No-Match Attacks and Robust Partnering Definitions – Defining Trivial Attacks for Security Protocols is Not Trivial

Yong Li
Huawei Technologies Düsseldorf
Germany
yong.li1@huawei.com

Sven Schäge
Ruhr-Universität Bochum
Germany
sven.schaege@rub.de

ABSTRACT

An essential cornerstone of the definition of security for key exchange protocols is the notion of partnering. The de-facto standard definition of partnering is that of (partial) matching conversations (MC), which essentially states that two processes are partnered if every message sent by the first is actually received by the second and vice versa. We show that proving security under MC-based definitions is error-prone. To this end, we introduce *no-match attacks*, a new class of attacks that renders many existing security proofs invalid. We show that no-match attacks are often hard to avoid in MC-based security definitions without a) modifications of the original protocol or b) resorting to the use of cryptographic primitives with special properties. Finally, we show several ways to thwart no-match attacks. Most notably and as one of our major contributions, we provide a conceptually new definition of partnering that circumvents the problems of a MC-based partnering notion while preserving all its advantages. Our new notion of partnering not only makes security definitions for key exchange model practice much more closely. In contrast to many other security notions of key exchange it also adheres to the high standards of good cryptographic definitions: it is general, supports cryptographic intuition, allows for efficient falsification, and provides a fundamental composition property that MC-based notions lack.

CCS CONCEPTS

• **Security and privacy → Security requirements**; **Security protocols**;

KEYWORDS

protocols, definitions, partnering, no-match, original key

1 INTRODUCTION

Authenticated key exchange (AKE) protocols are among the most important building blocks of secure network protocols. Intuitively, they allow a party A (Alice) to authenticate a communication partner B (Bob) and securely establish a common session key with B (and vice versa). Most security models for AKE try to model that an adversary can observe or initiate several executions of the protocol at the same party. Usually, each party is therefore modeled as having several oracles – processes that all have access to the same long-term key of that party but otherwise independently execute the protocol. The combination of two oracles that communicate with each other according to the protocol specification defines a (protocol) session.[1]

Informally, the basic requirements for security are that if an oracle of A communicates with one of B, no adversary can distinguish the session key computed by A's oracle from a random value without corrupting A's or B's secret values (i.e. their long-term keys, secret session states, or directly their session keys). This property is commonly referred to as *key indistinguishability* while (in this context) A's oracle is usually termed the Test-*oracle* – technically, the adversary sends a special Test-query to Alice's oracle and receives back the real session key of that oracle or a random key.

In the security game, the adversary can choose the Test-oracle among the set of all oracles that have computed a session key so-far while having access to generous queries that grant her access to the secret states of oracles of unrelated sessions. The important overall rationale behind the security model is that in a concurrent setting, leakage of session-dependent parameters should not compromise the security of unrelated sessions.

For the definition of security, the precise specification of what defines related – or more commonly called *partnered* – oracles is of utmost importance (and all our examples below provide convincing evidence for this). This is because all security models for key exchange allow the adversary to request the session keys of unrelated oracles (i.e. not partnered to the Test-oracle) only, usually via a so-called Reveal query. The rationale behind this restriction is that oracles partnered to the Test-oracle are assumed to share the same session key. And of course, the revealed session key of the partner oracle could directly be used to break key indistinguishability. This makes the notion of partnering an essential tool for defining trivial attacks in key exchange protocols.

In general, the adversary may also ask for the secret long-term keys of all parties different from A and B by sending a Corrupt-query to one of their oracles. However, in stronger formalizations of security for key exchange protocols [15, 29, 35–37], for example

[1] Note that at this point we make use of the terminology of the original Bellare-Rogaway model [6]. Canetti-Krawczyk like models use a different terminology [17], in particular they use the term session to refer what BR-like models call oracle.

CCS '17, October 30-November 3, 2017, Dallas, TX, USA

ACM ISBN 978-1-4503-4946-8/17/10…$15.00
https://doi.org/10.1145/3133956.3134006

when also modeling *key compromise impersonation* (KCI) attacks or *perfect forward secrecy* (PFS), a Corrupt-query may also be queried to the oracles of A and B (even to the Test-oracle).

Finally, some security models like [37] also allow the adversary to obtain the ephemeral secret keys that are computed by the oracles in the protocol run. This is usually realized via a so-called EphemeralKey-query (or a RevealState-query like in [31]).

Existing Definitions of Partnering and Session Identifiers. The de-facto standard definition for partnering is based on the notion of *(partial) matching conversations*. Essentially, two oracles have matching conversations if every message[2] that has been sent by the first oracle has actually been received by the second (without any modifications on transit) and vice versa. Matching conversations are explicitly used in many widespread security models like the original Bellare-Rogaway model [6], the extended Canetti-Krawczyk eCK model by LaMacchia, Lauter, and Mityagin [37], the CK+ model that was introduced by Krawczyk in the analysis of HMQV [35], and the recent security model for authenticated and confidential channel establishment (ACCE) by Jager, Kohlar, Schäge, and Schwenk [29].[3] *Partial* matching conversations are defined in the same way as matching conversations except that they do not cover the entire message transcript but only a part of it. Which messages exactly belong to the relevant partial matching conversation is highly protocol-dependent. Using partial matching conversations as opposed to plain matching conversations has several conceptual disadvantages, the most important being its *lack of general applicability*. In particular, while matching conversations can be used to formalize trivial attacks for any key exchange protocol, the definition of partial matching conversations is protocol-specific. Strictly speaking, this makes the entire security definition protocol-specific as well and comparisons among security protocols much harder.

Another common partnering concept relies on so-called *session identifiers*. Essentially, it requires that each oracle is assigned with a string that is referred to as its session identifier. Moreover, it postulates that oracles are partnered if and only if they share the same session identifier. Session identifiers *can* be based on (partial) matching conversations (and often are). To this end, one can define that the session identifier of an oracle consists of the list of all messages sent and received by that oracle. In this way, two oracles have the same session identifier (are partnered) iff they have matching conversations. However, session identifiers are much more general as they can also include, besides (partial) transcripts, other values including (secret) state values held by the oracles and parties. Protocol analyses that use this concept need to first define the session identifier concretely. This definition is usually highly protocol-specific.

In the literature, one alternative to session identifiers based on protocol information is to use session identifiers that are generated prior to the protocol execution, so-called *external* session identifiers[4]. Protocols for computing such session identifiers usually consist of an exchange of random nonces among the parties, as for example in [3]. However, despite being general this definition is problematic in many scenarios. Most importantly, it does not model cryptographic practice; for efficiency reasons key exchange protocols usually do not allow for an extra phase that is used for the sole purpose of deriving session identifiers.

Finally, there is another way of formalizing that two oracles are partnered which was introduced by Bellare and Rogaway in 1995 [7]. It simply assumes the existence of a so-called partnering function. However, as argued by Bellare, Pointcheval, and Rogaway in [5], the definition allows for unintuitive partnering functions. Therefore partnering functions are not widespread in the cryptographic literature.

In the literature, one can also find other partnering definitions which, however, are not widely used. One example is a definition proposed by Kobara, Shin, Strefler [33] in 2009. Essentially, the definition requires that two oracles are partnered if they have the same key and no other (third) oracle shares that same key. This definition has the highly counter-intuitive[5] feature that whether two oracles are partnered with each other is dependent on the existence or non-existence of some other oracle.

Desirable Properties of Partnering Definitions. Some of the existing partnering concepts have grave conceptual disadvantages. Essentially, they do not provide an appropriate level of abstraction that is suitable for a cryptographic analysis as we would like to argue in more detail.

In particular, most of the concepts are too general and require protocol-dependent concretizations (partial matching conversations, session identifiers, partnering functions). These concretizations often differ considerably. This has several drawbacks.

1) In key exchange protocols all definitions of partnering are used to define the same intuitive security property, key indistinguishability. This is different from for example security definitions of public key encryption where chosen ciphertext security, for example, refers to cryptographic systems which offer considerably more security guarantees than chosen plaintext secure ones. In light of this, it seems rather unnatural that key exchange protocols may have different security definitions although they aim to meet the same security requirements.

2) One of the most crucial problems of partnering definitions that need to be concretized protocol-wisely is that for each concretization the actual overall security of the protocol may vary greatly (although the rest of the security model is kept fixed). This is because the concrete formulation of partnering has a critical effect on the actual security guarantees provided by a provably secure protocol. For example, one can easily propose partnering definitions under which practically weak or even insecure protocols admit a proof of security. It is highly counter-intuitive and unsound to deem such a protocol as secure as protocols under traditional, strong notions of partnering. In Section 4.3 we provide a brief example.

[2]To achieve a general definition that is applicable to any key exchange protocol, messages are always interpreted as binary strings in the natural way.

[3]We remark that although the concrete technical formalizations of matching conversations given in these works may differ, they essentially model the same property.

[4]We note that the terminology is ambiguous. In fact some papers use 'external' session identifiers just as place-holders for any fitting definition of session identifiers. In these context, external session identifiers do not have to be pre-specified. The advantage is that one can refer to protocols in general and independent of their specification of session identifiers. We note that, typically, these papers use matching conversations as the default instantiations of session identifiers in practical settings.

[5]Support of cryptographic intuition is a highly valued feature of definitions in cryptography as [5] showed when deciding against the use of the notion in [7].

3) Fixing the security definition (and in particular the partnering definition) allows to treat key exchange protocols in an abstract way. In this way they can be used as generic building blocks (in a black-box way) in more complex protocols and give rise to new constructions. Protocol-specific definitions of partnering violate this approach.

4) Moreover, we would like to stress that it is problematic to not specify a single concrete partnering definition at all in the security model. In particular, there are several theoretical and conceptual disadvantages to simply requiring that there exists some appropriate, concrete partnering definition. Crucially, such an approach does not allow for *simple falsification* of the protocol's security. More concretely, an attack on a protocol does only violate a security proof if it actually holds for *all* partnering definitions, because the proof only states that there exists an appropriate partnering notion. Such an approach is highly impractical and violates cryptographic intuition. A security proof should always fix all circumstances it holds under – ultimately to allow for a simple verification of when it does not hold. Also it shifts the amount of work from the security proof to the attacker. In classical models it is rather simple to describe a successful attack because an attack only has to violate a single concrete security definition. For unspecified partnering definitions, an attack has to show (or rather to prove) that it is valid against all possible (even highly unreasonable) definitions of partnering. Showing this is usually much more difficult than specifying an attack against a fixed definition. We stress that similar problems still occur also when the security definition is slightly more concrete, for example if it requires that there exists a suitable partial matching transcript. Similar to before, to rule out the security of a protocol one has to show it for all possible definitions. Finally, simply requiring that an appropriate partnering definition exists in essence amounts to introducing a primitive-specific additional assumption (that one would rather prefer to avoid). In contrast to classical security reduction this is not an assumption that helps to deduce the security of a cryptographic scheme but rather an assumption to argue on the realizability of the security notion. We do not know anything about the plausibility of such an assumption in general.

5) There are even more (and similarly important) benefits when settling on a fixed partnering definition than the comparability of protocols: importantly, it allows for better comparability of security models. With a fixed partnering definition we may much more easily deduce implications between different security models. This allows to reveal fundamental connections between important cryptographic models and pave the way for improved security definitions.

6) Finally, defining and relying for each new protocol on a specialized security definition (that is specifically crafted for that protocol) is arguably more error-prone than using a general definition that is verified once and for all.

Considering these arguments, it is worthwhile to restrict the class of concrete partnering definitions to allow for a broad comparability of protocols and models among each other and for simple falsification. In fact these features are the prime benefits of matching conversations and the reason for its widespread adoption. However, on the other hand matching conversations are too restrictive. In particular, they deem protocols insecure which cryptographic intuition would not consider problematic at all. More concretely, (partial) matching conversations suffer from a lack of the following properties.

7) A partnering definition as part of a cryptographic security definition should be abstract and applicable to all 'reasonable' instantiations of the considered primitive/protocol class. A concrete partnering definition that is based on a specific partial transcript of some protocol however can in general not be used for the analysis of other protocols.

8) An important property a partnering definition (or more generally a security model) should provide, is a basic form of composability. More precisely, it should for example support cryptographers' best practice to first extract the cryptographic core of real-world protocols and then analyze that core in a formal model. Importantly, there should be no obstacle for the obtained results to be meaningful for the real-world implementation. Matching conversations do not support this approach as the validity of any result is formally dependent on the existence of unrelated administrative information (as detailed in Section 4.5 and Appendix C).

9) A partnering definition should support cryptographic intuition. In particular, it is highly counter-intuitive that the overall security of some cryptographic scheme is formally violated if entirely unrelated (random) messages are added to the message flow of a secure protocol.

