Session-Key Generation using Human Passwords Only*

Oded Goldreich[†]
Department of Computer Science
Weizmann Institute of Science
Rehovot, Israel.
oded@wisdom.weizmann.ac.il

Yehuda Lindell[‡]
IBM T.J.Watson Research
19 Skyline Drive, Hawthorne
New York 10532, USA.
lindell@us.ibm.com

July 10, 2003

Abstract

We present session-key generation protocols in a model where the legitimate parties share *only* a human-memorizable password, and there is no additional setup assumption in the network. Our protocol is proven secure under the assumption that trapdoor permutations exist. The security guarantee holds with respect to probabilistic polynomial-time adversaries that control the communication channel (between the parties), and may omit, insert and modify messages at their choice. Loosely speaking, the effect of such an adversary that attacks an execution of our protocol is comparable to an attack in which an adversary is only allowed to make a constant number of queries of the form "is w the password of Party A". We stress that the result holds also in case the passwords are selected at random from a small dictionary so that it is feasible (for the adversary) to scan the entire directory. We note that prior to our result, it was not known whether or not such protocols were attainable without the use of random oracles or additional setup assumptions.

Area: Cryptography.

Additional Keywords: Session-key generation (authenticated key-exchange), mutual authentication protocols, human-memorizable passwords, secure two-party computation, non-malleable commitments, zero-knowledge proofs, pseudorandom generators and functions, message authentication schemes.

*An extended abstract of this work appeared in *Crypto 2001*.

[†]Supported by MINERVA Foundation, Germany.

[‡]This work was carried out while the author was at the Weizmann Institute of Science.

Contents

1 Introduction

This work deals with the oldest and probably most important problem of cryptography: enabling *private and reliable* communication among parties that use a public communication channel. Loosely speaking, *privacy* means that nobody besides the legitimate communicators may learn the data communicated, and *reliability* (or *integrity*) means that nobody may modify the contents of the data communicated (without the receiver detecting this fact). Needless to say, a vast amount of research has been invested in this problem. Our contribution refers to a difficult and yet natural setting of two parameters of the problem: the *adversaries* and the *initial set-up*.

We consider only probabilistic polynomial-time adversaries. Still, even within this framework, an important distinction refers to the type of adversaries one wishes to protect against: *passive* adversaries only eavesdrop the channel, whereas *active* adversaries may also omit, insert and modify messages sent over the channel. Clearly, reliability is a problem only with respect to active adversaries (and holds by definition w.r.t passive adversaries). *We focus on active adversaries.*

The second parameter mentioned above is the initial set-up assumptions. Some assumption of this form must exist or else there is no difference between the legitimate communicators, called Alice and Bob, and the adversary (which may otherwise initiate a conversation with Alice pretending to be Bob). We list some popular initial set-up assumptions and briefly discuss what is known about them.

Public-key infrastructure: Here one assumes that each party has generated a secret-key and deposited a corresponding public-key with some trusted server(s). The latter server(s) may be accessed at any time by any user.

It is easy to establish private and reliable communication in this model (by a straightforward use of the public-key schemes, cf. [19, 48]). (However, even in this case (see, e.g. [45, 7]), one may want to establish "session-keys" as discussed below.)

Shared (high-quality) secret keys: By *high-quality keys* we mean strings coming from distributions of high entropy (e.g., uniformly chosen 128-bit long strings, uniformly chosen 1024-bit primes, etc). Furthermore, these keys are selected by a suitable program, and cannot be memorized by humans.

In case a pair of parties shares such a key, they can conduct private and reliable communication (either directly (cf. [12, 52, 27]) or by first establishing a session-key (cf. [7])).

Shared (low-quality) secret passwords: In contrast to high-quality keys, *passwords* are strings that may be easily selected, memorized and typed-in by humans. An illustrating (and simplified) example is the case in which the password is selected uniformly from a relatively small dictionary; that is, the password is uniformly distributed in $\mathcal{D} \subset \{0,1\}^n$, where $|\mathcal{D}| = \mathrm{poly}(n)$.

Note that using such a password in the role of a cryptographic key (in schemes as mentioned above) will yield a totally insecure scheme. A more significant observation is that the adversary may try to guess the password, and initiate a conversation with Alice pretending to be Bob and using the guessed password. So nothing can prevent the adversary from successfully impersonating Bob with probability $1/|\mathcal{D}|$. *But can we limit the adversary's success to about this much?*

The latter question is the focus of this paper.

Session-keys: The problem of establishing private and reliable communication is commonly reduced to the problem of generating a secure session-key (a.k.a "authenticated key exchange").

Loosely speaking, one seeks a protocol by which Alice and Bob may agree on a key (to be used throughout the rest of the current communication session) so that this key will remain unknown to the adversary.[1] Of course, the adversary may prevent such agreement (by simply blocking all communication), but this will be detected by either Alice or Bob.

1.1 What Security May be Achieved Based on Passwords

Let us consider the related (although seemingly easier) task of *mutual authentication*. Here Alice and Bob merely want to establish that they are talking to one another. Repeating an observation made above, we note that if the adversary initiates $t \leq |\mathcal{D}|$ instances of the mutual authentication protocol, guessing a different password in each of them, then with probability $t/|\mathcal{D}|$ it will succeed in impersonating Alice to Bob (and furthermore find the password). The question posed above is rephrased here as follows:

> *Can one construct a password-based scheme in which the success probability of any probabilistic polynomial-time impersonation attack is bounded by $O(t/|\mathcal{D}|) + \mu(n)$, where t is the number of sessions initiated by the adversary, and $\mu(n)$ is a negligible function in the security parameter n?*

We resolve the above question in the affirmative. That is, assuming the existence of trapdoor one-way permutations, *we prove that schemes as above do exist* (for any $\mathcal{D}$ and specifically for $|\mathcal{D}| = \text{poly}(n)$). Our proof is constructive. We actually provide a protocol of comparable security for the more demanding goal of *authenticated session-key generation*.

Password-based authenticated session-key generation: Our definition for the task of authenticated session-key generation is based on the simulation paradigm. That is, we require that a secure protocol emulates an ideal execution of a session-key generation protocol (cf. [2, 41, 15]). In such an ideal execution, a trusted party hands identical, uniformly distributed session-keys to the honest parties. The only power given to the adversary in this ideal model is to prevent the trusted party from handing keys to one of the parties. (We stress that, in this ideal model, the adversary learns *nothing* of the parties' joint password or output session-key).

Next, we consider a real execution of a protocol (where there is no trusted party and the adversary has full control over the communication channel between the honest parties). In general, a protocol is said to be secure if real-model adversaries can be emulated in the ideal-model such that the output distributions are computationally indistinguishable. Since in a password-only setting the adversary can always succeed with probability $1/|\mathcal{D}|$, it is impossible to achieve computational indistinguishability between the real model and above-described ideal model (where the adversary has zero probability of success). Therefore, in the context of a password-only setting, an authenticated session-key generation protocol is said to be secure if the above-mentioned ideal-model emulation results in an output distribution that can be distinguished from a real execution by (a gap of) at most $O(1/|\mathcal{D}|) + \mu(n)$.

