securely realizes $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$. This holds also for the "side-reversed" versions $\mathrm{DH}_{\mathcal{G},\mathcal{H}}^{(j,i)}$ and $\mathrm{PKKE}_{\mathrm{PK}}^{(j,i)}$. To see this, consider the following environment $\mathcal{Z}$, expecting to be run with the dummy adversary in the real model: $\mathcal{Z}$ corrupts P_i right before the session key is output, yet does everything P_i would do to carry out a key exchange with P_j, thereby communicating over the corrupted "relay party" P_i with P_j. When P_j finally outputs a key, $\mathcal{Z}$ checks if it is the same $\mathcal{Z}$ itself generated in its key exchange with P_j. If and only if this is the case, $\mathcal{Z}$ outputs 1. Furthermore, when P_j outputs no key at a time it should do in the real model or P_j sends messages of the wrong format or no messages at all, $\mathcal{Z}$ outputs 0.

For a very brief analysis, first note that by construction of the protocols in question, $\mathcal{Z}$ always outputs 1 in the real model. On the other hand, in the ideal model with functionality $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$, regardless of the simulator $\mathcal{S}$ and the messages simulated between P_i and P_j, P_j's output is chosen uniformly from $\{0,1\}^k$ since only P_i, but not P_j is corrupted. So P_j outputs the key $\mathcal{Z}$ locally generated in its key exchange only in a negligible fraction of runs and thus, $\mathcal{Z}$ outputs 0 in the ideal model with overwhelming probability. Hence $\mathcal{Z}$ serves as a distinguisher between any of the abovementioned protocols and $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$, as stated in Figure 6. In the next section a relaxation of $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$ is given, which yields a "stronger" security notion than $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$ but still is securely realized by $\mathrm{DH}_{\mathcal{G},\mathcal{H}}^{(i,j)}$ and $\mathrm{PKKE}_{\mathrm{PK}}^{(i,j)}$.

Consider protocol $\mathrm{PAD}^{(i,j)}$ given in Figure 7. As it makes use of exactly one instance of the ideal functionality $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$, it can be seen as an extension to any protocol intended to realize $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$. In the next proposition, we will show $\mathrm{PAD}^{(i,j)}$ to be securely realizing $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$. By the composition theorem of [6], this means

Protocol $\textsc{pad}^{(i,j)}$

These are instructions for two parties P_i and P_j to carry out a key exchange. Prior to acting upon these instructions, each of the parties waits for an initial `(ready,`sid`)` input. Furthermore, the parties expect to be run in the $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$-hybrid model, i. e., with access to a polynomial number of instances of the ideal functionality $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$.

1. Immediately after having received the initial `(ready,`sid`)` message, P_i as well as P_j sends the message `(ready,0)` to the $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$-instance with session ID 0.
2. Then P_j, after having received the key $\bar{\kappa}$ from this instance of $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$, uniformly chooses $\psi \in \{0,1\}^k$ and calculates $\kappa = \psi \oplus \bar{\kappa}$. It then erases all local information but κ and sends (sid,ψ) to P_i.
3. Upon receiving a message (sid,ψ) and after having received a key $\bar{\kappa}$ from the $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$-instance with session ID 0, P_i first calculates $\kappa = \psi \oplus \bar{\kappa}$ and erases all local information but κ. Then P_i sends `(`sid`,done)` to P_j, outputs `(key,`sid`,`κ`)`, erases κ and halts.
4. Upon receipt of `(`sid`,done)` from P_i, party P_j outputs `(key,`sid`,`κ`)`, erases κ and halts.

Fig. 7. Protocol $\textsc{pad}^{(i,j)}$

that for any protocol π which securely realizes $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$, the extension $\textsc{pad}_{\pi}^{(i,j)}$ (which is essentially protocol $\textsc{pad}^{(i,j)}$, but canonically uses instances of π as subprotocols instead of talking to instances of $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$) is guaranteed to still realize $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$ securely. So we have the interesting situation that neither $\textsc{dh}_{\mathcal{G},\mathcal{H}}^{(i,j)}$ nor $\textsc{pkke}_{\mathrm{PK}}^{(i,j)}$ securely realizes $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$ "by itself", but already simple refinements of these protocols do so. Namely, since both of them securely realize $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$, their extensions $\textsc{pad}_{\pi}^{(i,j)}$, with π taken as one of them, securely realize $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$.

Proposition 3. *Protocol* $\textsc{pad}^{(i,j)}$ *securely realizes* $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$ *in the* $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$*-hybrid model with respect to adaptive adversaries.*

Proof. The proof can be found in Appendix B. □

In principle, the protocol $\textsc{pad}^{(i,j)}$ is sufficient for securely realizing $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$ on the basis of a Diffie-Hellman-like key exchange like $\textsc{dh}_{\mathcal{G},\mathcal{H}}^{(i,j)}$. However, the additional communication introduced by protocol $\textsc{pad}^{(i,j)}$ might seem superfluous when dealing with a protocol like $\textsc{pkke}_{\mathrm{PK}}^{(i,j)}$, which intuitively provides the desired additional security guarantee "by itself". In Appendix A, we show how to catch this intuition by means of a suitable ideal functionality.

5 Conclusions

The above discussion shows that a universally composable notion of key exchange, as expressed through the functionality $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$, can be realized through quite natural key exchange protocols. However, additional security guarantees which are provided, e. g., by the described Diffie-Hellman based realization of $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$ are not reflected by functionalities like $\mathcal{F}_{\mathrm{KE}}$, $\mathcal{F}_{\mathrm{RKE}}^{\mathcal{N}}$, or $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$, and we have shown that at least a part of these additional qualities can be captured by an appropriately "strengthened" functionality that makes use of a non-information oracle.