Taking stock, matching conversations excel in providing the first set of desirable properties, 1)–6), while failing to support the second, 7)–9). In contrast, the more general notions of partnering like partial matching conversations, general session identifiers, and partnering functions may be concretized such that they provide the second set of properties, 7)–9). However, they usually do not provide the features of the first set of properties, 1)–6).

1.1 Contribution

We provide several contributions. First, we present a new class of attacks on key exchange protocols called *no-match attacks*. In a no-match attack the attacker slightly modifies the message exchanged between two oracles on transit such that the keys computed by these oracles remain equal. However, the decisive point is that due to the introduced modifications the oracles are not partnered. In all existing security models, this can be exploited to break the security by simply revealing the session key of one of the oracles to answer the Test-query for the other. We use no-match attacks to show that providing sound security analyses for key exchange protocols under MC-based definitions of partnering is difficult and error-prone. In particular, we show for a diverse collection of security protocols, that the corresponding security proofs are flawed. To this end, we present detailed descriptions of successful adversaries. The set of protocols that we consider is not complete and we believe many more security proofs to be flawed or at least incomplete in the sense that they do not formally rule out no-match attacks (although the protocol actually protects against them). We present three ways to thwart no-match attacks. Our solutions vary in the additional complexity added to the protocol and the additional theoretical assumptions necessary in the security proof as compared to the original protocol. This allows to choose the necessary modifications to the protocol (for making it provably secure) in a way that suits

the application it is used in. As our main solution and as one of our major contributions, we present a new definition of partnering that precisely renders no-match attacks invalid. We provide several arguments for why our notion is better than previous notions of partnering. First, our definition can generically be plugged into all existing security definitions. Second, it does not depend on the analyzed protocol and thus is very general. In fact, it is *independent* of the concrete messages that are exchanged between the parties. Third, it can be verified very easily, i.e. whether two oracles are partnered is decided solely by checking a single cryptographic value. Forth, it supports efficient falsification making it possible to decide whether for a protocol we cannot provide a proof of security at all. Fifth, and in contrast to some of the existing definitions it formalizes and supports cryptographic intuition. Finally, we stress that our new definition of partnering supports the form of composability that the MC-based notions provably lack and even a slightly stronger form of it. In particular, protocols proven secure under our definitions remain secure even if additional messages are exchanged that do depend on the public values of the protocol. As our last result, we show how to deal with no-match attacks in security models that also, besides key indistinguishability, formalize explicit authentication. In particular we provide a new notion of explicit authentication that similar to before renders no-match attacks useless. Advantageously, this new definition again supports the basic from of composability that MC-based notions lack. As a stepping stone and of independent interest, we essentially decouple authentication from the used partnering definition.

1.2 Related Work

The importance of precise security definitions for rigorous cryptographic analysis is well-recognized by cryptographers [25, 32] and can hardly be overestimated. Ultimately, one first needs to provide a precise definition of security before constructing a scheme that provably achieves it. No-match attacks show that it is hard to precisely define trivial attacks for key exchange protocols. Similar observations have been made for other primitives before. Most notably, Bellare, Hofheinz, and Kiltz showed that for public key encryption schemes there exists several non-equivalent variants for formalizing that the challenge ciphertext may not be asked to the decryption oracle [4]. As a result of their analysis, they isolate the strongest of the existing definitions and recommend it for future usage. Our results are similar in the sense that they also analyze different ways to define trivial attacks for an important cryptographic primitive, but they go much further. We do not only show that the existing ways of formalizing trivial attacks are not equivalent. We present new, subtle attacks that reveal flaws in formal security proofs which have been overlooked by cryptographers for many years. Moreover, we show that the most popular definition of trivial attacks – via matching conversations – is actually unsuited for cryptographic practice. Finally, we provide a new definition that solves many of the problems of the existing definitions (lack of composability, unrealistic no-match attacks) while keeping important advantages.

There exist few papers that consider attacks that are related to the definition of partnering in key exchange protocols. Most noteworthy, in 2005, Choo and Hitchcock [18] presented an attack on the Jeong-Katz-Lee protocol TS2 [30] and on the three-party protocol 3PKD of Bellare and Rogaway [7]. In retrospect and using our terminology, we can interpret the attack on the 3PKD protocol as a simple no-match attack (without advice). In 2007, Bresson, Manulis, and Schwenk [13] showed that under the original security definition for group key exchange by Bresson, Chevassut, Pointcheval, and Quisquater [12] there exist protocols where the adversary may impersonate users or where two partnered oracles accept with a different key. However, the authors explicitly state that these problems only appear in the group setting with at least three parties. The authors of [29] were pointed to a subtle problem with the defintion of partnering that, in retrospect, can be interpreted as a no-match attack (see the most recent full version [40]). This issue was also observed by [16]. We stress that the above works only consider attacks on single protocols. In contrast, our work provides a very general attack strategy that can be applied on a variety of protocols. We also introduce no-match attacks with advice showing that no-match attacks can be much harder to avoid (and more subtle) when the adversary is granted access to strong queries, like in KCI attacks or when modeling forward secrecy.

In 2011, Cremers, provided an in-depth analysis and comparison of three of the most wide-spread security models, the CK, the CK+ (or HMQV), and the eCK model [19]. As a result, he was able to show that these models are formally and practically incomparable to each other, meaning that for every pair of models one can find protocols which are secure in the first model but not in the second (and vice versa). Crucially, he exploited that the security models have different ways to formalize partnering and, in particular, different ways of defining security for non-symmetric protocols. Moreover, Cremers showed that the proofs of several protocols are not complete. In particular, he showed that in some models protocols may have computed distinct keys although they formally are partnered.[6] Our no-match attacks are rather orthogonal to this, where oracles have the same key but do not have matching conversations. This not only makes proofs incomplete but also allows us to describe concrete and often very subtle (no-match) attacks. However, there is also a more fundamental difference between our work and that of Cremers. Like some works before, Cremers focuses on and suggests to use definitions of partnering that guarantee that two oracles have computed the same key iff they are partnered. Jumping ahead we show that this is actually not an appropriate way to deal with no-match attacks and therefore we propose a distinct and conceptually new definition of partnering. Crucially, and in contrast to all other works that we are aware of, our definition of partnering does not only require that two (or more) oracles have computed the same key but that this key must additionally be equal to a special (ideal) key which only depends on the two oracles but not the attacker. Moreover, our results are not restricted to certain security models but reveal a general problem (and possible solutions to it) when defining security in the realm of key exchange and authentication protocols.

[6] Intuitively, one can view this as a violation of correctness (in the presence of an active attacker). However, since both oracles have computed distinct keys they will not be able to exchange authenticated messages with that key. This will already make the parties aware of a problem that has occurred in the key derivation phase. (Arguably, this argumentation is central to the concept of implicit authentication.) Our no-match attacks in contrast will make an attacker successfully break the secrecy of the communication without any indication of its presence to the oracles.

The authors of [33] also introduce a new partnering definition that aims to fix the problem that occurs "if the partnering definition takes information into account that is irrelevant to the computation of the session key". This notion is incomparable to ours and all other existing notions and, as sketched before, very counter-intuitive. In essence, it defines that the fact that two oracles are partnered (or not) depends on the (non-)existence of another oracle with some related properties. (All other notions of partnering concentrate on a relation between only two oracles.) More concretely, the definition in [33] states that two oracles are partnered if they have the same key and no other oracle has that key. We also remark that in general concentrating only on "irrelevant" information is not sufficient. Jumping slightly ahead, our results show that no-match attacks are also possible by modifying messages that are crucial to the computation of the message (also see Remark 1).

1.3 Open Problems

In this paper, we provide a collection of (flawed) security analyses that show that the corresponding protocols fail to meet their stated security claims. We stress that our list is by no means complete. We remark that we concentrate on protocols that are proven secure in widespread security models. Given the popular use of session identifiers based on matching conversations and the wide popularity of the original BR, CK+, eCK, and ACCE model, we believe that many other security proofs are vulnerable to no-match attacks. Furthermore, it remains an interesting open problem to more practically exploit a no-match attack in a real-world protocol but we have little hope in this regard. Finally, in Section 7 we consider potential future relaxations of our new security notion and conceptual and technical obstacles towards them.

1.4 A Note on Formalism

To focus on the subtle[7] core problem surrounding no-match attacks we try to keep our results as light on key exchange formalism as possible. More rigorously, technical definitions can be found in the cited security models. A very brief overview on key exchange models is given in Appendix A. We stress that in several ways our results do not rely on concrete formalizations of security definitions. For example, for our purposes it does not matter if we use the definition of matching conversations given [6], in [29], or the one given implicitly in [35]. For our purposes these definitions can be regarded as equivalent, all essentially trying to capture that everything that is sent by one oracle is actually received by the second and vice versa. It is this conceptual core of the definition our result relies on, not a concrete formalism of it. We therefore sometimes refrain from providing a concrete definition in this paper and basing our reasoning on it but refer to the literature for concrete formalizations.

1.5 Overview

We start with a gentle introduction to no-match attacks in Section 2. In Section 3, we present a diverse list of no-match attacks against existing security protocols. These attacks reveal subtle flaws in the security proofs of the protocols. A detailed, step-by-step example of such an attack can be found in Appendix B. Altogether we provide three solutions to deal with no-match attacks. In Section 4, we present our first solution and main contribution: a new partnering notion that does not require any modification of the protocol implementation at all to deal with no-match attacks. We extend our results in Section 5 to provide a new definition of explicit authentication for key exchange that helps to deal with no-match attacks in security models with explicit authentication. In Section 6, we show how to use our new security notion as a more general tool to strengthen existing security definitions. Finally, Section 7 contains an outlook on how our new security notion might be extended further. Additionally, we present several arguments why such an extension may indeed be rather difficult to achieve when it adheres to the high standards that we require (and which are so common in most other fields of cryptography). In Appendix C we show in a formal way that security models based on matching conversations cannot be composed with arbitrary random messages even if those messages are independent of the protocol. Next we show that, in contrast, our new notion of partnering does provably fulfill this and even a stronger form of composition (where the additional messages may even be generated from the public values of the protocol). Finally, in Appendix E we describe two alternative approaches (and their limitations) to deal with no-match attacks in existing security models. These solutions either rely on a careful instantiation of the primitives or on a (slight) modification of the protocol layout.

2 NO-MATCH ATTACKS

In this section we introduce no-match attacks. To precisely capture no-match attacks, we first introduce the notion of *original keys*. It will also be central to the new partnering definitions that we provide later on.

Definition 2.1 (Original Key). The original key of a pair of communicating oracles is the session key that is computed by each of the oracles in a protocol run which is executed in the presence of an entirely passive adversary (which only relays the messages sent by the oracles).

We remark that the original key of two oracles depends on the (secret) randomness available to the two oracles. We are now able to introduce no-match attacks.

Definition 2.2 (No-Match Attack). We say, an adversary has successfully launched a no-match attack if there is a pair of oracles such that each of them has computed their original key but the two oracles are not partnered (do not have the same session identifier).

We clarify that in the above definition we essentially consider two runs of the protocol although under different conditions. The first one is only imaginary and used to define the original key of the two oracles while assuming the adversary remains passive. In contrast, the real second run assumes an active adversary. We stress that in both runs, the oracles use the same randomness.

We also remark that defining no-match attacks relative to the original key is crucial (also see Figure 1). In particular, it is not sufficient that an adversary launches an attack that makes the two oracles accept merely with the same key, as some papers have suggested before [5]. For a no-match attack this must additionally be the original key. This distinction will be important in Section 4

[7] Recall that despite being peer-reviewed and public, several existing protocols are still susceptible to no-match attacks.

(and in particular in Section 4.3) when we propose several solutions to thwart protocols against no-match attacks. It is this (subtle) feature that solves many of the problems of previous definitions.

Finally, we emphasize that we have defined no-match attacks with a general partnering definition. Subsequently, we concentrate on partnering definitions based on (partial) matching conversations (including session identifiers based on matching conversations).

2.1 Warm-Up: No-Match Attacks without Advice

We start with an instructive example of a no-match attack. Imagine a key exchange protocol, where in the last protocol step, (an oracle of) Alice sends a message together with a signature on that message to (an oracle of) Bob. As a running example used here and later on we may more concretely use signed Diffie-Hellman (i.e. the basic Diffie-Hellman key exchange where each DH share is accompanied by a signature over that share and the identities of the communication parties). Recall that the standard security notion of digital signatures requires the adversary to output a forgery on a message that has not been queried before [27]. Also, assume that Alice uses a probabilistic signing algorithm. Now let us describe an attacker $\mathcal{A}$ that cannot be reduced to the security of the digital signature scheme. The attack proceeds in three steps:

(1) $\mathcal{A}$ intercepts Alice's last message on transit.
(2) $\mathcal{A}$ computes a new signature on the last message, for example by first brute-forcing Alice's secret key and then re-applying the signature algorithm.
(3) Finally, $\mathcal{A}$ sends the message and the new signature to Bob.