Main result (informally stated): *Assuming the existence of trapdoor one-way permutations, there exists a secure authenticated session-key generation protocol in the password-only setting.*

We stress that the above (informal) definition implies the intuitive properties of authenticated session-key generation (e.g., security of the generated session-key and of the initial password). In

[1] We stress that many famous key-exchange protocols, such as the one of Diffie and Hellman [19], refer to a passive adversary. In contrast, this paper refers to active adversaries.

particular, the output session-key can be distinguished from a random key by (a gap of) at most $O(1/|\mathcal{D}|) + \mu(n)$. This implies that when using the session-key as a key to a MAC, for example, the probability that any polynomial-time adversary can generate a valid MAC-tag to a message not sent by the legitimate party is small (i.e., $O(1/|\mathcal{D}|) + \mu(n)$). We stress that the session-key can be used for polynomially-many MACs and the probability that the adversary will forge even one message still remains bounded by $O(1/|\mathcal{D}|) + \mu(n)$. Likewise, when using the session key for private-key encryption, the probability that the adversary learns anything about the encrypted messages is small. That is, for every partial-information function, the adversary can guess the value of the function applied to the messages with $O(1/|\mathcal{D}|) + \mu(n)$ advantage over the a-priori probability. This success probability is exactly the same (ignoring $\mu(n)$ and constant factors) as for the naive adversary who just attempts to guess the password and succeeds with probability $1/|\mathcal{D}|$. See Section 2.3.3 for more discussion.

In addition to the above-described security of the session-key, the definition guarantees that the distinguishing gap between the parties' joint password and a uniformly distributed element in $\mathcal{D}$ is at most $O(1/|\mathcal{D}|) + \mu(n)$. (As we have mentioned, the fact that the adversary can distinguish with gap $O(1/|\mathcal{D}|)$ is an inherent limitation of password-based security.) The parties are also guaranteed that, except with probability $O(1/|\mathcal{D}|)+\mu(n)$, they either end-up with the same session-key or detect that their communication has been tampered with. Our definition also implies additional desirable properties of session-key protocols such as forward secrecy and security in the case of session-key loss (or known-key attacks). Furthermore, our protocol provides improved (i.e., negligible gap) security in case the adversary only eavesdrops the communication (during the protocol execution).

We mention that a suitable level of indistinguishability (of the real and ideal executions) holds when t sessions (referring to the same password) are conducted sequentially: in this case the distinguishing gap is $O(t/|\mathcal{D}|) + \mu(n)$ rather than $O(1/|\mathcal{D}|) + \mu(n)$ (which again is optimal). This holds also when any (polynomial) number of *other sessions w.r.t independently distributed passwords* are conducted concurrently to the above t sessions.

Caveat: Our protocol is proven secure only when assuming that the *same pair of parties* (using the same password) does not conduct several *concurrent* executions of the protocol. We stress that concurrent sessions of other pairs of parties (or of the same pair using a different password), are allowed. See further discussion in Sections 1.4 and 2.5.

1.2 Comparison to Other Work

The design of secure mutual authentication and key-exchange protocols is a major effort of the applied cryptography community. In particular, much effort has been directed towards the design of *password-based* schemes that should withstand active attacks.[2] An important restricted case of the mutual authentication problem is the *asymmetric* case in which a human user authenticates himself to a server in order to *access* some service. The design of secure *authentication* mechanisms based only on passwords is widely recognized as a central problem of computer practice and as such has received much attention.

The first protocol suggested for password-based session-key generation was by Bellovin and Merritt [8]. This work was very influential and became the basis for much future work in this

[2] A specific focus of this research has been on preventing *off-line dictionary attacks*. In such an off-line attack, the adversary records its view from past protocol executions and then scans the dictionary for a password consistent with this view. If checking consistency in this way is possible and the dictionary is small, then the adversary can derive the correct password. Clearly, a secure session-key generation protocol (as informally defined above) withstands any off-line dictionary attack.

area [9, 50, 34, 39, 46, 51]. However, these protocols have not been proven secure and their conjectured security is based on mere heuristic arguments. Despite the strong need for secure password-based protocols, the problem was not treated rigorously until quite recently. For a survey of works and techniques related to password authentication, see [40, 36] (a brief survey can be found in [33]).

A first rigorous treatment of the password-based *authentication* problem was provided by Halevi and Krawczyk [33]. They actually considered an *asymmetric* hybrid model in which one party (the server) may hold a high-quality key and the other party (the human) may only hold a password. The human is also assumed to have secure access to a corresponding public-key of the server (either by reliable access to a reliable server or by keeping a "digest" of that public-key, which they call a *public-password*).[3] The Halevi–Krawczyk model capitalizes on the asymmetry of the authentication setting, and is inapplicable to settings in which communication has to be established between two humans (rather than a human and a server). Furthermore, requiring the human to keep the unmemorizable public-password (although not secretly) is undesirable. Finally, we stress that the Halevi–Krawczyk model is a *hybrid* of the "shared-key model" and the "shared-password model" (and so their results don't apply to the "shared-password model"). Thus, it is of both theoretical and practical interest to answer the original question as posed above (i.e., without the public-password relaxation): *Is it possible to implement a secure authentication mechanism* (*and key-exchange*) *based only on passwords*?

Positive answers to the original problem have been provided *in the random oracle model.* In this model, all parties are assumed to have oracle access to a totally random (universal) function [5]. Secure (password-based) authenticated key-exchange schemes in the random oracle model were presented in [4, 14]. The common interpretation of such results is that *security is "likely" to hold* even if the random oracle is replaced by a ("reasonable") concrete function known explicitly to all parties.[4] We warn that this interpretation is not supported by any sound reasoning. Furthermore, as pointed out in [17], there exist protocols that are secure in the random oracle model but become insecure if the random function is replaced by *any* specific function (or even a function uniformly selected from any family of functions).

To summarize, this paper is the first to present session-key generation (as well as mutual authentication) protocols *based only on passwords* (i.e., in the shared-password model), using only standard cryptographic assumptions (e.g., the existence of trapdoor one-way permutations, which in turn follows from the intractability assumption regarding integer factorization). We stress that prior to this work it was not clear whether such protocols exist at all (i.e., outside of the random oracle model).

Independent related work. Independently of our work, Katz, Ostrovsky and Yung [35] presented a protocol for the task of session-key generation based on passwords. Their protocol is incomparable to ours: it uses a stronger set-up assumption and a stronger intractability assumption, but yields a seemingly practical protocol that is secure in a stronger concurrent sense. Specifically:

[3]The public-password is not memorizable by humans, and the human is supposed to carry a record of it. The good point is that this record need not be kept secret (but rather merely needs to be kept reliably). Furthermore, in the Halevi–Krawczyk protocol, the human is never asked to type the public-password; it is only asked to compare this password to a string sent by the server during the protocol.

[4]An alternative interpretation is to view the random oracle model literally. That is, assume that such oracle access is available to all parties via some trusted third party. However, in such a case, we are no longer in the "trust nobody" model in which the question was posed.

- Most importantly, Katz et al. [35] use a stronger set-up assumption than us. In addition to joint passwords, they require that all parties have access to a common reference string, chosen by some trusted third party. Although this is a stronger assumption than that of our password-only model, it is still significantly weaker than other models that have been studied (like, for example, the Halevi–Krawczyk model).[5]
- Their protocol is proven secure under a *specific* assumption. Specifically, they use the Decisional Diffie-Hellman assumption, which seems stronger than more standard assumptions such as the intractability of factoring and of extracting discrete logarithms. In contrast, we use a general complexity assumption (i.e., the existence of trapdoor permutations).
- Their protocol is highly efficient and could even be used in a practical setting. In contrast, our protocol is unsuitable for practical use, although it may eventually lead to practical consequences.
- Their protocol is secure in an unrestricted concurrent setting, whereas our protocol is shown to be secure only when concurrent executions are not allowed to use the *same* password (see Section 2.5).