Nevertheless, for all notions of key exchange discussed above, the adversary has complete control over the result of the key exchange, if the "wrong" party is corrupted, and it seems that a Diffie-Hellman-like key exchange protocol can allow for a stronger guarantee. E. g., for appropriate families $\mathcal{H}_{\langle g \rangle}$ the following variation of $\mathrm{DH}^{(i,j)}_{\mathcal{G},\mathcal{H}}$ seems to limit the possibilities of a corrupted P_j somewhat more: let P_j choose and send the index ν into the family $\mathcal{H}_{\langle g \rangle}$ directly after P_i has fixed the group description $\mathrm{D}(g)$ (and before $\alpha = g^x$ is received). In this protocol it is not obvious how P_j could force a specific key—even if the index ν is not chosen at random.

It remains an interesting open question if such additional guarantees of certain key exchange protocols can be captured through an appropriate ideal functionality in the framework of universal composition.

Acknowledgments

We thank Ran Canetti, Dieter Gollmann, María Isabel González Vasco and the anonymous referees for helpful discussions and comments.

References

1. William Aiello, Steven M. Bellovin, Matt Blaze, Ran Canetti, John Ioannidis, Angelos D. Keromytis, and Omer Reingold. Efficient, DoS-Resistant, Secure Key Exchange for Internet Protocols. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 48–58. ACM Press, 2002.
2. Michael Backes, Birgit Pfitzmann, and Michael Waidner. A Universally Composable Cryptographic Library. Cryptology ePrint Archive, Report 2003/015, January 2003. `http://eprint.iacr.org/2003/015/`.
3. Mihir Bellare, Ran Canetti, and Hugo Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 419–428. ACM Press, 1998. Full version at `http://eprint.iacr.org/1998/009`.
4. Mihir Bellare and Phillip Rogaway. Provably Secure Session Key Distribution: the Three Party Case. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing*, pages 57–66. ACM Press, 1995.
5. Dan Boneh. The Decision Diffie-Hellman Problem. In Joe P. Buhler, editor, *Proceedings of the Third Algorithmic Number Theory Symposium*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63. Springer, 1998.
6. Ran Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *Proceedings of 42nd Annual Symposium on Foundations of Computer Science, FOCS 2001*, pages 136–145. IEEE Computer Society, 2001. Full version at `http://eprint.iacr.org/2000/067`.
7. Ran Canetti and Marc Fischlin. Universally Composable Commitments. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 19–40. Springer, 2001. Full version at `http://eprint.iacr.org/2001/055`.
8. Ran Canetti and Hugo Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In Birgit Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Proceedings*, volume 2045 of *Lecture Notes in*

Computer Science. Springer, 2001. Full version at `http://eprint.iacr.org/2002/047`.

9. Ran Canetti and Hugo Krawczyk. Universally Composable Notions of Key Exchange and Secure Channels. In Lars R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings*, volume 2332 of *Lecture Notes in Computer Science*. Springer, 2002. All citations refer to the full version at `http://eprint.iacr.org/2002/059`.
10. Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the Limitations of Universally Composable Two-Party Computation Without Set-up Assumptions. In *Proceedings of EUROCRYPT 2003*, Lecture Notes in Computer Science. Springer, 2003. To appear.
11. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally Composable Two-Party and Multi-party Secure Computation. In *Proceedings on Thirty-Fourth Annual ACM Symposium on Theory of Computing, STOC 2002*, pages 494–503. ACM Press, 2002.
12. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally Composable Two-Party and Multi-party Secure Computation, July 2003. Full (and revised) version of [11], available at `http://eprint.iacr.org/2002/140`.
13. Ivan B. Damgård. Presentation of [14] at CRYPTO 2002, 2002.
14. Ivan B. Damgård and Jesper B. Nielsen. Perfect Hiding and Perfect Binding Universally Composable Commitment Schemes with Constant Expansion Factor. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*, pages 581–596. Springer, 2002. Full version at `http://eprint.iacr.org/2001/091`.
15. Danny Dolev and Andrew C. Yao. On the Security of Public Key Protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, March 1983.
16. Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *Journal of Computer and System Science*, 28, 1984.
17. Dennis Hofheinz, Jörn Müller-Quade, and Rainer Steinwandt. On Modeling IND-CCA Security in Cryptographic Protocols. Cryptology ePrint Archive, Report 2003/024, February 2003. `http://eprint.iacr.org/2003/024`.
18. Internet Key Exchange (IKEv2) Protocol. Charlie Kaufman, editor. IPSEC Working Group INTERNET-DRAFT draft-ietf-ipsec-ikev2-06.txt, March 2003. Available at `http://www.ietf.org/internet-drafts/draft-ietf-ipsec-ikev2-06.txt`.
19. Michael Luby. *Pseudorandomness and Cryptographic Applications*. Princeton Computer Science Notes. Princeton University Press, 1996.
20. Birgit Pfitzmann and Michael Waidner. A Model for Asynchronous Reactive Systems and its Application to Secure Message Transmission. In *IEEE Symposium on Research in Security and Privacy*, pages 184–200. IEEE Computer Society Press, 2001. Full version at `http://eprint.iacr.org/2000/066`.
21. Victor Shoup. On Formal Models for Secure Key Exchange. Cryptology ePrint Archive, Report 1999/012, 1999. `http://eprint.iacr.org/1999/012`.
22. Michael Steiner. *Secure Group Key Agreement*. PhD thesis, Universität des Saarlandes, 2002. Online available at `http://www.semper.org/sirene/publ/Stei_02.thesis-final.pdf`.