In many existing security protocols, only the last message is used in the derivation of the session key – but not the signature over that message. This crucially influences security. Observe, that in presence of the above attack the communication transcripts computed by Alice and Bob are very similar. The only difference is that the last signature is different from the one produced by Alice. At the same time, Alice cannot recognize the modification introduced by $\mathcal{A}$, because a) there is no further message from Bob to Alice (in which he could inform her of the modified signature he received) and b) the signature is not used to derive the session key (so the fact that the signatures are distinct will not be reflected in the two parties computing distinct keys). Now, any security proof in a model based on matching conversations is deemed to fail. In essence, the problem is that the above attack cannot be ruled out via a reductionist argument to the security of the digital signature scheme. Exchanging a signature with a new one on the *same* message does not violate the standard security definition of digital signatures. As a result, Alice and Bob still share the same session key after this attack but they do not have matching conversations anymore. Next, the attacker asks a Reveal-query to Bob's oracle to obtain the secret session key. Finally, $\mathcal{A}$ uses that key to successfully answer the Test-query to Alice's oracle.

The straight-forward solution is to require the signature scheme to be strongly-secure, meaning that in the signature security experiment the adversary is also allowed to output a new signature on a message that has already been queried to the signature oracle. In this way, $\mathcal{A}$ can directly be used to break the security of the signature scheme and the proof can go through.

We note that in the protocol the ephemeral Diffie-Hellman keys are well-protected against adversarial modifications by the digital signatures. It is rather the digital signatures themselves that are not appropriately secured against altering.

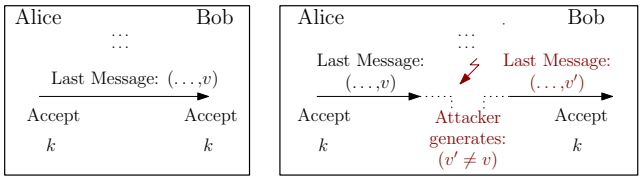

Figure 1: Protocol Execution in the Presence of a Passive Adversary (left) and under a No-Match Attack (right).

2.2 No-Match Attacks with Advice

The above attack only serves as an introductory example to no-match attacks and the problem they pose in formal security analyses. In the sequel, we will present a new class of (more subtle) no-match attacks which to the best of our knowledge have not been considered before. The crucial difference to the above example is that the adversary will always be able to efficiently obtain the key or some other valuable information of the cryptographic primitive (i.e. the signature scheme in our example) via the queries granted in the security definition. We generally refer to this information as advice. Correspondingly, we call these attacks 'no-match attacks with advice' as opposed to the above 'no-match attacks without advice'. Interestingly, our work shows that for some primitives no-match attacks with advice are much harder to protect against than no-match attacks without advice. For example, in our running example it turns out that requiring strong security is not sufficient to fix the security proof.

For the introduction of no-match attacks with advice we consider the same setting as before. Instead of a concrete signature we more generally consider a cryptographic value v. Again the general attack proceeds in three steps:

(1) $\mathcal{A}$ intercepts the last message. This message contains a cryptographic value v.
(2) The adversary uses the queries granted in the security game to *efficiently* compute a distinct cryptographic value $v' \neq v$ that makes Bob compute the same key as when using v.
(3) Finally, $\mathcal{A}$ replaces v with v' in the last message and sends it to Bob.

The difference between no-match attacks with and without advice is in Step 2 of the general attack pattern. Roughly, in a no-match attack with advice the adversary may, solely by the use of her access to the attack queries of the key exchange experiment, obtain secret information that allows her to compute v' efficiently whereas in no-match attacks without advice no such information is available. This information can include the secret signing key of a signature scheme, cryptographic message authentication code, or public key encryption scheme, or the plaintext message in case v is a PKE-ciphertext on some secret message.

This new type of no-match attacks is often applicable to protocols that are shown to provide security against key compromise

impersonation (KCI), exposure of the session states (e.g. ephemeral secret keys or intermediate values), or (full) perfect forward secrecy (PFS). In these scenarios the adversary is additionally provided with at least one long-term key or with secret session state information, either directly of the Test-oracle or of the oracle it communicates[8] with. Note that many widely-used models [14, 29, 35–37] allow the adversary to perform these actions.

In the following, for simplicity of exposition and due to their high relevance to existing security analyses, we concentrate on no-match attacks on the last protocol message. We emphasize that no-match attacks may also be launched against earlier protocol messages (as exemplified in Appendix B).

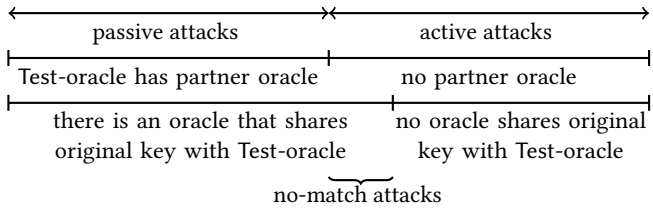

Figure 2: Attack spectrum and corresponding states of the Test**-oracle when relying on partnering definitions based on matching conversations.**

2.3 Exploiting No-Match Attacks and Scope

Finding a no-match attack on some protocol in some security model does not per se lead to a successful attack. To turn a no-match into a successful attack on the protocol we additionally need that the security model is exploitable.

Definition 2.3 (Exploitable Security Model). We say that a security model S is exploitable if the following holds for every protocol π analyzed in S: every adversary that launches a successful no-match attack on π can efficiently be used to break the security of the protocol with respect to the security definition of S.

In traditional security models, no-match attacks can be exploited in two ways. The most general way (that we already used before) is to ask a Reveal-query to the oracle with the same original key as the Test-oracle. Next we use this key to answer the Test-query. By the definition of a no-match attack, the two oracles are not partnered, and thus this constitutes a valid (as defined by the winning condition) attack in the security model.

Another way to exploit no-match attacks is by breaking explicit mutual [6] (or unilateral/server-only [24]) authentication, a security property that is not common in every security model for key exchange. Essentially, the requirement says that if an oracle accepts (i.e. computes some key), then there must always be another accepting oracle that has matching conversations to this oracle. It is clear that our no-match attacks directly break authentication as they can make two oracles accept (even with the same key) although they do not have matching conversations.[9] In combination, these two approaches for exploiting no-match attacks make them applicable to a broad class of security protocols (and their respective security models). This class contains key-exchange protocols that are based on models which only (explicitly) formalize key-indistinguishability like the CK [17] or eCK [37] model, and combinations of both, i.e. protocols secure in models which besides key-indistinguishability also explicitly require authentication (in the sense of [6]). Moreover, no-match attacks can also be launched against protocols that are secure under the notion of ACCE protocols [29]. In general, our no-match attacks can theoretically be applicable to all protocols that rely on matching conversations to define partnering, including for example password-based key exchange protocols like [5].

2.4 Strong Security and Deterministic Computation of v

It is relatively obvious that our no-match attacks can succeed if v is computed using a probabilistic algorithm. For example, if v is a digital signature, an attacker that obtains the secret signing key can simply re-sign the message to compute v'. With high probability we have $v' \neq v$. What is more subtle is that our attacks also work if the signature scheme used to compute v provides strong security or even is deterministic. This is exactly the point where many security proofs fail. Let us go into more detail for our running example, signed Diffie-Hellman. Recall that the security definition of strongly secure signatures gives the adversary access to the public key and a signing oracle. The winning condition is that the adversary can produce a new message/signature pair. Now consider an attempt to reduce the security of signed Diffie-Hellman to the strong security of the signature scheme. The crucial point is that in no-match attacks with advice the adversary is also given the secret key. In particular, the security definition of strongly secure signatures does not exclude that the adversary produces a new signature on a previously queried message *when the secret key is given.*

Quite similarly, it is not enough to require that the signature scheme has a deterministic signing procedure. The problem is that this only guarantees that the signing algorithm *specified by the signature scheme* outputs a single signature v per message m. However, there may exist other algorithms that output, given m, a signature $v' \neq v$ such that both m, v and m, v' pass the signature verification positively. At this point we remark that the very same argument on deterministic signatures can be made for the case of no-match attacks without advice as well showing that deterministic signing does not guarantee strong security. So in sum requiring deterministic signatures, or more generally deterministic cryptographic primitives that compute the protocol messages, is not helpful at all.

3 NO-MATCH ATTACKS ON EXISTING PROTOCOLS

As sketched before many security proofs of key exchange protocols fail to cover no-match attacks. At the same time, it can be hard to find out where the exact problem is. In this section, we therefore

[8] We want to avoid the term 'partnered' here because we consider no-match attacks after which two oracles are essentially not partnered.

[9] For simplicity we assume that there is no potential third oracle having matching conversations. (An active attacker, that controls the inputs to all oracles, can easily make any oracle not have matching conversation to some other oracle.)

present a diverse list of examples of existing protocols *together with concrete descriptions of successful no-match attacks*. Note that all our no-match attacks with advice apply to security models that also claim to model perfect forward secrecy (PFS), key compromise impersonation (KCI) attacks, or state (or ephemeral key) reveal attacks. Intuitively, security against key impersonation attacks means that key indistinguishability even holds if the adversary is provided with the long-term secret of the Test-oracle at the beginning of the security game. Perfect forward secrecy, in contrast, usually means that the adversary is also allowed to obtain the long-term keys of some oracle – which now even may be the Test-oracle or an oracle partnered to the Test-oracle – after the oracle has accepted and computed the session key. In stronger security models (like eCK) the adversary is even allowed obtain the long-term keys before the oracles accept. In the following we have classified the affected protocols (in their corresponding security models) with respect to the cryptographic mechanisms they rely on in the last message. We provide a detailed example of a no-match attack in Appendix B.

3.1 Integrity Protection – MACs

The classical definition of MACs is deterministic. However, recently Dodis, Kiltz, Pietrzak and Wichs [21] and Blazy, Kiltz, and Pan [11] introduced efficient probabilistic and provably secure MACs that have a variety of useful applications. We show that employing probabilistic MACs for integrity protection in key exchange protocols can be problematic. The first class of attacks deals with protocols where Alice and Bob (i.e. their oracles) use their long-term secret keys to derive a secret MAC key that is used to protect the integrity of the previous protocol messages. Assume that Alice sends a probabilistically computed tag over all previous messages in the last protocol message v. As in the general description of no-match attacks, the attacker intercepts this message and computes a new message $v' \neq v$ that instead is sent to Bob. This works as follows: By the power of the queries granted when modeling KCI and PFS security, the attacker can obtain Alice's secret key (at the latest) after she has sent v. In the next step, the attacker can compute the MAC key used for the last protocol message. Finally, the attacker uses this MAC key to compute a new tag v'. Both Alice and Bob accept with the same key although they are not partnered since Alice has sent v while Bob has received v'. A protocol which allows the use of probabilistic MACs that is susceptible to our attacks is the Jeong-Kwon-Lee KAM protocol (CANS'06) [31]. For concreteness, let us assume that Alice's oracle is the Test-oracle. The attack succeeds since, in the KAM protocol, the MAC key can be computed from the secret long-term key of Alice. However, in the KAM protocol the MAC key can also be computed from the secret ephemeral key of Bob's oracle which in turn can be obtained via a RevealState query. Thus there is a second way to launch a no-match attack that utilizes the RevealState query. The attack proceeds exactly in the same way except that the MAC key is computed using Bob's ephemeral secret key.

3.2 Authentication via Digital Signatures

In the following we will concentrate on protocols where Alice sends a digital signature on message m to Bob as the last protocol message. Our no-match attack proceeds exactly as outlined in our introductory example Signed Diffie-Hellman: the adversary first intercepts Alice's signature and uses her secret signing key to generate a new signature on m. With overwhelming probability this signature will differ from the original one. Finally, it sends the new signature to Bob who checks its validity (and, on success, possibly sends some other values to Alice). At the end of the protocol, Alice will accept although there is no matching conversation with Bob. Protocols that are susceptible to this attack are the signed Diffie-Hellman protocol by Sarr et al. [39] that relies on the NAXOS transformation [37] (SCN'10), the signature-based compiler [20] by Cremers and Feltz which achieves perfect forward secrecy (PFS) in two-message or one-round key exchange protocols (ESORICS'12)[10], the signature-based protocol [2] by Alawatugoda et al. (ASIACSS'14), and the recent signature-based protocol [9] by Bergsma et al. (PKC'15). The security models of these works all allow the adversary to obtain Alice's long-term secret before the last protocol message is received by Bob's oracle. At the same time the signature schemes used are either probabilistic or deterministic. We remark that since the authors of [2] do not require strong security, their protocol can also simply be attacked by a no-match attack without advice. We provide a detailed illustration of our attack on the compiler by Cremers and Feltz (when applied to the NAXOS protocol) in Appendix B. We remark that our no-match attack leads to the odd situation that this compiler, which is designed to increase the security of the input protocol, actually outputs a theoretically insecure protocol even if the input was secure to begin with.