Key exchange protocols. In the above description of prior work, we have focused only on papers dealing with the issue of *password-based* authentication and key-exchange. We note that there has been much work on this problem in the setting where the parties share high entropy keys, both with respect to determining appropriate definitions and constructing secure protocols. See [40, Chapter 12] for a survey of some of these works.

Necessary conditions for mutual authentication: Halevi and Krawczyk [33] proved that mutual authentication in the shared-password model implies (unauthenticated) secret-key exchange, which in turn implies one-way functions. Subsequently, Boyarsky [13] pointed out that, in the shared-password model, mutual authentication implies oblivious transfer. Thus finding a solution to this problem that relies on only "black-box" use of one-way functions is hard (in particular, it would constitute a proof that $\mathcal{P} \neq \mathcal{NP}$).[6]

1.3 Techniques

One central idea underlying our protocol is due to Naor and Pinkas [44]. They suggested the following protocol for the case of *passive* adversaries, using a secure protocol for polynomial evaluation.[7] In order to generate a session-key, party A first chooses a random linear polynomial $Q(\cdot)$ over a large field (which contains the dictionary of passwords). Next, A and B execute a secure polynomial

[5] We remark that the setup assumption of a common reference string is practical in some settings, but very restrictive in others. For example, a company that wishes to implement secure login for its employees would be trusted to correctly choose the reference string. Furthermore, within such a closed setting, this string could be securely distributed to all employees. However, in a general setting, such trust is highly undesirable. This is especially true since in the protocol of [35], if an adversarial party chooses the reference string, it can learn all the parties' passwords by merely eavesdropping on the communication.

[6] Oblivious transfer is known to imply (unauthenticated) secret-key exchange [37]. On the other hand, Gertner et al. [24] have shown that secret-key exchange does *not* imply oblivious transfer under black-box reductions.

[7] In the polynomial evaluation functionality, party A has a polynomial $Q(\cdot)$ over some finite field and Party B has an element x of the field. The evaluation is such that A learns nothing, and B learns $Q(x)$; i.e., the functionality is defined by $(Q, x) \mapsto (\lambda, Q(x))$.

evaluation in which B obtains $Q(w)$, where w is their joint password. The session-key is then set to equal $Q(w)$.

In [13] it was suggested to make the above protocol secure against *active* adversaries, by using non-malleable commitments. This suggestion was re-iterated to us by Moni Naor, and in fact our work grew out of his suggestion. In order to obtain a protocol secure against active adversaries, we augment the above-mentioned protocol of [44] by several additional mechanisms. Indeed, we use non-malleable commitments [21], but in addition we also use a *specific* zero-knowledge proof [47], ordinary commitment schemes [10], a *specific* pseudorandom generator (of [12, 52, 11]), and a message authentication scheme (MAC). The analysis of the resulting protocol is very complicated, *even when the adversary initiates a single session.* As explained below, we believe that these complications are unavoidable given the current state-of-art regarding *concurrent execution* of protocols.

Although not explicit in the problem statement, the problem we deal with actually concerns *concurrent* executions of a protocol. Even in case the adversary attacks a single session among two legitimate parties, its ability to modify messages means that it may actually conduct two *concurrent* executions of the protocol (one with each party).[8] Concurrent executions of some protocols were analyzed in the past, but these were relatively simple protocols. Although the high-level structure of our protocol can be simply stated in terms of a small number of modules, the currently known implementations of some of these modules are quite complex. Furthermore, these implementations are NOT known to be secure when two copies are executed *concurrently.* Thus, at the current state of affairs, the analysis cannot proceed by applying some composition theorems to (two-party) protocols satisfying some concurrent-security properties (because suitable concurrently-secure protocols and composition theorems are currently unknown). Instead, we have to analyze our protocol directly. We do so by reducing the analysis of (two concurrent executions of) our protocol to the analysis of non-concurrent executions of *related* protocols. Specifically, we show how a successful adversary in the concurrent setting contradicts the security requirements in the non-concurrent setting. Such "reductions" are performed several times, each time establishing some property of the original protocol. Typically, the property refers to one of the two concurrent executions, and it is shown to hold even if the adversary is given some secrets of the legitimate party in the second execution. This is shown by giving these secrets to the adversary, enabling it to effectively emulate the second execution internally. Thus, only the first execution remains and the relevant property is proven (in this standard non-concurrent setting). We stress that this procedure is not applied "generically", but is rather applied to the specific protocol we analyze while taking advantage of its specific structure (where some of this structure was designed so to facilitate our proof). Thus, our analysis is ad-hoc in nature, but still we believe that it can eventually lead to a methodology of analyzing concurrent executions of (two-party) protocols.

1.4 Discussion

We view our work as a theoretical study of the *very possibility* of achieving private and reliable communication among parties that share only a secret (low-quality) password and communicate over a channel that is controlled by an active adversary. Our main result is a demonstration of the *feasibility* of this task. That is, we demonstrate the *feasibility* of performing *session-key generation based only on* (low-quality) *passwords.* Doing so, this work is merely the first (rigorous) step in a research project directed towards providing a good solution to this practical problem. We discuss

[8] Specifically, the adversary may execute the protocol with Alice while claiming to be Bob, concurrently to executing the protocol with Bob while claiming to be Alice, where these two executions refer to the same joint Alice–Bob password.

two aspects of this project that require further study.

Concurrent executions for the same pair of parties: Our protocol is proven secure only when the *same pair of parties* (using the same password) does not conduct several *concurrent* executions of the protocol. Thus, actual use of our protocol requires a mechanism for ensuring that the same pair of parties execute the protocol strictly sequentially. A simple timing mechanism enforcing the above (and using local clocks only) is as follows. Let Δ be greater than the period of time that suffices for completing an execution of the protocol under "ordinary" circumstances. Then, if an execution takes longer than Δ units of time, the execution is timed-out (with the parties aborting). Furthermore, parties wait for at least Δ units of time between consecutive protocol executions. It is easy to see that this enforces strict sequentiality of executions. Indeed, it is desirable not to employ such a timing mechanism, and to prove that security holds also when many executions are conducted concurrently using the same password. Nevertheless, there are settings where such a mechanism can be used. See Section 2.5 for further details.

We stress that the above limitation relates only to the same pair parties using the same password. There is *no limitation* on the concurrency of executions involving *different* pairs of parties (or the same pair of parties and different passwords).

We note that the protocols of [4, 14] do not suffer from this limitation. However, as we have mentioned, these protocols are proven secure in the random oracle model (and thus the proofs of security are only heuristic).

Efficiency: It is indeed desirable to have more efficient protocols than the one presented here. Some of our techniques may be useful towards this goal.

1.5 Organization

In Section 2 we present the formal setting and state our results. Our protocol for password-based session-key generation is presented in Section 3. In Section 4 we present proof sketches of the main claims used in the analysis of our protocol, and derive our main result based on these claims. The full proofs of these claims are given in Sections 5 to 8. We note that, except in one case, the proof sketches (presented in Section 4) are rather detailed, and demonstrate our main techniques. Thus, we believe that a reading of the paper until the end of Section 4 suffices for obtaining a good understanding of the results presented and the proof techniques involved. The exceptional case, mentioned above, is the proof of Lemma 4.6, which is given in Section 6.1 and is far more complex than the corresponding proof sketch. Thus, we also recommend to read Section 6.1.

In Appendix A we recall the definitions of secure two-party computation as well as the various cryptographic tools used in our protocol.