A An Additional Security Guarantee of $\mathrm{DH}_{\mathcal{G},\mathcal{H}}^{(i,j)}$ and $\mathrm{PKKE}_{\mathrm{PK}}^{(i,j)}$

As explained in Remark 4, we cannot hope that $\mathrm{PKKE}_{\mathrm{PK}}^{(i,j)}$ or $\mathrm{DH}_{\mathcal{G},\mathcal{H}}^{(i,j)}$ provide a secure realization of $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$. To show that these protocols do in fact guarantee strictly more than needed for realizing $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$, we utilize so-called *non-information oracles*, a tool that has been introduced in [9]:

Definition 2. *Let $\mathcal{N}$ be an ITM with strict polynomial running time. Then $\mathcal{N}$ is a* non-information oracle *if no ITM $\mathcal{M}$, having interacted with $\mathcal{N}$ on security parameter k, can distinguish with non-negligible probability between the local output of $\mathcal{N}$ and a value drawn uniformly from $\{0,1\}^k$.*

Example 3. In the proof of the next proposition, we will make use of the following ITM $\mathcal{N}_{\mathcal{G},\mathcal{H}}$ (with $\mathcal{G},\mathcal{H}$ as in Example 1): when activated for the first time, $\mathcal{N}_{\mathcal{G},\mathcal{H}}$ randomly chooses $\langle\bar{g}\rangle \in \mathcal{G}$ (in dependence of the security parameter), two values $\bar{x},\bar{y} \in \{1,\ldots,|\langle\bar{g}\rangle|-1\}$, and a random index $\bar{\nu}$ into the family $\mathcal{H}_{\langle\bar{g}\rangle}$. Then $\mathcal{N}_{\mathcal{G},\mathcal{H}}$ sends a description $\mathrm{D}(\bar{g})$ (as in Example 1) as well as $\bar{\alpha} = \bar{g}^{\bar{x}}$, $\bar{\beta} = \bar{g}^{\bar{y}}$, and $\bar{\nu}$ to the ITM it interacts with. After this, when receiving a message `accept`, it locally outputs $H_{\langle\bar{g}\rangle,\bar{\nu}}(\bar{g}^{\bar{x}\bar{y}})$ and halts. On the other hand, upon receiving a value `reject`, $\mathcal{N}_{\mathcal{G},\mathcal{H}}$ sends $\bar{x}$ and $\bar{y}$ to the ITM it interacts with and waits to receive a pair $(\mathrm{D}(g'),\alpha)$ with $\mathrm{D}(g')$ describing a $\langle g'\rangle \in \mathcal{G}$ that is acceptable for the current security parameter and $\alpha \in \langle g'\rangle \setminus \{1\}$. Then $\mathcal{N}_{\mathcal{G},\mathcal{H}}$ uniformly selects $r \in \{1,\ldots,|\langle g'\rangle|-1\}$ and an index ν' into the family $\mathcal{H}_{\langle g'\rangle}$, locally outputs $H_{\langle g'\rangle,\nu'}(\alpha^r)$, and halts.

We claim that under the decisional Diffie-Hellman assumption, $\mathcal{N}_{\mathcal{G},\mathcal{H}}$ is a non-information oracle. To show this, assume that there is an ITM $\mathcal{M}$ that, after running with $\mathcal{N}_{\mathcal{G},\mathcal{H}}$, successfully distinguishes (i. e., differs in its output distribution) the local output of $\mathcal{N}_{\mathcal{G},\mathcal{H}}$ from a random k-bit string. When we modify $\mathcal{M}$ to never issue a `reject` message (possibly followed by some $(\mathrm{D}(g'),\alpha)$), but instead to send an `accept` message and to halt with random output without even looking at the challenge, this cannot downgrade $\mathcal{M}$'s advantage in distinguishing. This is so since in case of a `reject` message, followed by some $(\mathrm{D}(g'),\alpha)$ with $\alpha \in \langle g'\rangle \setminus \{1\}$, $\mathcal{N}_{\mathcal{G},\mathcal{H}}$ outputs the hash value of a uniformly selected element from $\langle g'\rangle \setminus \{1\}$ (for $\langle g'\rangle$ is of prime order and therefore $\langle g'\rangle = \langle\alpha\rangle$), this group element about which $\mathcal{M}$ has no information whatsoever.

Thus $\mathcal{M}$ is able to distinguish random k-bit strings from the hash values of group elements $\bar{g}^{\bar{x}\bar{y}} \in \langle\bar{g}\rangle \setminus \{1\}$. By the universal hash property of $\mathcal{H}_{\langle\bar{g}\rangle}$, this means that $\mathcal{M}$ can also distinguish triples $(\bar{g}^{\bar{x}},\bar{g}^{\bar{y}},\bar{g}^{\bar{x}\bar{y}})$ from triples $(\bar{g}^{\bar{x}},\bar{g}^{\bar{y}},\bar{g}^{r})$, hence contradicting the decisional Diffie-Hellman assumption.[9]

Now we are ready to give the definition of a key exchange functionality which, on the one hand, guarantees "essentially" random keys which are not predictable

[9] Formally, $\mathcal{M}$ only distinguishes triples $(\bar{g}^{\bar{x}},\bar{g}^{\bar{y}},\bar{g}^{\bar{x}\bar{y}})$ from triples $(\bar{g}^{\bar{x}},\bar{g}^{\bar{y}},\bar{g}^{r})$ both subject to the condition $\bar{x},\bar{y},r \neq 0$. Yet when choosing $\bar{x}$, $\bar{y}$, and r at random, this happens only in a negligible number of cases, and thus can be neglected.

or influencable by the adversary even when one party is corrupted. Yet, on the other hand, this fuctionality is securely realized by both $\mathrm{DH}^{(i,j)}_{\mathcal{G},\mathcal{H}}$ and $\mathrm{PKKE}^{(i,j)}_{\mathrm{PK}}$. (As with $\mathcal{F}^{(i,j)}_{\mathrm{KE+}}$, the party which "may" safely be corrupted without losing the feature of random keys needs to be fixed in advance.) More specifically, consider the family $\{\mathcal{F}^{\mathcal{N},(i,j)}_{\mathrm{KE+}}\}_{\mathcal{N},P_i,P_j}$, parametrized by a non-information oracle $\mathcal{N}$ and the indices of two parties P_i and P_j, specified in Figure 8.