3.3 Authentication via Public Key Encryption

In the following, we consider a no-match attack that can be launched if Bob sends an encrypted message to Alice that has to be decrypted and checked using Alice's secret key. In the following, Alice's oracle will serve as the Test-oracle. Recall that in the KCI security experiment the adversary is given Alice's secret key. It can thus intercept the ciphertext, decrypt it and compute a new ciphertext on the same message. If the encryption system is probabilistic (what is required to guarantee mere CPA security) the ciphertext will differ from the original one with high probability. However, Alice will accept the new ciphertext without having a matching conversation with Bob. Our no-match attacks can be applied to the recent protocol by Alawatugoda, Boyd, and Stebila [1] (ACISP'14) which presents a key exchange protocol that relies on a public key encryption scheme that is secure under adaptively chosen ciphertext attacks in the presence of after-the-fact leakage (CCLA2) [22]. (Essentially the

[10]The authors of [20] consider four models: M^w, eCK^w, M-PFS, and eCK-PFS. In contrast to (eCK^w, eCK-PFS), M^w and M-PFS do not consider EphemeralKey queries which reveal the ephemeral secret keys of protocol sessions. Note that in [20], Cremers and Feltz require that the signature scheme does not reveal the long-term keys even if the random coins are revealed (via an EphemeralKey-query). This can be realized using a deterministic signature scheme. In fact, in their security theorem Cremers and Feltz explicitly specify the signature scheme to be deterministic. The authors also claim (Remark 1 in [20]) that the signature-based compiler could use a randomized (strongly) unforgeable signature scheme if the security model does not allow to ask a EphemeralKey-query. However, we can show that even if the security models are restricted in one of these ways (deterministic signature scheme vs. probabilistic signature scheme with no EphemeralKey-query), the signature-based compiler is still vulnerable to our no-match attacks. (Moreover, the attack also works if the random coins involved in the signature generation cannot be revealed by a EphemeralKey.)

definition is like IND-CCA security but the adversary can obtain bounded information on the secret key.)

3.4 The BPR Framework (EC'00)

Under certain circumstances, no-match attacks can be launched against the password-based key exchange protocol of Bellovin and Merrit [8] that was provided by Bellare et al. (BPR) [5]. BPR analyze a two message protocol EKE2, where the last message consists of a single ciphertext c. Also, the derived session key does not depend on c but only on the plaintext encrypted in c. For the concrete instantiation of the encryption scheme they refer to [10], which proposes several deterministic ciphers with arbitrary finite domains. We observe that using any of the ciphers in [10] to instantiate EKE2 does not prevent no-match attacks if i) two distinct ciphertexts are mapped to the same plaintext by the decryption routine, as it, for example, is the case in the cycle-walking cipher of [10] and ii) the receiver of the ciphertexts does not check if the ciphertexts are in the correct domain. However, none of the ciphers in [10] explicitly check ii) in the decryption algorithm.

3.5 Further Primitives

The above list of possible no-match attacks is by no means complete. We have focused on the most popular cryptographic primitives used in key exchange protocols. Also our selection of no-match attacks should exemplify the diversity of possible no-match attacks.

There are many primitives that when used in key exchange protocols can lead to no-match attacks. Possible candidates are all primitives that make one or more messages be computed probabilistically like for example NIZK proofs, or probabilistic authenticated encryption systems. However, we stress that general statements on the vulnerability of some primitives to no-match attacks are inappropriate. No-match attacks are not launched against primitives but against a concrete protocol in a concrete security model. At the same time we also want to emphasize that no-match attacks not only rely on modifications of cryptographic values. In fact, they may modify any data that is exchanged between two communication partners. Indeed this is one reason why they are so hard to spot. For example, in Appendix E.2 we describe subtle no-match attack that exploit that group elements can have more than one representation. Additionally, Appendix C shows that no-match attacks can be launched by modifying values that are entirely independent of the remaining protocol messages and in particular of the session key.

4 NOVEL DEFINITIONS OF PARTNERING

Let us stress once again that the above attacks do seemingly not harm the practical security of the above protocols in any meaningful way. However, strictly speaking their security proofs are not sound. In this work, we propose several approaches to armor protocols against no-match attacks. Ideally our solutions should be as little invasive as possible to make them easily applicable to existing protocol implementations as well (either via only minor modifications of the protocol or no modification at all). In the following we will concentrate on our main and recommended solution to the problem of no-match attacks – a modification of the definition of partnering (as compared to modifications of the primitives or the overall protocol). In essence, we propose a careful relaxation of the notion of matching conversations. In Appendix E we propose two other solutions. The first relies on a careful instantiation of the building blocks that are used in a security protocol. More concretely it proposes to only use *unique* primitives and unique message encodings. The second solution relies on a compiler (in the random oracle model) which makes the computation of the session key crucially depend on every message bit that is exchanged between two parties. In this way, the computation of the session key needs to be adapted slightly. Before we detail our new definition of partnering, we note that although no-match attacks do not directly give rise to practical attacks they can be problematic for the application they are used in, for example if the protocol is used under the (implicit) assumption that there is a correspondence between matching conversations and successful key establishment. Thus protocols that have been analyzed on the basis of a modified definition of matching conversations should only be used after a careful analysis of the application scenario.

Intuitively, what we want from a secure protocol (from a design perspective) is that any active modification of the exchanged messages should result in the two communicating oracles to compute distinct and unrelated keys. This models that the adversary has no way of meaningfully tempering with the messages. Now since the keys are unrelated we can also allow the adversary to reveal one of the keys. This should not harm the secrecy of the other key. With this rationale in mind, we can clearly isolate where no-match attacks introduce a theoretical problem. In a no-match attack, the keys remain the same although there is an active attack (see Figure 1 and Figure 2). Our final solution is to change the security model such that no-match attacks (and only them!) do not break the security of the protocol anymore. Arguably, this is legitimate since no-match attacks do not constitute realistic attacks anyway. Thus, from a more abstract viewpoint we can view the omission of no-match attacks from the set of valid attacks as an attempt to adapt theoretical security definitions for key exchange to model practical protocol settings more realistically.

4.1 Robust Matching Conversations

What we intuitively require from our modified definition of partnering is that it deletes no-match attacks from the set of attacks that are considered valid. In our new definition, we try to capture only 'meaningful' modifications of the messages by the adversary.

Intuitively, what we want is a definition that is useful in situations where it may not be harmful for Bob to accept a cryptographic value v' that has been produced by an adversary. For example, in signed Diffie-Hellman, if the signature v' is valid and produced on the same message as Alice's signature v, one could argue that Bob may still be able to securely use the corresponding session key in practice. A natural first approach to define this kind of partnering is to relax the definition of matching conversations. More concretely one could define a form of 'robust matching conversations'. Under such a definition the session identifier of an oracle would not only consists of the transcript of messages that this oracle actually generated in a protocol run. In contrast, the session identifier of an oracle would additionally contain the entire set of transcripts that could have been produced by that oracle in presence of a no-match attack.

However, there are two major disadvantage of such a definition. The first one is that it ultimately ties the formalization of security (i.e. of the session identifier) to the design of the protocol (i.e. to the ways the protocol admits no-match attacks). The second one is that it may be difficult to specify the session identifier because they are hard to spot or just because there are too many. Our final definition circumvents these problems entirely.

4.2 Original Key Partnering

We propose to use the following partnering definition as a standard substitution of matching conversations to exclude no-match attacks while still providing a very high level of security.

Definition 4.1 (Original Key Partnering). Two oracles a and b are said to be partnered if both of them have computed their original key.

Notably, original key partnering ultimately *decouples* the definition of partnering from the sent protocol messages. Besides being theoretically compelling, this is an important benefit as one does not have to explicitly consider all no-match attacks possible in a protocol. Moreover, it is a useful partnering definition in situations in which it is hard to find an appropriate message-dependent partnering definition in the first place, for example when dealing with PKE encryption as detailed in Section 3.3 and Appendix E.1. Its independence from concrete messages makes original key partnering a very general and versatile definition of security. Furthermore, it is a very practical partnering concept as it concentrates on a single cryptographic value, independent of the complexity of the protocol. This supports simple falsification. Our definition can generically be plugged into all existing security definitions that rely on matching conversations. At the same time it fixes all MC-based security proofs that fail to address no-match attacks: as an immediate consequence of original key partnering, no-match attacks vanish from the spectrum of possible attacks against a protocol (Figure 3).

We remark that the definition of original key partnering nicely supports the following, natural intuition: if (despite there being an active attack) the session keys computed by two oracles are equal to those computed in a protocol run with a passive attacker then these keys are treated as being (cryptographically) as good as well. To give a slightly different perspective, our definition might be viewed to follow some form of ideal/real world paradigm. First we define an ideal run of a key exchange protocol where the attacker is passive only. Our definition decides that for key exchange protocols the defining characteristic of such a run is that both oracles have computed their original key, i.e. the original key is computed by Alice and Bob without any adversarial interference. Next we classify any attack with respect to this ideal run. If after some adversarial modification both oracles still have computed the original key, then this does not deviate from the ideal definition, and thus does not constitute a valid attack.

In our definition of original key partnering we deliberately avoid to explicitly enumerate all possible ways in which messages may be modified to launch a successful no-match attack. Besides being very complex such a classification is always highly protocol-dependent. Intuitively, our definition rather focuses on the fact that *the modified messages trigger the same behavior of the receiver as the original ones* – and ultimately lead to the computation of the same (original) key.

When comparing original key partnering with previous partnering notions we can observe that similar to matching conversations our new definition fulfills the first set of desirable properties, 1)–6), given in Section 1. However, in contrast to matching conversations it also fulfills properties 7)–9). Proving security under original key partnering is comparable to proofs under matching conversations, with a tendency to be *slightly simpler* (since less cases have to be considered): proofs under matching conversations usually distinguish active and passive attacks. Roughly, first active attacks are ruled out by reducing to the properties of the underlying primitives. Then security follows from the passive security of the protocol. In contrast, when using original key partnering, we only have to consider a subset of active attacks in the first step, namely those that make either Alice or Bob not compute their original key. We provide an illustrative example protocol in Appendix D together with a proof sketch.

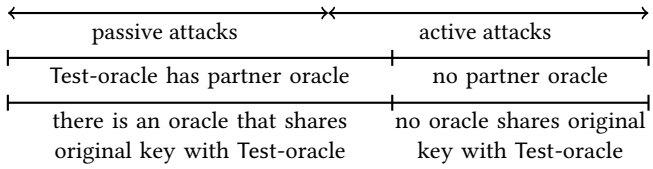

Figure 3: Attack spectrum and corresponding states of the Test**-oracle when relying on original key partnering.**

Remark 1. We stress that for our security definition it is not helpful to isolate messages that *contribute* to the computation of the session key from those that do not, as it is usually done when relying on partial matching conversations. The core problem is that no-match attacks may aim to modify both, messages which do contribute to the session key derivation but also messages which do not. For example, the signatures in our running example obviously contribute to the session key generation in the sense that if they are invalid, the session key will not be computed at all. However, our results show that in some protocols one can launch a successful no-match attack by modifying these signatures.[11] One the other hand, in Appendix C we show that in traditional security models, augmenting protocol messages with new but entirely independent messages can lead to no-match attacks, too.

We also remark that despite its apparent generality the partnering function given in [7] does not cover our definition. Roughly, a partnering function takes as input a single oracle, a communication transcript, and the (so-called) intended partner of that oracle (see Section 5) and outputs the partnered oracle, if it exists. (If the input oracle communicates with the adversary, the function simply outputs a fixed error symbol. It is important to note that in such situations, it is trivial for the adversary to compute the session key of the Test-oracle - it can simply run the protocol with the Test-oracle.) In contrast, our partnering definition rather takes as input *two* oracles. Conceptually, each pair of oracles, is associated with their original key. (So, for n oracles there are $O(n^2)$ original

[11] Some signature schemes, like re-randomizable signatures, are even by definition modifiable, albeit in some well-restricted way.

keys.) In our definition, two oracles are then considered partnered if both have indeed computed their original key. To deal with situations in which an oracle is communicating with the adversary we can require that the peer of the Test-oracle is uncorrupted, as for example in [37].