2 Formal Setting

In this section we present notation and definitions that are specific to our setting, culminating in a definition of Authenticated Session-Key Generation. Given these, we state our main result.

2.1 Basic Notations

- Typically, C denotes the *channel* (i.e., a probabilistic polynomial-time adversary) through which parties A and B communicate. We adopt the notation of Bellare and Rogaway [6] and model the communication by giving C oracle access to A and B. We stress that, as in [6], these oracles have memory and model parties who participate in a session-key generation protocol. Unlike in [6], when A and B share a single password, C has oracle access to only a single copy of each party.

We denote by $C^{A(x),B(y)}(\sigma)$, an execution of C (with auxiliary input σ) when it communicates with A and B, holding respective inputs x and y. Channel C's output from this execution is denoted by $\mathsf{output}\Big(C^{A(x),B(y)}(\sigma)\Big)$.

- The password dictionary is denoted by $\mathcal{D} \subseteq \{0,1\}^n$, and is fixed throughout the entire discussion. We assume that this dictionary can be sampled in probabilistic polynomial-time. We denote $\epsilon = \frac{1}{|\mathcal{D}|}$.
- We denote by U_n a random variable that is uniformly distributed over the set of strings of length n.
- For a set S, we denote $x \in_R S$ when x is chosen *uniformly* from S.
- We use "ppt" as shorthand for probabilistic polynomial time.
- An unspecified negligible function is denoted by $\mu(n)$. That is, for every polynomial $p(\cdot)$ and for all sufficiently large n's, $\mu(n) < \frac{1}{p(n)}$. For functions f and g (from the integers to the reals), we denote $f \approx g$ if $|f(n) - g(n)| < \mu(n)$.
- Finally, we denote computational indistinguishability by $\stackrel{\rm c}{\equiv}$.

A security parameter n is often implicit in our notations and discussions. Thus, for example, by the notation $\mathcal{D}$ for the dictionary, our intention is really $\mathcal{D}_n$ (where $\mathcal{D}_n \subseteq \{0,1\}^n$). Recall that we make no assumptions regarding the size of $\mathcal{D}_n$, and in particular it may be polynomial in n.

Uniform or non-uniform model of computation. Some of the definitions in Appendix A are presented in the non-uniform model of computation. Furthermore, a number of our proofs appear to be in the non-uniform complexity model, but can actually be carried out in the uniform model. Thus, a straightforward reading of our proofs makes our main result hold assuming the existence of trapdoor permutations that cannot be inverted by polynomial-size circuits. However, realizing that the analogous uniform-complexity definitions and proofs hold, it follows that our main result can be achieved under the analogous uniform assumption.

2.2 $(1-\epsilon)$-Indistinguishability and Pseudorandomness

Extending the standard definition of computational indistinguishability [31, 52], we define the concept of $(1-\epsilon)$-indistinguishability. Loosely speaking, two ensembles are $(1-\epsilon)$-indistinguishable if for every ppt machine, the probability of distinguishing between them (via a single sample) is at most negligibly greater than ϵ.

Definition 2.1 ($(1-\epsilon)$-indistinguishability): *Let $\epsilon : \mathsf{N} \to [0,1]$ be a function, and let $\{X_n\}_{n\in\mathsf{N}}$ and $\{Y_n\}_{n\in\mathsf{N}}$ be probability ensembles, so that for any n the distribution X_n (resp., Y_n) ranges over strings of length polynomial in n. We say that the ensembles are* $(1-\epsilon)$-indistinguishable, *denoted $\{X_n\}_{n\in\mathsf{N}} \stackrel{\epsilon}{\equiv} \{Y_n\}_{n\in\mathsf{N}}$, if for every probabilistic polynomial time distinguisher D, and all auxiliary information $z \in \{0,1\}^{\mathrm{poly}(n)}$*

$$|\Pr[D(X_n, 1^n, z) = 1] - \Pr[D(Y_n, 1^n, z) = 1]| < \epsilon(n) + \mu(n)$$

The standard notion of computational indistinguishability coincides with 1-indistinguishability. Note that $(1-\epsilon)$-indistinguishability is not preserved under multiple samples (even for efficiently constructible ensembles); however (for efficiently constructible ensembles), $(1-\epsilon)$-indistinguishability implies $(1-m\epsilon)$-indistinguishability of sequences of m samples.

Definition 2.2 ($(1-\epsilon)$-pseudorandomness): *We say that $\{X_n\}_{n\in\mathsf{N}}$ is $(1-\epsilon)$-*pseudorandom *if it is $(1-\epsilon)$-indistinguishable from $\{U_n\}_{n\in\mathsf{N}}$.*

Similarly, extending the definition of pseudorandom functions [27], we define $(1-\epsilon)$-pseudorandom functions as follows.

Definition 2.3 ($(1-\epsilon)$-pseudorandom function ensembles): *Let $F=\{F_n\}_{n\in\mathsf{N}}$ be a function ensemble where for every n, the random variable F_n assumes values in the set of functions mapping n-bit long strings to n-bit long strings. Let $H=\{H_n\}_{n\in\mathsf{N}}$ be the uniform function ensemble in which H_n is uniformly distributed over the set of all functions mapping n-bit long strings to n-bit long strings. Then, a function ensemble $F=\{F_n\}_{n\in\mathsf{N}}$ is called $(1-\epsilon)$-*pseudorandom *if for every probabilistic polynomial-time oracle machine D, and all auxiliary information $z\in\{0,1\}^{\mathrm{poly}(n)}$*

$$\left|\Pr[D^{F_n}(1^n,z)=1]-\Pr[D^{H_n}(1^n,z)=1]\right|<\epsilon(n)+\mu(n)$$

2.3 Authenticated Session-Key Generation: Definition and Discussion

The main definition is presented in Subsection 2.3.2 and augmented in Subsection 2.3.4. In Subsection 2.3.3 we show that the main definition implies all natural security concerns discussed in the literature (with one notable exception that is addressed by the augmented definition). Finally, in Subsection 2.3.5 we relate our definitions to the framework of general secure multi-party computation.

2.3.1 Motivation for the definition

Our definition for password-based authenticated session-key generation is based on the "simulation paradigm" (cf. [31, 32, 2, 41, 15]). This paradigm has been used before in the context of session-key generation in the high-entropy case (e.g., [3, 49]), and also in the context of password-based authentication [14] (the difference between our definition and that of [14] is described in Section 2.3.3). According to this paradigm, we require a secure protocol that is run in the *real* model to emulate an *ideal* execution of a session-key generation functionality (where emulation usually means that the output distributions in both cases are computationally indistinguishable). In such an ideal execution, communication is via a trusted party who receives the parties inputs and (honestly) returns to each party its output, as designated by the functionality. Thus, defining the ideal model is essentially the same as defining the desired functionality of the problem at hand. We now described this functionality.