Functionality $\mathcal{F}^{\mathcal{N},(i,j)}_{\mathrm{KE+}}$

$\mathcal{F}^{\mathcal{N},(i,j)}_{\mathrm{KE+}}$ proceeds as follows, running on security parameter k, with parties $P_1,\ldots,P_n$ and an adversary $\mathcal{S}$.

1. Invoke a new copy of $\mathcal{N}$ with fresh random input. Allow $\mathcal{S}$ to interact with $\mathcal{N}$, i. e., to receive messages from and send messages to $\mathcal{N}$. If P_i or P_j gets corrupted at a time $\mathcal{N}$ already produced output, or $\mathcal{N}$ produces output at a time at least one of these parties is corrupted, send the complete state of $\mathcal{N}$ (including randomness and in particular the local output) to the adversary. (Of course, after step 3, $\mathcal{F}^{\mathcal{N},(i,j)}_{\mathrm{KE+}}$ has halted and this is no longer possible.)
2. Wait to receive values (`ready`,*sid*) from the parties P_i and P_j and from the adversary $\mathcal{S}$. When receiving (`ready`,*sid*) from either P_i or P_j, forward this message (including the sender identity) to $\mathcal{S}$.
3. After having received values (`ready`,*sid*) from P_i, P_j, and $\mathcal{S}$ (in any order), and after $\mathcal{N}$ produced local output, proceed as follows:
   (a) If neither P_i nor P_j is corrupted, choose κ uniformly from $\{0,1\}^k$.
   (b) If only P_i is corrupted, but *not* P_j, set κ to the local output of $\mathcal{N}$.
   (c) If P_j *is* corrupted, send a message (`choose-key`,*sid*) to the adversary $\mathcal{S}$. Upon receiving an answer (`key`,*sid*,κ) from $\mathcal{S}$, extract κ from it.

   Once κ is set, send (`key`,*sid*,κ) to P_i and P_j, and send (`key`,*sid*) to the adversary. Then halt.

Fig. 8. The key exchange functionality $\mathcal{F}^{\mathcal{N},(i,j)}_{\mathrm{KE+}}$

Proposition 4. *Presuming authenticated links and trusted erasures, protocol* $\mathrm{DH}^{(i,j)}_{\mathcal{G},\mathcal{H}}$ *securely realizes functionality* $\mathcal{F}^{\mathcal{N},(i,j)}_{\mathrm{KE+}}$ *with respect to adaptive adversaries for an appropriate non-information oracle* $\mathcal{N}=\mathcal{N}_{\mathcal{G},\mathcal{H}}$ *under the decisional Diffie-Hellman assumption.*

Proof. As already mentioned above, $\mathcal{N}=\mathcal{N}_{\mathcal{G},\mathcal{H}}$ is the non-information oracle from Example 3. We now present a simulator $\mathcal{S}$ mimicking attacks carried out by the dummy adversary on protocol $\mathrm{DH}^{(i,j)}_{\mathcal{G},\mathcal{H}}$. The idea behind the simulation is as follows: $\mathcal{S}$ is provided with a transcript of a Diffie-Hellman key exchange by $\mathcal{N}$. It then simulates messages from this transcript between P_i and P_j. When one of the parties is corrupted, $\mathcal{N}$ (resp., $\mathcal{F}^{\mathcal{N},(i,j)}_{\mathrm{KE+}}$) provides $\mathcal{S}$ with corresponding secret information consistent with the protocol transcript. Furthermore, if P_i is corrupted, $\mathcal{S}$ has the chance to perform the "second half" of a key exchange with $\mathcal{N}$ to provide $\mathcal{Z}$ with a consistent view of a party actually taking part in the key exchange. If P_j is corrupted, $\mathcal{S}$ may even pick the common key freely, reflecting that the message sent from P_j to P_i in the Diffie-Hellman key exchange fully determines this common key.

So $\mathcal{S}$ behaves exactly like the simulator $\mathcal{S}_\pi^{(i,j)}$ from the proof of Proposition 2 (with protocol $\mathrm{DH}_{\mathcal{G},\mathcal{H}}^{(i,j)}$ taken as π), with the following exceptions:

- When being supplied by $\mathcal{N}$ with $\mathrm{D}(\bar{g})$, $\bar{\alpha}$, $\bar{\beta}$, and $\bar{\nu}$, $\mathcal{S}$ stores these for future use. (Note that this happens at the first activation of the ideal functionality, so before the actual protocol simulation takes place.)
- When the simulated $P_i^{(s)}$ wishes to send a message $(sid,\mathrm{D}(g),\alpha)$ to $P_j^{(s)}$, and P_i is not corrupted, $\mathcal{S}$ simulates a message $(sid,\mathrm{D}(\bar{g}),\bar{\alpha})$ from P_i to P_j instead.
- When $P_j^{(s)}$ is delivered a message $(sid,\mathrm{D}(g),\alpha)$ with $\alpha \in \langle g \rangle \setminus \{1\}$, and P_j is uncorrupted, then $\mathcal{S}$ proceeds as follows: if P_i is uncorrupted (then we have $\alpha = \bar{\alpha}$ and $\mathrm{D}(g) = \mathrm{D}(\bar{g})$), $\mathcal{S}$ sends `accept` to $\mathcal{N}$ and simulates a message $(sid,\bar{\beta},\bar{\nu})$ from P_j to P_i; if, on the other hand, P_i is corrupted, $\mathcal{S}$ sends $(\mathrm{D}(g), \alpha)$ to $\mathcal{N}$ and, when receiving $\mathcal{N}$'s complete state (which by definition happens immediately afterwards), simulates a message (sid,g^r,ν') from P_j to P_i, where r and ν' are extracted from $\mathcal{N}$'s state.
- When $\mathcal{S}$ is requested to corrupt P_i, then $P_i^{(s)}$'s internal state first has to be modified so as to match the simulated protocol execution. If P_i already erased internal data, only the common key may have to be acquired from either $\mathcal{N}$'s state (if so far, no uncorrupted party generated output) or from $\mathcal{F}_{\mathrm{KE+}}^{\mathcal{N},(i,j)}$ itself by corrupting the dummy party P_i and delivering $\mathcal{F}_{\mathrm{KE+}}^{\mathcal{N},(i,j)}$'s output message to P_i. If, on the other hand, $P_i^{(s)}$ already chose, but did not yet erase its secret exponent x, $\mathcal{S}$ has to obtain the corresponding secret exponent $\bar{x}$ from $\mathcal{N}$: if $P_j^{(s)}$ has not yet received its first message from $P_i^{(s)}$, $\mathcal{S}$ does this by sending `reject` to $\mathcal{N}$ (by definition, $\mathcal{N}$ immediately replies with $\bar{x}$ and $\bar{y}$); else, $\mathcal{S}$ has already sent `accept` to $\mathcal{N}$ and therefore gets to know $\mathcal{N}$'s complete state immediately. Now $\mathcal{S}$ modifies the internal state, which is sent to $\mathcal{Z}$ as that of P_i, to be consistent with the exponent $\bar{x}$ and the group description $\mathrm{D}(\bar{g})$.
- When $\mathcal{S}$ is requested to corrupt P_j at a time $\mathcal{S}$ has already simulated a message back from P_j to P_i, $\mathcal{S}$ may have to provide $\mathcal{Z}$ with a key consistent with the preceding protocol transcript. (Note that P_j erases all other secret information in the same activation it generates it, so $\mathcal{S}$ needs never provide such "temporary secrets".) If no uncorrupted party generated output so far, $\mathcal{S}$ can obtain this common key by extracting $\mathcal{N}$'s output from $\mathcal{N}$'s state, with which $\mathcal{F}_{\mathrm{KE+}}^{\mathcal{N},(i,j)}$ provides $\mathcal{S}$ upon corruption of P_j. However, if some party already generated output, the ideal functionality already halted and therefore, the common key has to be acquired by corrupting the dummy party P_j and delivering the output message from $\mathcal{F}_{\mathrm{KE+}}^{\mathcal{N},(i,j)}$ to P_j.

The analysis of $\mathcal{S}$ is very similar to that of $\mathcal{S}_\pi^{(i,j)}$ in the proof of Proposition 2; we therefore only treat the differences caused by the above modifications. First, note that the states of corrupted parties with which $\mathcal{S}$ supplies the environment machine are by construction always consistent with the messages sent by the respective party prior to its corruption. In particular, this holds although the messages simulated by $\mathcal{S}$ between P_i and P_j are in general not those sent between $P_i^{(s)}$ and $P_j^{(s)}$. Moreover, by construction, the common key agreed upon in the ideal model is consistent with the messages between P_i and P_j as long as at least one of them is corrupted: when P_i is corrupted *before* its first message to P_j is delivered, the common key is consistent with the sent messages since $\mathcal{N}$ then fixes the key according to the message actually sent to an uncorrupted P_j. Later corruptions of P_i do not influence the key. When P_j is corrupted, $\mathcal{S}$ may freely choose the key and can thereby guarantee consistency of the common key by itself exactly as $\mathcal{S}_\pi^{(i,j)}$. If, on the other hand, neither P_i nor P_j gets corrupted, then the common key is indistinguishable from a k-bit random string by the universal hash property of $\mathcal{H}_{\langle g\rangle}$ in combination with the decisional Diffie-Hellman assumption. Putting all this together, we can now apply the argumentation of the proof of Proposition 2. □

Proposition 5. *When supposing authenticated links and trusted erasures, protocol* $\text{PKKE}_{\text{PK}}^{(i,j)}$ *securely realizes functionality* $\mathcal{F}_{\text{KE+}}^{\mathcal{N},(i,j)}$ *with respect to adaptive adversaries for an appropriate non-information oracle* $\mathcal{N} = \mathcal{N}_{\text{PK}}$ *once* PK *is a semantically secure public-key cryptosystem.*

Proof. The proof is very similar to the proof of Proposition 4, yet much easier, as by construction of $\text{PKKE}_{\text{PK}}^{(i,j)}$, P_j may choose the common key by itself. The non-information oracle $\mathcal{N} = \mathcal{N}_{\text{PK}}$ generates a key pair $(\bar{d}, \bar{e})$ via the key generation algorithm $\mathtt{K}$ (internally invoked on input k) and stores it. Then $\mathcal{N}$ chooses a random k-bit string $\bar{\kappa}$, encrypts it via $\bar{c} \leftarrow \mathtt{E}(\bar{e}, \bar{\kappa})$ and sends the public key $\bar{e}$ and the ciphertext $\bar{c}$ to the ITM it interacts with. It then locally outputs $\bar{\kappa}$ and halts. By the semantic security of PK, $\mathcal{N}$ has the non-information property.