Similarly, our new definition of original key partnering cannot be viewed as an instantiation of the session identifier concept, where each oracle is associated with a session identifier and partnering is decided by comparing the session identifiers of two oracles. In our new definition each oracle may rather be viewed to have several potential session identifiers (original keys), one with each other oracle. Partnering only holds if both oracles have indeed computed it.

4.3 Partnering via Mere Key Comparison is Not Sufficient

We would like to emphasize that we *do not define partnering by comparing session keys*. Such a definition would *fail to reflect real-world attacks in the security model*. Let us provide a brief example. For example, we can easily consider a protocol where the adversary can – via an active attack – make an oracle accept with a key of some other oracle. In practice, an adversary could for example make a client computer with no physical protection mechanisms compute the same key that is shared between two strongly protected servers. Attacking the client computer would then also reveal the servers' session key and an adversary could simply distinguish it from a random key. We clearly want that such an attack is reflected in a strong security model. More generally, any protocol where the adversary can make an oracle compute a given key should be considered insecure and most security protocols successfully protect against this kind of attacks.[12] However, when we use mere key comparison to define partnering, revealing the client computer before breaking key indistinguishability of one of the server oracles is deemed a trivial attack as it holds the same key as the two servers. This is highly unintuitive. When using original key partnering, in contrast, this is a legitimate attack in the security model. This choice of partnering definition clearly distinguishes our work from all previous results that (partly) define partnering by considering if two oracles have merely computed the same key (not necessarily the original key), for example [5]. Our definition is stronger than all these definitions: with original key partnering, if an attacker manages to make two oracles accept with the same key (distinct from the original key) then this is *guaranteed* to be deemed a successful attack. In contrast, mere key comparison may not recognize this as an attack at all.

4.4 Public vs. Secret Partnering

As the original key must be kept secret, original key partnering does in general not allow the adversary to efficiently test whether two oracles are partnered. In general, it is thus harder for the adversary to base its strategy on the fact that in the past certain pairs of oracles have been partnered or not. We would like to comment on this in more detail and argue why relying on such a 'hidden' partnering definition is not a serious restriction.

[12] If the attacker is also allowed to reveal session keys it should even be possible to successfully break key indistinguishability independent of the attackers computational power.

First, we stress that the adversary controls the entire network and can decide on which message is delivered to which oracle. Moreover when behaving entirely passive, the adversary already knows exactly which oracles are partnered. Thus any unclarity in deciding if two oracles are partnered is always introduced by the adversary herself in exactly the way she desires to. It is also worth mentioning that the adversary knows the protocol layout and the used primitives. So it already knows which changes to the transmitted messages influence the computation of the session key. At the same time we stress that our definition of partnering does by no means exclude that there also exist (efficient) public algorithms for checking if two oracles are partnered. This is important when we consider results like the compiler of [15] which explicitly requires an efficient *public* function for checking whether two oracles are partnered with each other.

Second, from a theoretical point of view the partnering definition is only required in the security analysis. For provable security, it is not required that any entity can actually compute partnered oracles efficiently. Partnering is merely used to define the winning conditions of the adversary. In particular, we do not have to make the simulator be able to check if the partnering condition (as part of the winning condition) is violated. In the security proof, we simply assume an adversary that breaks the security definition and, as part of that, respects the partnering definition. This is quite similar to the fact, that we cannot have the simulator in the indistinguishability game be able to verify if the adversary's guess (for a real or a random key) is actually true. (The latter fact is simply because any simulator that can verify the adversary's guess can already compute the solution on its own. Thus it cannot embed the security challenge at this point.)

Finally, we would also like to emphasize that it is not necessary to use session identifiers to formalize the access of the adversary to its queries. Although there exist security models like the Canetti-Krawczyk model [17] or the model in [5] that uses session identifiers to access oracles (note that this strategy is problematic when session identifiers are post-specified), it is always possible to formulate the entire security game without them, for a recent example of this type of formalization see the model used in [24].

We also remark that some security models specify that the session identifier computed by two oracles is given to the adversary after they computed the session key, for example in [5]. This also holds for the Test-oracle. However, in these models session identifiers are based on public information only (and used to formalize access to queries). As usual, the session key must be kept secret. Therefore basing our session identifier on secret original keys is, in some sense, not a restriction. In particular, the adversary can still access the same public information generated in the protocol run by two oracles as in the model of [5].

4.5 Robust Composition

Original key partnering can essentially relax security models such that no-match attacks are not considered as legitimate attacks anymore. In Appendix C we show that our new definition has another important advantage, a basic form of composability that we call robust composition. Moreover, we show that partnering based on

matching conversations lacks even the weakest form of robust composition.

5 NO-MATCH ATTACKS AND EXPLICIT AUTHENTICATION

Our solutions so far deal with no-match attacks in security models for key exchange that do not explicitly define authentication. Correspondingly no-match attacks can only break key indistinguishability in these models. When dealing with explicit authentication, we can also rely on our alternative solutions given in Appendix E. They inherit the same benefits and disadvantages as when used to deal with no-match attacks against key indistinguishability.

However, again our main and recommended solution is to provide a new definition of explicit authentication that does not inherently rely on matching conversations. The new definition is inspired by the new partnering definition given in Section 4.1. In the following we will elaborate on our new definition of explicit authentication in more detail. We proceed in two steps to obtain our final notion. In the first step, we develop *a new notion of explicit authentication* by decoupling the classical notion of authentication from the notion of matching conversations. We arrive at what we believe to be the first notion of authentication that is independent from a concrete partnering definition. (Previous definitions have usually been based on matching conversations.) This is conceptually very attractive as it allows to *parametrize both key indistinguishability and explicit authentication with a single common choice of partnering notion*. In the second step we instantiate our new general notion of explicit authentication with our new notion of original key partnering. We arrive at a notion that is well-suited to deal with no-match attacks in models with explicit authentication.

5.1 Decoupling Explicit Authentication and Matching Conversations

As a template we use the definitions provided in the original work of Bellare and Rogaway [6]. However, we only focus on the second property of the definition for authentication given in [6].[13] We stress that, as we rely on original key partnering, our definitions are specific to key exchange models.

Let a be an oracle of A. The *peer* B of a is the holder of the key that a uses to verify/authenticate the (received) communication with. (For example, we may assume that Alice uses Bob's public key to verify messages that has been sent from Bob to Alice. Likewise Alice could use MAC key shared with Bob to check if some message has been authenticated by Bob.) The peer of a is sometimes also called the intended partner of a.

Definition 5.1 (General Explicit Mutual Authentication). We say that a protocol provides explicit authentication if the following holds: if an oracle a of A with uncorrupted[14] peer B accepts, then there exists an oracle b with peer A that is partnered with a.

[13] We regard the first part of this definition – which states that oracles with matching conversations both accept – as a definition of completeness rather than a formulation of a security property.

[14] We note that the authentication definition [6] does not explicitly require that the peer is uncorrupted. However, the entire security model does not allow for corruptions at all.

When plugging in matching conversation as the partnering definition of choice we obtain the classical definition of security for authentication.

5.2 Explicit Authentication and Original Key Partnering

We will now plug our new partnering definitions into the general definition. Putting things together we obtain the following simple notion of explicit authentication that rules out no-match attacks.

Definition 5.2 (Explicit Mutual Authentication with Original Key Partnering). We say that a protocol provides explicit authentication with original key partnering if the following always holds: if an oracle a of A with uncorrupted peer B accepts, then there exists an oracle b with peer A that accepts such that a and b have computed their original key.

Similar to before, our new notion allows the adversary to perform any modification on the protocol as long as the oracles of both A and B compute the original key. Again, such modifications are not considered an attack. We remark that the same rationale for using original key partnering in key exchange protocols applies to our new definition of authentication as well. We note that since our partnering definition is independent of the message flow, we do not need to separately cover situations where A or B has sent the last protocol message like for example in [29]. Moreover no artificial last message is required as in [6].

6 STRENGTHENING SECURITY DEFINITIONS

So far we have showed that relying on original key partnering can relax security models such that no-match attacks are not considered as legitimate attacks anymore. We would like to remark that original key partnering can also be used to *strengthen* security models. In particular, we can show that original key partnering can be used to re-define what it means for an oracle to communicate with an honest oracle (and not with the attacker). In this way we immediately obtain stronger notions for, for example, unilateral authentication and perfect forward secrecy than existing approaches. Let us expand on this for unilateral authentication – the widespread scenario where an unauthenticated client communicates with an authenticated server.

Unilateral security typically requires that only client oracles may serve as Test-oracles, since in settings with an authenticated server and an unauthenticated client, only the client can verify if incoming messages really were sent by its communication partner. However, usually security definitions are extended so that server oracles may serve as Test-oracles as well, albeit only under very severe restrictions (for example [24, 34]). Essentially, these restrictions guarantee that the server oracle indeed communicates with an honest client oracle and not with the adversary. This is reasonable since an attacker that simply behaves like an unauthenticated client can always engage in a communication with a server oracle and know the session key afterward (it just runs the algorithms any other client would run). Obviously it is very crucial to define for a server oracle what it means to communicate with an honest oracle. Existing approaches to define this rely on matching conversations.

They essentially state that server oracles may also serve as Test-oracles if there exists a fresh (i.e. session key and secret long-term key have not been given to the adversary) client oracle that has accepted with matching conversations with the server oracle. This is very strict. In particular it means that the communication between those oracles has not been modified on transit at all, i.e. the attacker must remain passive. Our new definition may provide a slightly more generous alternative: one may alternatively allow a server oracle to serve as a Test-oracle if there exists a fresh client oracle that has accepted and both client and server oracle have computed their original key. In contrast to before, this means that the attacker is allowed to perform at least some active attacks, as long as these attacks do not change the fact that both oracles compute their original key. Another definition that can be strengthened using the concept of original key partnering is the notion of origin session that was introduced by Cremers and Feltz [20] and used to formalize notions of perfect forward secrecy. For a similar reason as with unilateral authentication it tries to capture messages that really originate from some oracle (and were not crafted by the attacker). The definition requires that for a considered message there must exist an oracle that exactly output this message before. Original key partnering can relax this requirement by additionally allowing that this message has been produced via a no-match attack by the adversary.

7 FURTHER RELAXATIONS AND OBSTACLES

Our new notion improves security definitions based on matching conversations while still keeping its important conceptual advantages. It does so by relaxing the security definition such that certain active modifications by the attacker are not considered attacks anymore. However, there is a strict, reasonable limit to the set of allowed modifications: only those attacks are not considered harmful that make both parties still compute their original key. Intuitively, this means that in the presence of any such attack the computed shared secret is still as good as in case of no attack at all. We feel that this well-defined lower limit that corresponds to a strong cryptographic intuition is one of the most compelling and valuable features of our new notion. In general finding a relaxation of a strict security notion should always be accompanied by a thorough explanation of why it does not consider attacker actions harmless which might actually be dangerous. We emphasize, that potentially there is still room for improvement of our partnering definition. Consider for example (artificial) protocols in which two key exchange protocols of equal strength are run at the same time between Alice and Bob. The session key is one of the session keys of the underlying protocols. Now assume the attacker launches an active attack that simply makes Alice and Bob use the other session key. Corresponding to our notion such an influence on the shared key is considered an attack, as it is the case with security under matching conversations. Cryptographic intuition however, may deem this protocol secure. Essentially our security notion requires that whenever the attacker changes the key of some party this is considered an attack. We find this generally very reasonable since we have no guarantee of the strength of the key after the modification (in contrast to modifications deemed harmless according to original partnering). Future work may strive to further weaken the notion of original key partnering to also recognize these kind of modifications as non-harmful. However, to us it is not clear if any such definition would not also consider modifications as harmless that cryptographic intuition would deem legitimate attacks. In particular, to be convincing, such a definition should have a strong security argument akin to the strict limit of allowed modifications that we mentioned before which guarantees that after the modification of used session keys by the attacker, the actually used keys are still strong.

Apart from that, we believe that such a relaxation may be problematic on a technical level as well. Let as go into more detail. As an example one could think of a similar protocol with two runs of atomic key exchange protocols in parallel. The only difference now is that the two atomic protocols have distinct strength, one with strong security and one with weak security.Imagine that in case of no active attack Alice and Bob take as session key the one key from the strongly secure protocol. Now the attacker may modify the protocol to make both parties use the weak key. Clearly this must be considered an attack. However, the only difference between the protocol mentioned before is the strength of the second atomic protocol run. Before both atomic protocols had the same strength, now the second is weaker. So to account for this, a general security definition (that crucially relies on a good partnering definition) must define what is a legitimate attack and what can be considered harmless *depending on the strength of one of its building blocks*. Such an approach is very counter-intuitive. For example, the *security definition* of a digital signature construction that computes long messages via a hash function should also not depend on the strength of the underlying hash function. It is only the security proof that should rely on the hash function's strength.