The problem of password-based authenticated session-key generation can be cast as a three-party functionality involving honest parties A and B, and an adversary C.[9] Parties A and B should input their joint password and receive identical, uniformly distributed session-keys. On the other

[9]We stress that unlike in most works regarding secure multi-party computation, the scenario includes three parties, but a protocol is constructed for only two of them. Furthermore, the identity of the adversary (among the three) is fixed beforehand. What makes the problem non-trivial is the fact that the honest parties communicate only via a communication line controlled by the adversary.

hand, the adversary C should have no output (and specifically should not obtain information on the password or output session-key). Furthermore, C should have no power to maliciously influence the outcome of the protocol (and thus, for example, cannot affect the choice of the key or cause the parties to receive different keys). However, recall that in a real execution, C controls the communication line between the (honest) parties. Thus, it can block all communication between A and B, and cause any protocol to fail. This (unavoidable) adversarial capability is modeled in the (modified) functionality by letting C input a single bit b indicating whether or not the execution is to be successful. Specifically, if $b = 1$ (i.e., success) then both A and B receive the above-described session-key. On the other hand, if $b = 0$ then A receives a session-key, whereas B receives a special abort symbol $\perp$ instead.[10] We stress that C is given no ability to influence the outcome beyond determining this single bit (i.e., b). In conclusion, the problem of password-based session-key generation is cast as the following three-party functionality:

$$(w_A, w_B, b) \mapsto \begin{cases} (U_n, U_n, \lambda) & \text{if } b = 1 \text{ and } w_A = w_B, \\ (U_n, \perp, \lambda) & \text{otherwise.} \end{cases}$$

where w_A and w_B are A and B's respective passwords. This functionality forms the basis for our definition of security.

An important observation in the context of password-based security is that, in a real execution, an adversary can always attempt impersonation by simply guessing the secret password and participating in the protocol, claiming to be one of the parties. If the adversary's guess is correct, then impersonation always succeeds (and, for example, the adversary knows the generated session-key). Furthermore, by executing the protocol with one of the parties, the adversary can verify whether or not its guess is correct, and thus can learn information about the password (e.g., it can rule out an incorrect guess from the list of possible passwords). Since the dictionary may be small, this information learned by the adversary in a protocol execution may not be negligible at all. Thus, we cannot hope to obtain a protocol that emulates an ideal-model execution (in which C learns *nothing*) up to computational indistinguishability. Rather, the inherent limitation of password-based security is accounted for by (only) requiring that a real execution can be simulated in the ideal model such that the output distributions (in the ideal and real models) are $(1 - O(\epsilon))$-*indistinguishable* (rather than 1-indistinguishable), where (as defined above) $\epsilon = 1/|\mathcal{D}|$.[11]

We note that the above limitation applies only to active adversaries who control the communication channel. Therefore, in the case of a passive (eavesdropping) adversary, we demand that the ideal and real model distributions be computationally indistinguishable (and not just $(1 - O(\epsilon))$-indistinguishable).

[10] This lack of symmetry in the definition is inherent as it is not possible to guarantee that A and B both terminate with the same "success/failure bit". For sake of simplicity, we (arbitrarily) choose to have A always receive a uniformly distributed session key and to have B always output $\perp$ when $b = 0$.

[11] Another way of dealing with this limitation of password-based security is to allow the ideal-model adversary a constant number of password guesses to the trusted party (such that if the adversary correctly guesses the password then it obtains full control over the honest parties' outputs; otherwise it learns nothing other than the fact that its guess was wrong). (We stress that this ideal-model adversary is stronger than the one considered in our formulation, which restricts the ideal-model adversary to obliviously decide whether to enable the execution or abort it.) Security is guaranteed by requiring that a real protocol execution can be simulated in this ideal model so that the output in the ideal model is *computationally indistinguishable* from that in a real execution. This is the approach taken by [14]; however we do not know how whether or not our protocol satisfied such a definition. See Section 2.3.3 for more discussion.

2.3.2 The actual definition

Following the simulation paradigm, we now define the ideal and real models (mentioned above), and present the actual definition of security.

The ideal model: Let $\hat{A}$ and $\hat{B}$ be honest parties and let $\hat{C}$ be any ppt ideal-model adversary (with arbitrary auxiliary input σ). An ideal-model execution proceeds in the following phases:

Initialization: A password $w \in_R \mathcal{D}$ is uniformly chosen from the dictionary and given to both $\hat{A}$ and $\hat{B}$.

Sending inputs to trusted party: $\hat{A}$ and $\hat{B}$ both send the trusted party the password they have received in the initialization stage. The adversary $\hat{C}$ sends either 1 (denoting a successful protocol execution) or 0 (denoting a failed protocol execution).

The trusted party answers all parties: In the case $\hat{C}$ sends 1, the trusted party chooses a uniformly distributed string $k \in_R \{0,1\}^n$ and sends k to both $\hat{A}$ and $\hat{B}$. In the case $\hat{C}$ sends 0, the trusted party sends $k \in_R \{0,1\}^n$ to $\hat{A}$ and $\perp$ to $\hat{B}$. In both cases, $\hat{C}$ receives no output.[12]

The ideal distribution is defined as follows:

$$\text{IDEAL}_{\hat{C}}(\mathcal{D}, \sigma) \stackrel{\text{def}}{=} (w, \text{output}(\hat{A}), \text{output}(\hat{B}), \text{output}(\hat{C}(\sigma)))$$

where $w \in_R \mathcal{D}$ is the input given to $\hat{A}$ and $\hat{B}$ in the initialization phase. Thus,

$$\text{IDEAL}_{\hat{C}}(\mathcal{D}, \sigma) = \begin{cases} (w, U_n, U_n, \text{output}(\hat{C}(\sigma))) & \text{if } \mathsf{send}(\hat{C}(\sigma)) = 1, \\ (w, U_n, \perp, \text{output}(\hat{C}(\sigma))) & \text{otherwise.} \end{cases}$$

where $\mathsf{send}(\hat{C}(\sigma))$ denotes the value sent by $\hat{C}$ (to the trusted party), on auxiliary input σ.

The real model: Let A and B be honest parties and let C be any ppt real-model adversary with arbitrary auxiliary input σ. As in the ideal model, the real model begins with an initialization stage in which both A and B receive an identical, uniformly distributed password $w \in_R \mathcal{D}$. Then, the protocol is executed with A and B communicating via C.[13] The execution of this protocol is denoted $C^{A(w),B(w)}(\sigma)$, where C's view is augmented with the accept/reject decision bits of A and B (this decision bit denotes whether a party's private output is a session-key or $\perp$). This augmentation is necessary, since in practice the decisions of both parties can be implicitly understood from their subsequent actions (e.g., whether or not the parties continue the communication after the session-key generation protocol has terminated). We note that in our specific formulation, A always accepts

[12] Since $\hat{A}$ and $\hat{B}$ are always honest, we need not deal with the case that they hand the trusted party different passwords. In fact, we can modify the definition so that there is no initialization stage or password received by the honest parties. The "send inputs" stage then involves $\hat{C}$ only, who sends a single success/fail bit to the trusted party. This definition is equivalent because the session key chosen by the trusted party is independent of the password and the honest parties always send the same password anyway.

[13] We stress that there is a fundamental difference between the real model as defined here and as defined in standard multi-party computation. Here, the parties A and B do *not have the capability of communicating directly with each other*. Rather, A can only communicate with C and likewise for B. This is in contrast to standard multi-party computation where all parties have direct communication links or where a broadcast channel is used.

and thus it is only necessary to provide C with the decision-bit output by B. With some abuse of notation,[14] the real distribution is defined as follows:

$$\text{REAL}_C(\mathcal{D},\sigma) \stackrel{\text{def}}{=} (w, \text{output}(A), \text{output}(B), \text{output}(C^{A(w),B(w)}(\sigma)))$$

where $w \in_R \mathcal{D}$ is the input given to A and B in the initialization phase, and $\text{output}(C^{A(w),B(w)}(\sigma))$ includes an indication of whether or not $\text{output}(B) = \perp$.