We shortly describe the simulator $\mathcal{S}$ mimicking attacks carried out by the dummy adversary on $\text{PKKE}_{\text{PK}}^{(i,j)}$. In particular, $\mathcal{S}$ differs from the simulator $\mathcal{S}_\pi^{(i,j)}$ (with protocol $\text{PKKE}_{\text{PK}}^{(i,j)}$ taken as π) only as follows: when P_i is uncorrupted at that time, $\mathcal{S}$ replaces an initial message (sid, e) sent from $P_i^{(s)}$ to $P_j^{(s)}$ with a message $(sid, \bar{e})$, where the public key $\bar{e}$ together with a ciphertext $\bar{c}$ is obtained from $\mathcal{N}$ at the beginning of the protocol run. Consequently, a message (sid, c) sent back from $P_j^{(s)}$ (with uncorrupted P_j) to $P_i^{(s)}$ is replaced by a simulation of the message $(sid, \bar{c})$. Upon corruption requests of P_i (resp., P_j), $\mathcal{S}$ first modifies the internal data of $P_i^{(s)}$ (resp., $P_j^{(s)}$) to be consistent with $\bar{e}$, $\bar{d}$, and $\bar{\kappa}$ (resp., $\bar{e}$, $\bar{c}$, and $\bar{\kappa}$), where upon such a corruption, $\bar{d}$ and $\bar{\kappa}$ are extracted from $\mathcal{N}$'s state with which $\mathcal{S}$ is supplied. □

Remark 5. The requirement for trusted erasures might seem a bit hard. If one is willing to give up perfect forward secrecy, one can modify $\mathcal{F}_{\text{KE+}}^{\mathcal{N},(i,j)}$ so as to be realizable by the non-erasing counterparts of the protocols $\text{DH}_{\mathcal{G},\mathcal{H}}^{(i,j)}$ and $\text{PKKE}_{\text{PK}}^{(i,j)}$. Namely, $\mathcal{F}_{\text{KE+}}^{\mathcal{N},(i,j)}$ has to be modified to ask the non-information oracle $\mathcal{N}$ for a key even when neither P_i nor P_j is corrupted at the time the key is fixed. Furthermore, $\mathcal{F}_{\text{KE+}}^{\mathcal{N},(i,j)}$ must not halt after having sent the key to P_i and P_j; instead, even upon *later* corruptions of P_i or P_j, it has to send the complete state of the (terminated) non-information oracle $\mathcal{N}$ to supply the adversary with the (in the real model non-erased) secret information of the simulated parties.

B Proof of Proposition 2

Proof. Consider the simulator $\mathcal{S}_\pi^{(i,j)}$ presented in Figure 9 mimicking attacks carried out by the dummy adversary $\tilde{\mathcal{A}}$ on protocol $\pi^{(i,j)}$. Fix an environment $\mathcal{Z}$ trying to distinguish between execution of protocol $\pi^{(i,j)}$ together with the dummy adversary $\tilde{\mathcal{A}}$ and running with the ideal functionality $\mathcal{F}_{\text{KE}}^{(i,j)}$ and the simulator $\mathcal{S}_\pi^{(i,j)}$. As parties different from P_i and P_j are not involved at all in protocol $\pi^{(i,j)}$, without losing generality we may assume $\mathcal{Z}$ not to request corruptions of parties P_l with $l \notin \{i,j\}$.

In a first step, we consider a modified $\mathcal{Z}$ which we will call $\mathcal{Z}_1$ and which instead of corrupting P_i or P_j *before* any of them generated output, halts with a random bit as output. We claim that $\mathcal{Z}_1$ has exactly the same success probability in distinguishing real and ideal model as $\mathcal{Z}$ has. For proving this, note that $\mathcal{Z}$ has completely identical views in the real and the ideal model before either P_i or P_j generates output. But if either P_i or P_j gets corrupted *before* any of them generated output, by construction of $\mathcal{F}_{\text{KE}}^{(i,j)}$ and $\mathcal{S}_\pi^{(i,j)}$, the simulator may choose the output value for the respective other party and thereby simulate a protocol run exactly as in the real model. So as $\mathcal{Z}$ has completely identical views of the real and the ideal model in this case, it cannot do better than to toss a coin (thereby doing exactly what $\mathcal{Z}_1$ does). Hence, $\mathcal{Z}_1$ distinguishes real from ideal exactly as good as $\mathcal{Z}$ does. Moreover, for on the one hand, we assumed authenticated links (so the adversary is only allowed to block, but not to forge messages sent by uncorrupted parties), and on the other hand $\mathcal{Z}_1$ does not corrupt P_i or P_j while their protocol output is being fixed (i. e., before one of them actually outputs its key), $\mathcal{Z}_1$ is guaranteed matching outputs of P_i and P_j both in the real and the ideal model. (In the real model this is implied by assumption about $\pi^{(i,j)}$; in the ideal model it is clear by construction of $\mathcal{F}_{\text{KE}}^{(i,j)}$.)

Consider an environment $\mathcal{Z}_2$ which internally simulates environment $\mathcal{Z}_1$ and outputs whatever $\mathcal{Z}_1$ outputs. All communication of $\mathcal{Z}_1$ with the parties and the adversary is relayed, with the following exception: if the simulated $\mathcal{Z}_1$ wishes to ask the adversary to corrupt a party P_l (where $l \in \{i,j\}$) *after* the first party generated output κ, then $\mathcal{Z}_2$ answers $\mathcal{Z}_1$'s request on its own. It presents $\mathcal{Z}_1$ with a state of P_l that contains as local information *only* the key which was output by the first party, *except* when P_l did already generate output. By assumption about protocol $\pi^{(i,j)}$ and the simulator $\mathcal{S}_\pi^{(i,j)}$, this is exactly what $\mathcal{Z}_1$ would have got both in the real and in the ideal model. From this point on, $\mathcal{Z}_2$ ignores all output possibly generated by the (actually uncorrupted) party P_l. Messages the adversary is asked to deliver in the name of P_l are ignored unless for acknowledgment messages sent to the respective other party; in this case, $\mathcal{Z}_2$ pretends to $\mathcal{Z}_1$ that the receiving party generated output κ. As by assumption about protocol $\pi^{(i,j)}$, both P_i and P_j have already fixed their output (in our case to κ), $\mathcal{Z}_1$ will have identical views in the simulation inside $\mathcal{Z}_2$ and running "live" with parties and an adversary. Then $\mathcal{Z}_2$ has still exactly the same advantage in distinguishing the real model from the ideal one as $\mathcal{Z}_1$ and therefore $\mathcal{Z}$ has, even though $\mathcal{Z}_2$ corrupts neither P_i nor P_j at any point in time.