REFERENCES

[1] Janaka Alawatugoda, Colin Boyd, and Douglas Stebila. 2014. Continuous After-the-Fact Leakage-Resilient Key Exchange. In *ACISP 14 (LNCS)*, Willy Susilo and Yi Mu (Eds.), Vol. 8544. Springer, Heidelberg, 258–273. https://doi.org/10.1007/978-3-319-08344-5_17

[2] Janaka Alawatugoda, Douglas Stebila, and Colin Boyd. 2014. Modelling after-the-fact leakage for key exchange. In *ASIACCS 14*, Shiho Moriai, Trent Jaeger, and Kouichi Sakurai (Eds.). ACM Press, 207–216.

[3] Boaz Barak, Yehuda Lindell, and Tal Rabin. 2004. Protocol Initialization for the Framework of Universal Composability. Cryptology ePrint Archive, Report 2004/006. (2004). http://eprint.iacr.org/.

[4] Mihir Bellare, Dennis Hofheinz, and Eike Kiltz. 2015. Subtleties in the Definition of IND-CCA: When and How Should Challenge Decryption Be Disallowed? *Journal of Cryptology* 28, 1 (Jan. 2015), 29–48. https://doi.org/10.1007/s00145-013-9167-4

[5] Mihir Bellare, David Pointcheval, and Phillip Rogaway. 2000. Authenticated Key Exchange Secure against Dictionary Attacks. In *EUROCRYPT 2000 (LNCS)*, Bart Preneel (Ed.), Vol. 1807. Springer, Heidelberg, 139–155.

[6] Mihir Bellare and Phillip Rogaway. 1994. Entity Authentication and Key Distribution. In *CRYPTO'93 (LNCS)*, Douglas R. Stinson (Ed.), Vol. 773. Springer, Heidelberg, 232–249.

[7] Mihir Bellare and Phillip Rogaway. 1995. Provably Secure Session Key Distribution: The Three Party Case. In *27th ACM STOC*. ACM Press, 57–66.

[8] Steven M. Bellovin and Michael Merritt. 1992. Encrypted Key Exchange: Password-Based Protocols Secure against Dictionary Attacks. In *1992 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 72–84. https://doi.org/10.1109/RISP.1992.213269

[9] Florian Bergsma, Tibor Jager, and Jörg Schwenk. 2015. One-Round Key Exchange with Strong Security: An Efficient and Generic Construction in the Standard Model. In *PKC 2015 (LNCS)*, Jonathan Katz (Ed.), Vol. 9020. Springer, Heidelberg, 477–494. https://doi.org/10.1007/978-3-662-46447-2_21

[10] John Black and Phillip Rogaway. 2002. Ciphers with Arbitrary Finite Domains. In *CT-RSA 2002 (LNCS)*, Bart Preneel (Ed.), Vol. 2271. Springer, Heidelberg, 114–130.

[11] Olivier Blazy, Eike Kiltz, and Jiaxin Pan. 2014. (Hierarchical) Identity-Based Encryption from Affine Message Authentication. In *CRYPTO 2014, Part I (LNCS)*,

Juan A. Garay and Rosario Gennaro (Eds.), Vol. 8616. Springer, Heidelberg, 408–425. https://doi.org/10.1007/978-3-662-44371-2_23
[12] Emmanuel Bresson, Olivier Chevassut, David Pointcheval, and Jean-Jacques Quisquater. 2001. Provably Authenticated Group Diffie-Hellman Key Exchange. In *ACM CCS 01*. ACM Press, 255–264.
[13] Emmanuel Bresson, Mark Manulis, and Jörg Schwenk. 2007. On Security Models and Compilers for Group Key Exchange Protocols. In *IWSEC 2007, Nara, Japan, October 29-31, 2007, Proceedings (Lecture Notes in Computer Science)*, Atsuko Miyaji, Hiroaki Kikuchi, and Kai Rannenberg (Eds.), Vol. 4752. Springer, 292–307. https://doi.org/10.1007/978-3-540-75651-4_20
[14] Christina Brzuska, Mark Fischlin, Nigel P. Smart, Bogdan Warinschi, and Stephen C. Williams. 2013. Less is More: Relaxed yet Composable Security Notions for Key Exchange. *International Journal of Information Security* 12, 4 (August 2013), 267–297. https://doi.org/10.1007/s10207-013-0192-y
[15] Christina Brzuska, Marc Fischlin, Bogdan Warinschi, and Stephen C. Williams. 2011. Composability of Bellare-Rogaway key exchange protocols. In *ACM CCS 11*, Yan Chen, George Danezis, and Vitaly Shmatikov (Eds.). ACM Press, 51–62.
[16] Christina Brzuska, Nigel P. Smart, Bogdan Warinschi, and Gaven J. Watson. 2013. An analysis of the EMV channel establishment protocol. In *ACM CCS 13*, Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung (Eds.). ACM Press, 373–386.
[17] Ran Canetti and Hugo Krawczyk. 2001. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *EUROCRYPT 2001 (LNCS)*, Birgit Pfitzmann (Ed.), Vol. 2045. Springer, Heidelberg, 453–474.
[18] Kim-Kwang Raymond Choo and Yvonne Hitchcock. 2005. Security Requirements for Key Establishment Proof Models: Revisiting Bellare-Rogaway and Jeong-Katz-Lee Protocols. In *ACISP 05 (LNCS)*, Colin Boyd and Juan Manuel González Nieto (Eds.), Vol. 3574. Springer, Heidelberg, 429–442.
[19] Cas Cremers. 2011. Examining indistinguishability-based security models for key exchange protocols: the case of CK, CK-HMQV, and eCK. In *ASIACCS 11*, Bruce S. N. Cheung, Lucas Chi Kwong Hui, Ravi S. Sandhu, and Duncan S. Wong (Eds.). ACM Press, 80–91.
[20] Cas J. F. Cremers and Michele Feltz. 2012. Beyond eCK: Perfect Forward Secrecy under Actor Compromise and Ephemeral-Key Reveal. In *ESORICS 2012 (LNCS)*, Sara Foresti, Moti Yung, and Fabio Martinelli (Eds.), Vol. 7459. Springer, Heidelberg, 734–751.
[21] Yevgeniy Dodis, Eike Kiltz, Krzysztof Pietrzak, and Daniel Wichs. 2012. Message Authentication, Revisited. In *EUROCRYPT 2012 (LNCS)*, David Pointcheval and Thomas Johansson (Eds.), Vol. 7237. Springer, Heidelberg, 355–374.
[22] Stefan Dziembowski and Sebastian Faust. 2011. Leakage-Resilient Cryptography from the Inner-Product Extractor. In *ASIACRYPT 2011 (LNCS)*, Dong Hoon Lee and Xiaoyun Wang (Eds.), Vol. 7073. Springer, Heidelberg, 702–721.
[23] Dario Fiore and Dominique Schröder. 2012. Uniqueness Is a Different Story: Impossibility of Verifiable Random Functions from Trapdoor Permutations. In *TCC 2012 (LNCS)*, Ronald Cramer (Ed.), Vol. 7194. Springer, Heidelberg, 636–653.
[24] Marc Fischlin and Felix Günther. 2014. Multi-Stage Key Exchange and the Case of Google's QUIC Protocol. In *ACM CCS 14*, Gail-Joon Ahn, Moti Yung, and Ninghui Li (Eds.). ACM Press, 1193–1204.
[25] Oded Goldreich. 2001. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press.
[26] Shafi Goldwasser and Silvio Micali. 1984. Probabilistic Encryption. *J. Comput. Syst. Sci.* 28, 2 (1984), 270–299.
[27] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. 1984. A "Paradoxical" Solution to the Signature Problem (Abstract) (Impromptu Talk). In *CRYPTO'84 (LNCS)*, G. R. Blakley and David Chaum (Eds.), Vol. 196. Springer, Heidelberg, 467.
[28] Shafi Goldwasser and Rafail Ostrovsky. 1993. Invariant Signatures and Non-Interactive Zero-Knowledge Proofs are Equivalent (Extended Abstract). In *CRYPTO'92 (LNCS)*, Ernest F. Brickell (Ed.), Vol. 740. Springer, Heidelberg, 228–245.
[29] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. 2012. On the Security of TLS-DHE in the Standard Model. In *CRYPTO 2012 (LNCS)*, Reihaneh Safavi-Naini and Ran Canetti (Eds.), Vol. 7417. Springer, Heidelberg, 273–293.
[30] Ik Rae Jeong, Jonathan Katz, and Dong Hoon Lee. 2004. One-Round Protocols for Two-Party Authenticated Key Exchange. In *ACNS 04 (LNCS)*, Markus Jakobsson, Moti Yung, and Jianying Zhou (Eds.), Vol. 3089. Springer, Heidelberg, 220–232.
[31] Ik Rae Jeong, Jeong Ok Kwon, and Dong Hoon Lee. 2006. A Diffie-Hellman Key Exchange Protocol Without Random Oracles. In *CANS 06 (LNCS)*, David Pointcheval, Yi Mu, and Kefei Chen (Eds.), Vol. 4301. Springer, Heidelberg, 37–54.
[32] Jonathan Katz and Yehuda Lindell. 2007. *Introduction to Modern Cryptography*. Chapman and Hall/CRC Press.
[33] Kazukuni Kobara, SeongHan Shin, and Mario Strefler. 2009. Partnership in key exchange protocols. In *ASIACCS 09*, Wanqing Li, Willy Susilo, Udaya Kiran Tupakula, Reihaneh Safavi-Naini, and Vijay Varadharajan (Eds.). ACM Press, 161–170.
[34] Florian Kohlar, Sven Schäge, and Jörg Schwenk. 2013. On the Security of TLS-DH and TLS-RSA in the Standard Model. Cryptology ePrint Archive, Report 2013/367. (2013). http://eprint.iacr.org/2013/367.
[35] Hugo Krawczyk. 2005. HMQV: A High-Performance Secure Diffie-Hellman Protocol. In *CRYPTO 2005 (LNCS)*, Victor Shoup (Ed.), Vol. 3621. Springer, Heidelberg, 546–566.
[36] Hugo Krawczyk, Kenneth G. Paterson, and Hoeteck Wee. 2013. On the Security of the TLS Protocol: A Systematic Analysis. In *CRYPTO 2013, Part I (LNCS)*, Ran Canetti and Juan A. Garay (Eds.), Vol. 8042. Springer, Heidelberg, 429–448. https://doi.org/10.1007/978-3-642-40041-4_24
[37] Brian A. LaMacchia, Kristin Lauter, and Anton Mityagin. 2007. Stronger Security of Authenticated Key Exchange. In *ProvSec 2007 (LNCS)*, Willy Susilo, Joseph K. Liu, and Yi Mu (Eds.), Vol. 4784. Springer, Heidelberg, 1–16.
[38] Kristin Lauter and Anton Mityagin. 2006. Security Analysis of KEA Authenticated Key Exchange Protocol. In *PKC 2006 (LNCS)*, Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin (Eds.), Vol. 3958. Springer, Heidelberg, 378–394.
[39] Augustin P. Sarr, Philippe Elbaz-Vincent, and Jean-Claude Bajard. 2010. A New Security Model for Authenticated Key Agreement. In *SCN 10 (LNCS)*, Juan A. Garay and Roberto De Prisco (Eds.), Vol. 6280. Springer, Heidelberg, 219–234.
[40] Sven Schäge Tibor Jager, Florian Kohlar and Jörg Schwenk. 2011. On the Security of TLS-DHE in the Standard Model. Cryptology ePrint Archive, Report 2011/219. (2011). http://eprint.iacr.org/2011/219.