The definition of security: Loosely speaking, the definition requires that a secure protocol (in the real model) emulates the ideal model (in which a trusted party participates). This is formulated by saying that adversaries in the ideal model are able to simulate the execution of a real protocol, so that the input/output distribution of the simulation is $(1 - O(\epsilon))$-indistinguishable from in a real execution. We further require that passive adversaries can be simulated in the ideal-model so that the output distributions are computationally indistinguishable (and not just $(1 - O(\epsilon))$-indistinguishable).[15]

Definition 2.4 (password-based authenticated session-key generation): *A protocol for* password-based authenticated session-key generation *is secure if the following two requirements hold:*

1. Passive adversaries: *For every ppt real-model* passive *adversary C there exists a ppt ideal-model adversary $\hat{C}$ that always sends 1 to the trusted party such that*

$$\left\{\text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)\right\}_{n,\mathcal{D},\sigma} \stackrel{\text{c}}{\equiv} \left\{\text{REAL}_C(\mathcal{D},\sigma)\right\}_{n,\mathcal{D},\sigma}$$

   where $\mathcal{D} \subseteq \{0,1\}^n$ is any ppt samplable dictionary and $\sigma \in \{0,1\}^{\text{poly}(n)}$ is the auxiliary input for the adversary.

2. Arbitrary (active) adversaries: *For every ppt real-model adversary C there exists a ppt ideal-model adversary $\hat{C}$ such that*

$$\left\{\text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)\right\}_{n,\mathcal{D},\sigma} \stackrel{O(\epsilon)}{\equiv} \left\{\text{REAL}_C(\mathcal{D},\sigma)\right\}_{n,\mathcal{D},\sigma}$$

   where $\mathcal{D} \subseteq \{0,1\}^n$ is any ppt samplable dictionary, $\sigma \in \{0,1\}^{\text{poly}(n)}$ is the auxiliary input for the adversary, and $\epsilon \stackrel{\text{def}}{=} \frac{1}{|\mathcal{D}|}$. We stress that the constant in $O(\epsilon)$ is a universal one.

We note that the ideal-model as defined here reflects exactly what one would expect from a session-key generation protocol for which the honest parties hold joint *high-entropy* cryptographic keys (as in [6]). The fact that in the real execution the honest parties only hold *low-entropy* passwords is reflected in the relaxed notion of simulation that requires only $(1 - O(\epsilon))$-indistinguishability (rather than computational indistinguishability) between the real and ideal models.

[14] Here and in the sequel, $\text{output}(A)$ (resp., $\text{output}(B)$) denotes the output of A (resp., B) in the execution $C^{A(w),B(w)}(\sigma)$, whereas $\text{output}(C^{A(w),B(w)}(\sigma))$ denotes C's output in this execution.

[15] A passive adversary is one that does not modify, omit or insert any messages sent between A or B. That is, it can only eavesdrop and thus is limited to analyzing the transcript of a protocol execution between two honest parties. Passive adversaries are also referred to as *semi-honest* in the literature (e.g., in [30]).

2.3.3 Properties of Definition 2.4

Definition 2.4 asserts that the joint input–output distribution from a real execution is at most "$O(\epsilon)$-far" from an ideal execution in which the adversary learns nothing (and has no influence on the output except from the possibility of causing B to reject). This immediately implies that the output session-key is $(1 - O(\epsilon))$-pseudorandom (which, as we have mentioned, is the best possible for password-based key generation). Thus, if such a session-key K is used for encryption then for any (partial information) predicate P and any distribution on the plaintext m, the probability that an adversary learns $P(m)$ given the ciphertext $E_K(m)$ is at most $O(\epsilon) + \mu(n)$ greater than the a-priori probability (when the adversary is not given the ciphertext). Likewise, if the key K is used for a message authentication code (MAC), then the probability that an adversary can generate a correct MAC-tag on a message not sent by A or B is at most negligibly greater than $O(\epsilon)$. We stress that the security of the output session-key does not deteriorate with its usage; that is, it can be used for polynomially-many encryptions or MACs and the advantage of the adversary remains $O(\epsilon) + \mu(n)$. Another important property of Definition 2.4 is that, except with probability $O(\epsilon)$, (either one party detects failure or) both parties terminate with the *same* session-key.

Definition 2.4 also implies that the password used remains $(1 - O(\epsilon))$-indistinguishable from a randomly chosen (new) password $\tilde{w} \in_R \mathcal{D}$: This can be seen from the fact that in the ideal model, the adversary learns nothing of the password w, which is part of the IDEAL distribution. This implies, in particular, that a secure protocol is resistant to off-line dictionary attacks (whereby an adversary scans the dictionary in search of a password that is "consistent" with its view of a protocol execution).

Other desirable properties of session-key protocols are also guaranteed by Definition 2.4. Specifically, we mention *forward secrecy* and security in the face of *loss of session keys* (also known as *known-key attacks*). Forward secrecy states that the session-key remains secure even if the password is revealed after the protocol execution [20]. Analogously, security in the face of loss of session-keys means that the password and the current session-key maintain their security even if prior session-keys are revealed [6]. These properties are immediately implied by the fact that, in the ideal-model, there is no dependence between the session-key and the password and between session-keys from different sessions. Thus, learning the password does not compromise the security of the session-key and vice versa.[16]

An additional property that is desirable is that of *intrusion detection*. That is, if the adversary modifies any message sent in a session, then with probability at least $(1 - O(\epsilon))$ this is detected and at least one party rejects. This property is not guaranteed by Definition 2.4 itself. However, it does hold for our protocol (as shown in Proposition 4.13, see Section 4.6). Combining this with Part 1 of Definition 2.4 (i.e., the requirement regarding passive adversaries), we conclude that in order for C to take advantage of its ability to learn "$O(\epsilon)$-information", C must take the chance of being detected with probability $1 - O(\epsilon)$.

Finally, we observe that the above definition also enables mutual authentication. This is because A's output session-key is always $(1 - O(\epsilon))$-pseudorandom to the adversary. As this key is secret, it can be used for explicit authentication via a (mutual) challenge–response protocol.[17] By adding such a step to any secure session-key protocol, we obtain explicit mutual authentication.

[16] The independence of session-keys from different sessions relates to the multi-session case, which is discussed in Section 2.5. For now, it is enough to note that the protocol behaves as expected in that after t executions of the real protocol, the password along with the outputs from all t sessions are $(1 - O(t\epsilon))$-indistinguishable from t ideal executions. The fact that security is maintained in the face of session-key loss is explicitly shown in Section 2.5.

[17] It is easy to show that such a key can be used directly to obtain a $(1 - O(\epsilon))$-pseudorandom function, which can then be used in a standard challenge–response protocol.

Comparison to other definitions. The focus of this work is not with respect to finding the "right" definition for password-based session-key generation. Rather, the main question that we consider is the feasibility of *solving* this problem under some reasonable definition. We believe that at the very least our definition is reasonable, and in particular it implies the natural security concerns discussed in prior work. Furthermore, our definition is in agreement with the traditions of the general area (cf. [31, 32, 2, 41, 15]) as well as of the study of this specific problem (cf. [6, 3, 49, 4]). However, as mentioned in Footnote 11, there is one specific alternative formulation, aimed at addressing the same security concerns (cf. [14]), which we wish to further discuss below.

Recall that the inherent limitation of password-based security (which in turn arises from the straightforward password-guessing attack) is dealt with in our formulation by requiring that the real and ideal executions be only $(1-O(\epsilon))$-indistinguishable (rather than 1-indistinguishable). An alternative way of dealing with this limitation (of password-based security) is to explicitly allow the ideal-model adversary a constant number of password guesses to the trusted party (such that if the adversary correctly guesses the password then it obtains full control over the honest parties' outputs; otherwise it learns nothing other than the fact that its guess was wrong). Security is then guaranteed by requiring that the real and ideal executions are *computationally indistinguishable*.[18] This definition is somewhat more elegant than ours, and is the one considered by [14].