The simulator $\mathcal{S}_\pi^{(i,j)}$

$\mathcal{S}_\pi^{(i,j)}$ internally keeps a simulation of two parties $P_i^{(s)}$ and $P_j^{(s)}$ running protocol $\pi^{(i,j)}$.

- **Communication with $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$:**
  - Upon receiving from $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$ a forwarded message `(ready,`sid`)` with initial sender P_l (where $l \in \{i, j\}$), forward this message to $P_l^{(s)}$.
  - As soon as the first simulated party $P_l^{(s)}$, whose dummy counterpart P_l is *not* corrupted, produces output (which we will call κ here), store κ and send `(ready,`sid`)` to $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$.
  - Upon receiving `(choose-key,`sid`)` from $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$ (which by construction of the ideal functionality can only happen when $\mathcal{S}_\pi^{(i,j)}$ signalized through its ready signal that it has output κ available), send `(key,`sid`,`κ`)` back to $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$.
  - Deliver output messages sent from the ideal functionality to either P_i or P_j exactly when the respective simulated party $P_l^{(s)}$ generated output in the simulation.
- **Communication with $\mathcal{Z}$:**
  - When being requested by $\mathcal{Z}$ to check for messages sent by parties, reply that no messages were sent, except in the following case: when the dummy party P_i is uncorrupted and the simulated party $P_i^{(s)}$ wants to send a message m to $P_j^{(s)}$, then, upon being asked by the environment $\mathcal{Z}$, claim that P_i wants to send m to P_j but do not yet deliver m in the simulation (similarly for messages being sent from $P_j^{(s)}$ to $P_i^{(s)}$ when P_j is uncorrupted).
  - When being asked by $\mathcal{Z}$ to deliver a message m to some uncorrupted party P_l, store this request for future use and additionally, if $l \in \{i, j\}$, deliver m to $P_l^{(s)}$ in the simulation.
  - When being told by $\mathcal{Z}$ to corrupt some party P_l, first corrupt the dummy party P_l; then prepare state information for P_l taking into account all messages $\mathcal{S}_\pi^{(i,j)}$ was asked to deliver to P_l. Additionally, if $l \in \{i, j\}$ and P_l has not done so before, let P_l send a ready signal to the ideal functionality as soon as the corresponding session ID *sid* becomes available to $\mathcal{S}$. Moreover, if $l \in \{i, j\}$, proceed as follows:
    (a) If $\mathcal{S}_\pi^{(i,j)}$ has not yet sent its ready signal to the ideal functionality (i. e., if no simulated party $P_t^{(s)}$, whose counterpart P_t is uncorrupted, produced output), then deliver $P_l^{(s)}$'s internal state as P_l's internal state.
    (b) If $\mathcal{S}_\pi^{(i,j)}$ sent its ready signal and therefore, the ideal functionality sent output κ to both P_i and P_j, then P_l's state must be prepared to include κ *if and only if* it was not output by P_l. Then, κ can be obtained by $\mathcal{S}_\pi^{(i,j)}$ through delivering κ from the ideal functionality to P_l. So in any case, prepare state information for P_l consistent with κ. (This is possible by assumption about protocol $\pi^{(i,j)}$.)

Fig. 9. The simulator $\mathcal{S}_\pi^{(i,j)}$

We have just shown that we may restrict to environments not corrupting *any* party. For such an environment, $\mathcal{S}_\pi^{(i,j)}$ simulates the communication of a complete protocol run of $\pi^{(i,j)}$ in the ideal model exactly as it would happen in the real model. Hence the *only* difference between real and ideal model is the value κ output as a common key by both P_i and P_j. In the ideal model, this value is a random k-bit string which in general "does not fit" to the simulated protocol run of $\pi^{(i,j)}$; however as $\pi^{(i,j)}$ is a *universally composable key exchange protocol* and therefore has output computationally indistinguishable from random k-bit strings, there can be no environment $\mathcal{Z}$ distinguishing the real from the ideal model. $\square$

Proof (of Proposition 3). For any hybrid-model adversary $\mathcal{H}$, we describe a simulator $\mathcal{S} = \mathcal{S}_\mathcal{H}$ mimicking attacks carried out by $\mathcal{H}$ in the hybrid model. $\mathcal{S}$ internally runs a simulation of a complete run of $\textsc{pad}^{(i,j)}$ in the hybrid model, including parties $P_1^{(s)}, \ldots, P_n^{(s)}$, the adversary $\mathcal{H}$, and (as needed) instances of the ideal functionality $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$. Yet all communication of the simulated $\mathcal{H}$ with the environment is forwarded to the (non-simulated) environment $\mathcal{Z}$ with which $\mathcal{S}$ is to interact. That means, incoming messages from $\mathcal{Z}$ are forwarded to $\mathcal{H}$ and vice versa.