A BRIEF OVERVIEW ON SECURITY MODELS FOR AUTHENTICATED KEY EXCHANGE

With the so-called Bellare-Rogaway model, Bellare and Rogaway have initiated the rigorous study of cryptographic models for key agreement protocols in the 1990s. This model has become the basis for many refined security models (e.g. [7, 29]). In 2001, Canetti and Krawczyk introduced a new key exchange model [17] – today known as the CK-model that additionally covers session state revelations but does not require explicit authentication. It reflects a wide range of practical attacks and has also become very popular. In 2005, Krawczyk extended this model to also capture key impersonation (KCI) security and weak forward secrecy. The resulting model is often referred to as $\mathrm{CK}_{\mathrm{HMQV}}$, or CK+, or simply the HMQV model [35]. In 2007, LaMacchia et al. introduced an even further refined model known as the eCK model for the analysis of two-party key exchange protocols [37]. The eCK model captures the exposure of ephemeral keys by using a dedicated EphemeralKey-query and also allows the exposure of ephemeral secret keys of the Test-oracle and its partner oracle. The eCK model captures a wide variety of practical attacks and is used in many security proofs. Subsequently, we will briefly recall the general setup. This is necessary when illustrating our no-match attacks in Appendix B. In particular, we stick to the term oracle to describe protocol instances run at a party. We remark that many of models refer to such instances as sessions.

Let π be a security protocol. Suppose we have a set $\mathsf{P}_1, \ldots, \mathsf{P}_n$ of honest parties (as potential protocol participants), where each honest party P_i has a long-term secret sk_i. We use π_i^s to denote the s-th instance of a protocol run at a protocol participant P_i. More intuitively, we will view π_i^s as an oracle of P_i. Each oracle may either be an initiator or responder type oracle defining which algorithms it is going to use to respond to incoming messages and to compute the session key. Moreover, we will assume that each oracle has an associated session state, including the intermediate random values used in the computation of the session key. At the same time, each oracle has access to the long-term secrets of P_i. In most security models, security is defined in a security game that is played between an adversary and a challenger. The adversaries task is to distinguish the key computed by the Test-oracle from a random key. We assume that the active adversary $\mathcal{A}$ is granted access to Send, Reveal, Corrupt, EphemeralKey and Test-queries.

B DETAILED EXAMPLE OF A NO-MATCH ATTACK

In the following we will in detail illustrate a concrete vulnerable protocol and a corresponding no-match attack against it. We do not provide detailed descriptions of the corresponding security model here. For more information we instead refer to the original papers. However, we remark that partnering is defined via (partial) matching conversations.

The SIG(NAXOS) Protocol. For simplicity and concreteness, we concentrate on the SIG(NAXOS) protocol (Figure 4) that relies on the protocol compiler of Cremers and Feltz [20] when applied to the NAXOS protocol [37].

Protocol Description.

- Setup: each party P_c has two independent valid long-term secret/public key pairs, one key pair for the NAXOS protocol ($pk_c = g^{x_c}, sk_c = x_c$), with $c \in \{i, j\}$, and one pair (pk_c^{sig}, sk_c^{sig}) for a randomized and strongly secure digital signature scheme SIG. Let $\mathsf{H}_1{:}\{0,1\}^* \to \mathbb{Z}_p$ and $\mathsf{H}_2{:}\{0,1\}^* \to \{0,1\}^\kappa$ be two hash functions, where κ is a security parameter.
- Party P_i selects a random number $\alpha_i \xleftarrow{\$} \mathbb{Z}_q$, computes $\Gamma_i = g^{\mathsf{H}_1(\alpha_i, sk_i)}$. Then, P_i computes a signature σ_i by evaluating the probabilistic SIG.Sign algorithm under the private key sk_i^{sig} with internal random coins r_i over input $(\Gamma_i||\mathsf{P}_j)$: $\sigma_i = \mathsf{SIG.Sign}(sk_i^{\text{sig}}, \Gamma_i||\mathsf{P}_j)$. Finally, P_i sends (Γ_i, σ_i) to P_j.
- Party P_j behaves similarly: it selects a random number $\alpha_j \xleftarrow{\$} \mathbb{Z}_q$, computes Γ_j and the signature σ_j over $\Gamma_j||\Gamma_i||\mathsf{P}_i$, and sends (Γ_j, σ_j) to P_i.
- Computation of session key: Party P_i checks if it holds that: $\mathsf{SIG.Vfy}(pk_i^{\text{sig}}, \Gamma_j||\Gamma_i||\mathsf{P}_i, \sigma_j) = 1$. If the check is passed, P_i computes the session key via the following equation: $\mathsf{K}_{i,j} = \mathsf{H}_2(\Gamma_j^{sk_i}, pk_j^{\mathsf{H}_1(\alpha_i, sk_i)}, \Gamma_j^{\mathsf{H}_1(\alpha_i, sk_i)}, i, j)$. Accordingly, Party P_j computes $\mathsf{K}_{i,j} = \mathsf{H}_2(pk_i^{\mathsf{H}_1(\alpha_j, sk_j)}, \Gamma_i^{sk_j}, \Gamma_i^{\mathsf{H}_1(\alpha_j, sk_j)}, i, j)$.

No-Match Attack against the SIG(NAXOS) Protocol. In our no-match attack described below the adversary $\mathcal{A}$ modifies message σ_i via queries granted in the KCI security game. Moreover, we choose π_i^s to be the Test-oracle. (We remark that the protocol is also susceptible to a no-match attack that, very similarly, exploits the power of the PFS security game. To this end the attacker can substitute σ_j with a fresh signature and use π_j^t as the Test-oracle.)

- The adversary $\mathcal{A}$ makes a Corrupt(P_i)-query and obtains the long-term private key of party P_i, sk_i^{sig}.
- According to the protocol specification, Party P_i computes $\Gamma_i = g^{\mathsf{H}_1(\alpha_i, sk_i)}$. In the next step, P_i computes a signature σ_i with internal random coins r_i over $\Gamma_i||\mathsf{P}_j$ as follows: $\sigma_i = \mathsf{SIG.Sign}(sk_i^{\text{sig}}, \Gamma_i||\mathsf{P}_j)$. Finally, P_i sends (Γ_i, σ_i) to P_j.
- Since the adversary $\mathcal{A}$ controls all communication between parties it can intercept and delete the message σ_i generated by P_i. Then $\mathcal{A}$ computes a fresh signature σ^* with internal random coins r^* ($r^* \neq r_i$) such that $\sigma^* = \mathsf{SIG.Sign}(sk_i^{\text{sig}}, \Gamma_i||\mathsf{P}_j) \neq \sigma_i$. Finally, $\mathcal{A}$ sends (Γ_i, σ^*) to P_j.
- Party P_j behaves in the same way. P_j computes (Γ_j, σ_j) and sends them to P_i.
- Computation of session key: parties P_i and P_j accept and compute the same session key $\mathsf{K}_{i,j}$.
- The adversary $\mathcal{A}$ queries Test(π_i^s) and gets K_b from π_i^s. Then, $\mathcal{A}$ queries Reveal(π_j^t) and obtains the session key $\mathsf{K}_{i,j}$. Note that according to the definition of partnership π_i^s and π_t^t are not partnered since $\sigma^* \neq \sigma_i$. Now, $\mathcal{A}$ can easily check if $\mathsf{K}_b = \mathsf{K}_{i,j}$.

C ROBUST COMPOSITION

We believe that our new definition of partnering not only fixes the notion of matching conversations but that it provides a superior notion of partnering. To further support this argument we isolate a fundamental problem with matching conversations showing that protocols secure under this partnering definition cannot provide what we call robust composition. Roughly, this means that the protocol cannot be securely composed with independently generated messages. This culminates in the odd situation that some protocols may be secure under the matching conversations definitions when analyzed in isolation but lose all security, when a single random value is added to the message flow. An illustrative example is the protocol compiler by Cremers and Feltz [20] (Appendix B) that aims to increase the security of an input protocol by making it provide forward secrecy. As detailed in Section 3.2, the resulting protocol is vulnerable to a no-match attack even if the input protocol is secure (for example, consider a protocol that relies on unique messages and primitives only). This shows that the security of protocols analyzed in the MC-based definition depends on the context they are used in, even if the context is independent from the protocol– a theoretically highly unsatisfactory and counter-intuitive feature. Fortunately, we can show that our new notion of partnering does not suffer from this fundamental problems.

Definition C.1 (Robust Augmentation). Let π be a protocol (viewed as the sequence of its messages). We say that protocol π' is a robust augmentation of π if:

- the protocol messages of π form a subsequence (subprotocol) of π', i.e. π' can be computed from π only by adding messages to the protocol (either via appending new values to existing messages flows or by adding entirely new messages flows to the protocol) and
- the session key generated by π' is the session key that is output by the subprotocol π.

Subsequently, we use $\pi' - \pi$ to denote the additional messages that π' contains besides the messages of π.

Definition C.2 (Weak Robust Composition). We say that a security model S supports weak robust composition if for all protocols π and every robust augmentation π' of π we have

- the additional messages $\pi' - \pi$ are computed independently of those of the protocol π
- if π is secure under S then π' is secure under S.

We are now ready to immediately prove the following theorem:

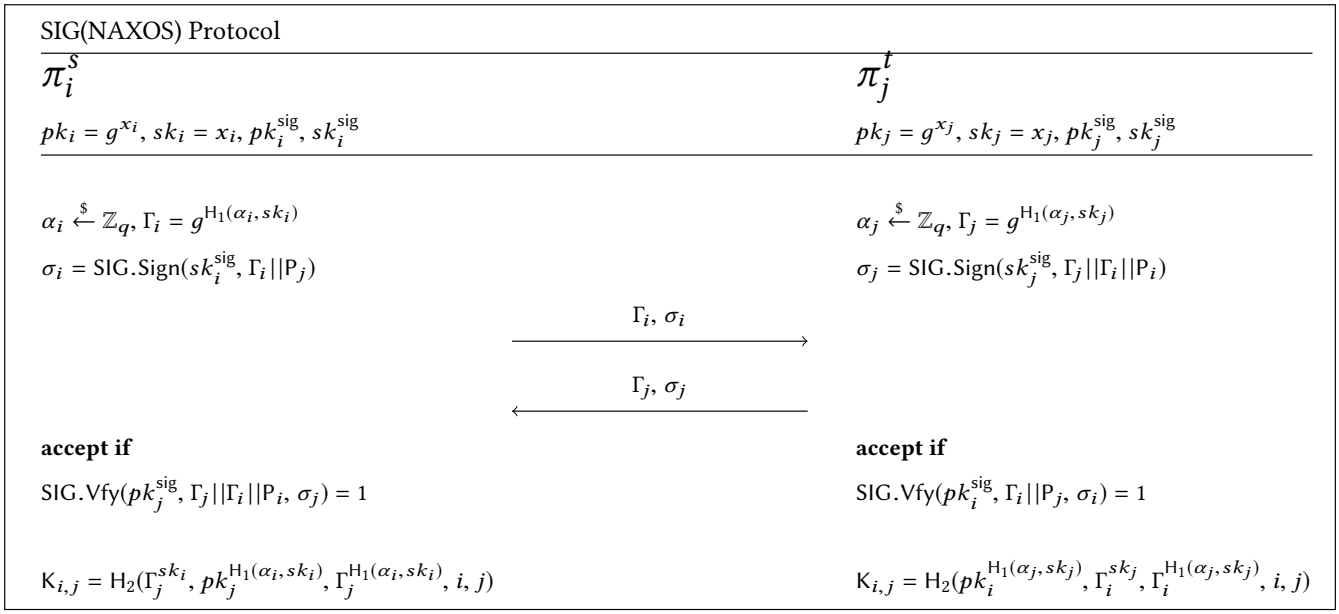

Figure 4: Description of the SIG(NAXOS) Protocol

Theorem C.3. *Any exploitable security model S that defines partnering via matching conversations cannot support weak robust composition.*

Proof. Consider a protocol π that is secure under S. Let π' be a robust augmentation of π that only adds a single random bit b to the first message of the protocol. Then there exists an adversary that can easily launch a no-match attack against π'. All she has to do is to slightly modify the messages sent between two oracles by inverting the bit b in the first protocol message. Obviously, the oracles do not have matching conversations, but they have computed the same session key. This is a no-match attack and since S is exploitable breaks the security of π'. □

This negative result on the composability of protocols under weak robust composition covers not only security models formalizing key indistinguishability but also those that define explicit authentication. Solely the mechanism to exploit the security model is different. In contrast to weak robust composition, our next definition also allows the additional messages to be derived from the public messages of protocol π.

Definition C.4 (Strong Robust Composition). We say that a security model S supports strong robust composition if for all protocols π and every robust augmentation π' of π we have that

- the additional messages $\pi' - \pi$ can be efficiently computed from the messages of the subprotocol π (that have already been sent)
- if π secure under S then π' is secure under S.