The latter formulation implies ours, but it is not clear whether the converse holds. Still, it seems that the actual consequences (i.e., in the sense discussed above) of both definitions are the same. That is, in both cases the difference between using the protocol-generated session-key and a fully random key is at most $O(\epsilon)$. For example, consider the case that the parties use the session-key for authenticating messages with a MAC. Both under our definition and under the definition of [14], no polynomial-time adversary will be able to forge any MAC-tag (i.e., in a way that fools the parties), except than with probability $O(\epsilon)$, during the entire session in which the key is used. On the other hand, under both definitions, an adversary can always succeed in forging a MAC with probability ϵ (e.g., by just carrying out a straightforward password-guessing attack).

2.3.4 Augmenting the definition

Although Definition 2.4 seems to capture all that is desired from authenticated session-key generation, there is a subtlety that it fails to address (as pointed out by Rackoff in a personal communication to the authors of [6]). The issue is that the two parties do not necessarily terminate the session-key generation protocol simultaneously, and so one party may terminate the protocol and start using the session-key while the other party is still executing instructions of the session-key generation protocol (i.e., determining its last message). This situation is problematic because the use of a session-key inevitably leaks information. Thus, the adversary may be able to use this information in order to attack the protocol execution that is still in progress from the point of view of the other party.

This issue is highlighted by the following attack devised by Rackoff. Consider any protocol that is secure by Definition 2.4 and assume that in this protocol A concludes first. Now, modify B so that if the last message received by B equals $f_k(0)$, where k is the output session key and $\{f_s\}_s$ is a pseudorandom function ensemble, then B publicly outputs the password w. The modified protocol is still secure by Definition 2.4, because in the original protocol, the value $f_k(0)$ is pseudorandom

[18] We stress that the ideal-model in the alternative formulation is stronger than the ideal-model considered by our formulation (which makes the alternative formulation of security potentially weaker), but the level of indistinguishability required by the alternative formulation is stronger (which makes the alternative formulation of security potentially stronger). However, the latter aspect dominates because the ideal-model of the alternative formulation can be emulated in a $(1-O(\epsilon))$-indistinguishable manner by the ideal-model of our formulation.

with respect to the adversary's view (otherwise this would amount to the adversary being able to distinguish the session-key from a random key). However, consider a scenario in which upon completing the session-key generation protocol, A sends a message that contains the value $f_k(0)$ (such use of the session-key is not only legitimate, but also quite reasonable). In this case, the adversary can easily obtain the password by passing $f_k(0)$ (as sent by A) to B, who has not yet completed the session-key protocol. In summary, Definition 2.4 should be modified in order to ensure that any use of the session-key after one of the parties has completed the session-key protocol cannot help the adversary in its attack on this protocol.

In order to address this issue, Definition 2.4 is augmented so that the adversary receives a session-key challenge after the first party concludes its execution of the session-key protocol. The session-key challenge is chosen so that with probability 1/2 it equals the actual session-key (as output by the party that has finished) and with probability 1/2 it is a uniformly distributed string. The augmentation requires that the adversary be unable to distinguish between these challenge cases. Intuitively, this solves the above-described problem because the adversary can use the session-key challenge it receives in order to simulate any messages that may be sent by A following the session-key protocol execution.

The augmented ideal model. Let $\hat{A}$, $\hat{B}$, $\hat{C}$ and σ be as in the above definition of the ideal model. Then, the augmented ideal model proceeds in the following phases:

Initialization: $\hat{A}$ and $\hat{B}$ receive $w \in_R \mathcal{D}$.

Honest parties send inputs to the trusted party: $\hat{A}$ and $\hat{B}$ both send w.

Trusted party answers $\hat{A}$: The trusted party chooses $k \in_R \{0,1\}^n$ and sends it to $\hat{A}$.

Trusted party chooses session-key challenge for $\hat{C}$: The trusted party chooses $\beta \in_R \{0,1\}$ and gives $\hat{C}$ the string k_β, where $k_1 \stackrel{\text{def}}{=} k$ and $k_0 \in_R \{0,1\}^n$.

Adversary $\hat{C}$ sends input to the trusted party: $\hat{C}$ sends either 1 (denoting a successful protocol execution) or 0 (denoting a failed protocol execution).

Trusted party answers $\hat{B}$: If $\hat{C}$ sent 1 in the previous phase, then the trusted party gives the key k to $\hat{B}$. Otherwise, it gives $\hat{B}$ an abort symbol $\perp$.

The augmented ideal distribution is defined by:

$$\textsc{ideal-aug}_{\hat{C}}(\mathcal{D},\sigma) \stackrel{\text{def}}{=} (w, \text{output}(\hat{A}), \text{output}(\hat{B}), \text{output}(\hat{C}(\sigma, k_\beta)), \beta)$$

where $w \in_R \mathcal{D}$. (Notice the inclusion of β in the $\textsc{ideal-aug}$ distribution.) We remark that in an ideal execution, $\hat{A}$ always concludes first and always accepts.

The augmented real model. The real model execution is the same as above except for the following modification. Recall that the scheduling of a protocol execution is controlled by C. Therefore, C controls which party (A or B) concludes first. If the first party concluding outputs an abort symbol $\perp$, then the adversary is simply given $\perp$. (Since the accept/reject bit is anyway public, this is meaningless.) On the other hand, if the first party to terminate the execution locally outputs a session-key, then a bit $\beta \in_R \{0,1\}$ is chosen, and C is given a corresponding challenge: If $\beta = 0$, then C is given a uniformly distributed string $r \in_R \{0,1\}^n$, else (i.e., $\beta = 1$) C is given the session-key as output by the terminating party. The augmented real distribution is defined as follows:

$$\text{REAL-AUG}_C(\mathcal{D},\sigma) \stackrel{\rm def}{=} (w, \text{output}(A), \text{output}(B), \text{output}(C^{A(w),B(w)}(\sigma)), \beta)$$

where $C^{A(w),B(w)}(\sigma)$ denotes the above described (augmented) execution.

Finally, the definition of security is analogous to Definition 2.4:

Definition 2.5 (augmented password-based authenticated session-key generation): *We say that a protocol for* augmented password-based authenticated session-key generation *is secure if the following two requirements hold:*

1. Passive adversaries: *For every ppt real-model* passive *adversary C there exists a ppt ideal-model adversary $\hat{C}$ that always sends 1 to the trusted party such that*

$$\left\{\text{IDEAL-AUG}_{\hat{C}}(\mathcal{D},\sigma)\right\}_{n,\mathcal{D},\sigma} \stackrel{\rm c}{\equiv} \left\{\text{REAL-AUG}_C(\mathcal{D},\sigma)\right\}_{n,\mathcal{D},\sigma}$$

   where $\mathcal{D} \subseteq \{0,1\}^n$ is any ppt samplable dictionary and $\sigma \in \{0,1\}^{\mathrm{poly}(n)}$ is the auxiliary input for the adversary.