The idea is to give $\mathcal{Z}$ a complete view of a hybrid-model run with adversary $\mathcal{H}$. By construction, the described simulation already does this job well, with two exceptions: by definition, $\mathcal{Z}$ is informed about corruptions as they take place. The solution to this issue is easy: every time $\mathcal{H}$ corrupts a party $P_l^{(s)}$ in the simulation, $\mathcal{S}$ first corrupts the corresponding party P_l in the ideal model. The state of P_l with which $\mathcal{Z}$ possibly expects to be supplied is then delivered by the simulated $\mathcal{H}$. Furthermore, if P_l has not yet sent its ready signal to the ideal functionality, $\mathcal{S}$ lets P_l do that as soon as the corresponding session ID *sid* becomes available to $\mathcal{S}$. (This is to allow the ideal functionality to generate output even in face of corrupted parties which did not yet send a ready signal.)

The other thing which has to be taken care of is the communication of $\mathcal{Z}$ with the parties P_l in the ideal model. The messages sent (as input) to and received (as output) from these parties by $\mathcal{Z}$ have to be consistent with the view of the hybrid-model execution by $\mathcal{Z}$. This is a little more involved and we describe our solution in detail.

In protocol $\textsc{pad}^{(i,j)}$, only the respective first message of the form $(sid,\texttt{ready})$ to P_i or P_j has some effect, all other messages are ignored. Yet by construction of $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$, $\mathcal{S}$ is informed about exactly these inputs as they arrive. So when $\mathcal{S}$ is informed about a message $(sid,\texttt{ready})$ which P_l ($l \in \{i,j\}$) got as input, it immediately forwards this as input to the simulated party $P_l^{(s)}$.

It remains to take care of the *output* behavior of the parties. We describe the necessary modifications:

- When the simulated party $P_j^{(s)}$ sends a message (sid,ψ) to $P_i^{(s)}$ at a time $P_i^{(s)}$, but *not* $P_j^{(s)}$ is corrupted, then $\mathcal{S}$ temporarily stops the simulation, sends $(sid,\texttt{ready})$ to $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$ and delivers to P_i (i. e., to itself, since P_i is

corrupted) the key κ which is sent from $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$ to P_i. It then sets $\bar{\psi} = \kappa \oplus \bar{\kappa}$ (where $\bar{\kappa}$ is the key $P_j^{(s)}$ received from $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$) and modifies the internal state of $P_j^{(s)}$ so to hold only the secret key κ from $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$ instead of $\bar{\kappa} \oplus \psi$.
Consequently, the corresponding message (sid,ψ) is then altered to read $(\mathit{sid},\bar{\psi})$ and the simulation is continued. Note that $\bar{\psi}$ as a random variable has uniform distribution over $\{0,1\}^k$ because κ has and is, as a random variable, independent of $\bar{\kappa}$ (even when $\mathcal{H}$ chooses $\bar{\kappa}$); that means that the pad $\bar{\psi}$ "looks" exactly as if generated by $P_j^{(s)}$ itself and thus this does not disturb the authenticity of the hybrid-model simulation.

- When an uncorrupted simulated party (say, $P_l^{(s)}$, where $l \in \{i,j\}$) generates output, then $\mathcal{S}$ has to deliver the corresponding output message containing the common key from $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$ to P_l. If the key has not yet been determined, $\mathcal{S}$ first sends $(\mathit{sid},\mathtt{ready})$ to $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$. (When P_j is corrupted at that time, $\mathcal{S}$ is additionally asked for the common key. It then replies with the key $P_l^{(s)}$ produced.)
- The one case in which the output of the simulated $\textsc{pad}^{(i,j)}$ still differs from that in the ideal model is the case in which at the time the message (sid,ψ) is delivered from $P_j^{(s)}$ to $P_i^{(s)}$, neither of them is corrupted.
This is no problem when nobody gets corrupted later, since then $\mathcal{H}$ has no information about the key agreed upon in the simulation. On the other hand, we have to take that into consideration upon later corruptions of $P_i^{(s)}$ or $P_j^{(s)}$. Namely, if in the situation in question a later corruption of $P_i^{(s)}$ is requested by $\mathcal{H}$, and the message just mentioned is *not* yet delivered, the key $\bar{\kappa}$ delivered from an instance of $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$ has to be replaced by $\kappa \oplus \psi$ in $P_i^{(s)}$'s memory; here, κ denotes the key generated by $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$ in the ideal model. Upon a later corruption of $P_i^{(s)}$ or one of $P_j^{(s)}$, we only have to replace the key they locally hold as output by κ. (Note that κ is accessible to $\mathcal{S}$ upon corruption of the corresponding dummy party P_l if P_l did not yet generate output.)
Again, if the key κ generated in the ideal model is not yet determined, $\mathcal{S}$ first corrupts P_l (where $P_l^{(s)}$ is the respective party to be corrupted in the simulation), then sends $(\mathit{sid},\mathtt{ready})$ to $\mathcal{F}_{\mathrm{KE+}}^{(i,j)}$, possibly chooses the common key (in that case $\mathcal{S}$ chooses it at random), and delivers to P_l (i. e., to itself) the common key just generated. Furthermore, since no information about the key $\bar{\kappa}$ is revealed to $\mathcal{H}$ when neither $P_i^{(s)}$ nor $P_j^{(s)}$ is corrupted, the above modifications still yield a "valid" view of a protocol execution in the hybrid model to $\mathcal{H}$ and therefore to $\mathcal{Z}$. (Note that the value $\kappa \oplus \psi$ replaced for $\mathcal{F}_{\mathrm{KE}}^{(i,j)}$'s output in the hybrid model has uniform output distribution, as in our case, ψ is picked uniformly and in particular independent of κ, for P_j is by assumption not corrupted when picking ψ.)

By the above discussion, in any case, $\mathcal{S}$ provides $\mathcal{Z}$ with an "authentic-looking", complete view of the hybrid model. $\square$