Theorem C.5. *Let S be a security model for key exchange protocols that defines security via key indistinguishability and partnering via original key partnering. Then S supports strong robust composition.*

Proof. We can simply reduce security of π' to the protocol π. To this end we assume a simulator that is given protocol π as a black-box. It simulates the protocol π' by efficiently computing all necessary messages in $\pi' - \pi$ from the subprotocol π which by definition is possible. Observe that the definition of partnering in π' corresponds to the definition of partnering in π, i.e. whenever two oracles are partnered with respect to π', they are also partnered with respect to the (black-box) subprotocol π and vice versa. Now, any successful adversary against π' can by definition distinguish the session key of π' from a random value. Since the session key of π' is essentially that of π and since the partnering definition in π' corresponds to that of π we can directly use the the adversary's guess to break the security of π. We note that we can simulate all queries made by the adversary against π' by using the queries for π and since the additional messages can all be computed solely from π. □

As a result, no adversary can augment the protocol with additional messages in a way that makes it insecure. From another perspective, the protocol remains secure in contexts where not only the protocol messages but also other messages are exchanged as long as the additional messages do not rely on the secret information of the protocol.

We can also provide a positive result for key exchange models based on explicit authentication.

Theorem C.6. *Let S be a security model for key exchange protocols that defines security via explicit authentication in the sense of Definition 5.2. Then S supports strong robust composition.*

We can almost re-use the proof of Theorem C.5 and therefore omit it.

D EXAMPLE PROTOCOL AND PROOF SKETCH

To get a better understanding of what impact our new partnering definition has on security proofs that are usually based on matching conversations let us consider an example. In particular consider a protocol where Alice and Bob have long-term signature keys for signature scheme SIG. Alice sends the ephemeral public key g^a, Bob responds with g^b, s_B where $s_B = \text{SIG.Sign}(sk_B, (g^a|g^b))$ and the symbol | indicates a special separator symbol. Alice checks if it holds that $\text{SIG.Vfy}(pk_B, (g^a|g^b), s_B) = 1$. On success it computes $k_A = (g^b)^a$ and sends s_A with $s_A = \text{SIG.Sign}(sk_A, (g^a|g^b))$ to Bob who in turn checks whether $\text{SIG.Vfy}(pk_A, (g^a|g^b), s_A) = 1$. In case the verification of the received signature is positive, Bob computes $k_B = (g^b)^a$.

Let us now sketch a security proof that relies on original key partnering and contrast it with matching conversations. To this end we dive into the Test-session. After excluding collisions when choosing ephemeral public keys we can distinguish several cases:

1) First, we consider the case that the attacker does not make any modifications to the exchanged messages. In this case security follows form the security of the DDH assumption. Namely in one session, the challenger embeds a DDH challenge in g^a and g^b. The key is indistinguishable from random by the DDH assumption which says that given g, g^a, g^b, the value g^{ab} is indistinguishable from g^t for a random t.

2) In the second case we always (only) consider (active) attacks that make Alice or Bob not compute their original key. For the protocol at hand, we consider either a) attackers that modify any of the messages g^a or g^b on transit such that one of the computed keys is distinct from the original key or b) we consider attacks that modify the signatures so that any of them becomes invalid (resulting in no computation of a session key of the receiving oracle at all.). In case a) we can use such an attacker to break the security of the signature schemes. This is because any attacker that modifies any of those two message has to produce a valid signature on the message (which includes fresh ephemeral public keys) as well to make the receiver accept it and generate a key. Using the signature oracle in the security game we can easily simulate all other sessions. In contrast, analyzing case b) is actually not necessary in a formal proof and we include it only for illustrative purposes. In fact, the security definition requires that the Test-oracle has computed a key. Moreover if Bob's oracle does not compute a key at all, this only increases the security for Alice's oracle: Bob's oracle does not store any information that may help to distinguish Alice's key.

3) The third case is also only considered as an illustrative argument for the purpose of this paper. In actual security proofs using original key partnering it would not appear. In this case the attacker modifies any message such that as a result Alice and Bob compute their original key. One attack is to modify the signatures s_A or s_B such that as a result, the verification is still positive. (Another way would be to change the representation of the ephemeral public key for example to send $g^a - p$ instead of g^a in case $< g >$ is a subgroup of $\mathbb{Z}_p^*$. This works if the signature scheme works on messages in $< g >$ but does not apply an interval test, see Section E.1.) Here the difference of our security notion with respect to matching conversations comes into play: with matching conversations the two oracles would not be partnered although they both compute the same session key. That would allow for an exploitable no-match attack – revealing Bob's session key would trivially help to distinguish Alice's key from random. Using original key partnering the two oracles remain to be partnered even if the signatures are modified since both oracles compute their original key. Importantly, observe that this is crucial when analyzing KCI attacks and perfect forward secrecy. In these cases the attacker has access to sk_A before Alice's last message arrives at Bob and the attacker can thus easily compute a new signature $s'_A = \text{SIG.Sign}(sk_A, (g^a|g^b))$ that Bob will accept. So in sum security defined under matching conversations would fail to reason about this protocol under these strong AKE definitions. Even worse, the protocol could not be proven secure unless the signature scheme fulfills the notion of strong unforgeability even when not considering KCI attacks or perfect forward secrecy, i.e. when proving classical key indistinguishability. However, under our new notion, the protocol can be proven secure in even this strong model. Intuitively, this is legitimate since Alice and Bob compute the same key that would be computed in case of no modification at all (case 1).

E DESIGN SOLUTIONS

In the following, we propose two approaches to armor protocols against no-match attacks.

E.1 Uniqueness

Having ruled out the usefulness of deterministically computed values v to prevent no-match attacks in general and strong security for no-match attacks with advice in particular, the open question is what properties of the algorithms that compute v are required to actually thwart no-match attacks. What we want is that the adversary may not present a *distinct* v' that behaves like the original v *independent of whether the adversary has access to the secret key or not.* This can be captured by the notion of uniqueness. For example, in a unique signature scheme, for every public key, there is only one signature per message that passes the signature verification. Our definition captures general verification functions.

Definition E.1 (Unique Verification). A verification algorithm VRF defined over $(\mathcal{K}, \mathcal{M}, \mathcal{V}, \mathcal{X}^{aux})$ is an efficient algorithm $\text{VRF}: \mathcal{K} \times \mathcal{M} \times \mathcal{V} \times \mathcal{X}^{aux} \to \{0,1\}$ described as a deterministic Turing Machine. The set $\mathcal{K}$ is called the key space, $\mathcal{M}$ is called the message space, $\mathcal{V} = \{0,1\}^*$ is called the verification space and $\mathcal{X}^{aux}$ is called the auxiliary information space (which may possibly be empty). We say that VRF provides unique verification if for all $k \in \mathcal{K}$, all messages $m \in \mathcal{M}$ and all $\text{aux} \in \mathcal{X}^{aux}$ we always have that

$$|\{v \in \mathcal{V} | \text{VRF}(k, m, v, \text{aux}) = 1\}| \leq 1.$$

In the above definition we generally consider algorithms that are keyed with some key k. Depending on the primitive this key can be a symmetric, public, or secret key.

We clarify that the definitions of deterministic signatures and unique signatures concentrate on properties of distinct algorithms. Whereas a deterministic signature scheme refers to a signature scheme with a deterministic *signing* algorithm, unique signatures refer to signature schemes whose *verification* algorithm meets the uniqueness property.

In the next step, we may explore ways to substitute cryptographic primitives with unique versions. This has the advantage that protocol implementations where the primitives can be agreed upon by the communication partners can remain secure under matching conversations without any protocol modification just via a suitable instantiation of the primitive. Important results on common building blocks for security protocols show that while it is relative simple to obtain unique MACs (and unique symmetric encryption).[15], it is much more difficult to construct unique signatures (than non-unique ones)[16]. For public key encryption in contrast, it is impossible to do so since even mere IND-CPA encryption requires the scheme to be probabilistic [26].

E.2 Unique Message Representations and Group Membership Testing.

We observe that, as a consequence of our no-match attacks and unless other means do not protect protocols against them, parties should generally use unique representations of *all* messages. Of course, not only need the protocol participants use a unique set of representations but they also have to check whether the received values actually belong to this set. This is particularly important for group membership tests that are implemented as simple exponentiation. Popular ways for Bob to check if Alice's message a is in a subgroup of prime order q is to check if $a^q = 1 \bmod p$ or $a^r \neq 1 \bmod p$ where $r = |\mathbb{Z}_p^*|/q$ is the co-factor. These checks are often called *ephemeral public key validation*.

We remark that mere ephemeral public key validation is not sufficient to protect the protocol against no-match attacks since any adversary can easily sent $a' = a + p \neq a$ (over the integers) instead of a to Bob and Bob will accept both values since $a' = a \bmod p$.

The problem is that ephemeral public key validation as sketched above does not check for uniqueness. However, if Bob always checks if $0 \leq a \leq p-1$ then it would not accept a' and our no-match attack is thwarted. We call this additional check *interval test*.

Also observe that in case $a = g^x \bmod p$ is Alice's ephemeral key that Bob combines with his ephemeral secret key $y \in \mathbb{Z}_q$ to compute a common secret $a^y \bmod p$, we have that $a'^y \bmod p = a^y \bmod p$ although $a \neq a'$. So the common secrets computed from a, y and a', y are equal. The substitution of a by a' can be *combined* with a no-match attack with advice which helps the adversary to substitute a signature on a with one of a'. Note that the resulting attack would still work if the signature scheme offers unique verification! (We believe that such a combination is very hard to spot in a security proof. At the same time it further exemplifies the diversity of no-match attacks.) The resulting attack can be used to break the NAXOS protocol [37] or the KEA+ protocol that was published at PKC'06 [38] *if these protocols do not apply an interval test* because the final session key is not derived from a but only from $a^y \bmod p$. Finally, a similar attack can be launched against a proposal to instantiate the symmetric encryption system in [5] by encrypting message m with password pw as $c = h(pw) \cdot m$, where h is a hash function that is modeled as a random oracle and the arithmetic is in the underlying group. When for example implemented over a subgroup of $\mathbb{Z}_p^*$ and not applying an interval test it is easy to come up with a second string that when decrypted maps to the same message using the above technique.

We emphasize that these papers, like most papers on key exchange protocols, are unclear on how they implement group membership tests exactly. Our no-match attacks convincingly show that mere ephemeral key validation is in some scenarios not sufficient.

E.3 Protocol Compiler

As a third solution we propose the following efficient transformation that makes the computation of the session key critically depend on any message bit of the transcript. Assume we have a t-move key exchange protocol π that is executed between Alice and Bob. Let us now describe a new key exchange protocol π' that is constructed from π and a hash function H. Let $m_j^{(i)}$ for $i \in \mathbb{N}, t \geq i \geq 1$ be the i-th message sent or received by party $j \in \{\mathsf{A}, \mathsf{B}\}$. Suppose $\mathsf{H} : \{0,1\}^* \to \{0,1\}^l$ is a hash function which is modeled as a random oracle in the security proof. Let K_j be the final session key that is output by $j \in \{\mathsf{A}, \mathsf{B}\}$. In our compiler, each party j will maintain a (small) state variable c_j that is updated after each protocol move. We now describe how each party j can recursively compute the final state $c_j^{(t)}$ that in turn is used to compute the session key K_j.

- Define $c_j^{(0)}$ to some global constant c.
- For each sent or received message $m_j^{(i)}$ computes the value $c_j^{(i)} = \mathsf{H}(c_j^{(i-1)}, m_j^{(i)})$.
- The session key is computed as $\mathsf{K}_j^* = \mathsf{H}(c_j^{(t)}, \mathsf{K}_j)$.

The advantage of our compiler is its generality. In particular, it does not require the protocol to fulfill *any* security requirements and excludes no-match attacks – even for insecure protocols (with overwhelming probability). This is because any modification made to the messages exchanged between two oracles results in distinct inputs to the final random oracle call. The following lemma captures this intuition.

Lemma E.2. *Assume that in the run of a key exchange protocol the adversary launches an active attack on the modified protocol π' executed between two oracles. Then, with overwhelming probability $1 - 2^{-l}$, the two oracles do not compute the same session key K^*.*

Proof. Each modification will make Bob's oracle receive at least one message that is distinct from what Alice's oracle has sent (or vice versa). Now the output of the corresponding random oracle call is random and independent from the output produced by Alice. Because of the continual feedback of previous values into the next call of the random oracle the session keys are independent and random as well. So they differ with all but negligible probability $1 - 2^{-l}$. □

As an advantage, the above compiler is computationally very efficient. However, it does modify the key derivation function of the protocol.

[15] If necessary use a PRF on the message and some secret symmetric key to derive the input randomness and make the scheme *deterministic*. To make the scheme *unique*, modify the verification procedure to just re-run the tagging (or encryption) algorithm and check for equality with the received value. In this way only a single value will be accepted.

[16] While plain digital signatures can be constructed from one-way functions only, unique signatures require even more than ideal trapdoor permutations [23, 28].