2. Arbitrary adversaries: *For every ppt real-model adversary C there exists a ppt ideal-model adversary $\hat{C}$ such that*

$$\left\{\text{IDEAL-AUG}_{\hat{C}}(n,\mathcal{D},\sigma)\right\}_{n,\mathcal{D},\sigma} \stackrel{O(\epsilon)}{\equiv} \left\{\text{REAL-AUG}_C(n,\mathcal{D},\sigma)\right\}_{n,\mathcal{D},\sigma}$$

   where $\mathcal{D} \subseteq \{0,1\}^n$ is any ppt samplable dictionary, $\sigma \in \{0,1\}^{\mathrm{poly}(n)}$ is the auxiliary input for the adversary, and $\epsilon \stackrel{\rm def}{=} \frac{1}{|\mathcal{D}|}$.

We first explain how this augmentation addresses the problem discussed above (i.e., prevents the attack of Rackoff). In the augmented ideal model, $\hat{C}$ learns *nothing* about the value of β. Therefore, by Definition 2.5, it follows that in the augmented real model, C can distinguish the case that $\beta = 0$ from the case that $\beta = 1$ with probability at most $O(\epsilon)$. Now, consider the case that the session-key challenge given to C is a uniformly distributed string (i.e., $\beta = 0$). Then, since C can generate the challenge itself, it clearly cannot help C in any way in its attack on the protocol. On the other hand, we are interested in analyzing the probability that the session-key itself can help C in its attack on the protocol. The point is that if C could utilize knowledge of this key, then this additional knowledge could be used to distinguish the case that $\beta = 0$ from the case that $\beta = 1$. We conclude that the information that C can obtain about the session-key in a real setting does not help it in attacking the session-key generation protocol (except with probability $O(\epsilon)$).

As we have seen the above augmentation resolves the problem outlined by Rackoff. However, in contrast to Definition 2.4, it is not clear that Definition 2.5 implies all the desired properties of secure session-key generation protocols.[19] We therefore show that all the properties of Definition 2.4 are indeed preserved in Definition 2.5. In fact:

Proposition 2.6 *Any protocol that is secure by Definition 2.5 is secure by Definition 2.4.*

[19] Clearly, if C were always given the session key, then the definition guarantees no security with respect to the session-key. So, we must show that in Definition 2.5, where C is given the key with probability 1/2, security (as per Definition 2.4) is maintained.

Proof: Intuitively, the proposition holds because in the case that $\beta = 0$, the adversary in the augmented model has no additional advantage over the adversary for the basic model, where we refer to the model of Def. 2.4 as the basic or unaugmented model. (Recall that when $\beta = 0$, the adversary merely receives a uniformly distributed string.) Therefore, any success by an adversary for the basic model can be translated into an adversarial success in the augmented model, provided that $\beta = 0$ (in the augmented model). Since $\beta = 0$ with probability $1/2$, a protocol proven secure for the augmented model must also be secure in the basic model. Details follow.

Assume that there exists a protocol that is secure by Definition 2.5 (the rest of this proof refers implicitly to this protocol). First, notice that for any real-model adversary C (as in Definition 2.4), there exists an *augmented* real-model adversary C' such that

$$\{\text{REAL}_C(\mathcal{D}, \sigma)\} \equiv \{\text{REAL-AUG}_{C'}(\mathcal{D}, \sigma) \mid \beta = 0\} \tag{1}$$

In order to see this, consider an adversary C' who simply invokes the basic-model adversary C in the augmented model and ignores the additional session-key challenge provided, which in the case that $\beta = 0$ provides no information anyway. (In fact, it holds that $\{\text{REAL}_C(\mathcal{D}, \sigma)\} \equiv \{\text{REAL-AUG}_{C'}(\mathcal{D}, \sigma)\}$, but for this proof we only need to consider the conditional space where $\beta = 0$).

Next, by Definition 2.5, we have that for any augmented real-model adversary C', there exists an augmented ideal-model adversary $\hat{C}'$ such that

$$\{\text{REAL-AUG}_{C'}(\mathcal{D}, \sigma)\}_{n,\mathcal{D},\sigma} \stackrel{\kappa\cdot\epsilon}{\equiv} \{\text{IDEAL-AUG}_{\hat{C}'}(\mathcal{D}, \sigma)\}_{n,\mathcal{D},\sigma} \tag{2}$$

where κ is a constant. This implies that

$$\{\text{REAL-AUG}_{C'}(\mathcal{D}, \sigma) \mid \beta = 0\} \stackrel{2\kappa\cdot\epsilon}{\equiv} \{\text{IDEAL-AUG}_{\hat{C}'}(\mathcal{D}, \sigma) \mid \beta = 0\} \tag{3}$$

Eq. (3) holds because $\beta = 0$ with probability $1/2$, and thus any distinguishing gap greater than $2\kappa \cdot \epsilon$ can be translated into a distinguishing gap of greater than $\kappa \cdot \epsilon$ for the distributions in Eq. (2).

Finally, we claim that for any augmented ideal-model adversary $\hat{C}'$, their exists an ideal-model adversary $\hat{C}''$ (as in Definition 2.4) such that

$$\{\text{IDEAL-AUG}_{\hat{C}'}(\mathcal{D}, \sigma) \mid \beta = 0\} \equiv \{\text{IDEAL}_{\hat{C}''}(\mathcal{D}, \sigma)\} \tag{4}$$

Eq. (4) holds because when $\beta = 0$, adversary $\hat{C}'$ receives a uniformly distributed string in the ideal execution. Thus, $\hat{C}''$ can invoke $\hat{C}'$ (while in the basic, unaugmented model) and pass it a uniformly distributed string for its session-key challenge.

Combining Equations (1), (3) and (4) we obtain the proposition. ■

2.3.5 Session-key generation as secure multiparty computation

We have cast the problem of password-based session-key generation in the framework of secure multiparty computation. However, there are a number of essential differences between our model here and the standard model of multiparty computation.

- *Real-model communication:* In the standard model, all parties can communicate directly with each other. However, in our context, the honest parties A and B may only communicate with the adversary C. This difference models the fact that A and B communicate over an "open" communication channel that is vulnerable to active man-in-the-middle attacks.

- *Adversarial parties:* In the standard model, any party may be corrupted by the adversary. However, here we assume that A and B are always honest and that only C can be adversarial.
- *Quantification over the inputs:* In the standard model, security is guaranteed for *all* inputs. In particular, this means that an adversary cannot succeed in affecting the output distribution even if it knows the honest parties' inputs. This is in contrast to our setting where the honest parties' joint password must be kept secret from the adversary. Thus, we quantify over all ppt samplable dictionaries and all auxiliary inputs to the adversary, rather than over specific inputs (to the honest parties). Another way of viewing the difference is that, considering the inputs of the honest parties, we quantify over efficiently samplable input *distributions* (of certain min-entropy), whereas in the standard model the quantification is over input *values*.
- *The "level" of indistinguishability:* Finally, in the standard model, the real and ideal output distributions are required to be computationally indistinguishable (and thus "essentially" the same). On the other hand, due to the inherent limitation resulting from the use of low-entropy passwords, we only require that these distributions be $(1 - O(\epsilon))$-indistinguishable.

2.4 Our Main Result

Given Definition 2.5, we can now formally state our main result.

Theorem 2.7 *Assuming the existence of trapdoor permutations, there exist secure protocols for* (augmented) *password-based authenticated session-key generation.*

Other distributions over $\mathcal{D}$: For simplicity, we have assumed above that the parties share a uniformly chosen password $w \in_R \mathcal{D}$. However, our proofs extend to any ppt samplable distribution (over any dictionary) so that no element occurs (in this distribution) with probability greater than ϵ.

2.5 Multi-Session Security

The definition above relates to two parties executing a session-key generation protocol once. Clearly, we are interested in the more general case where many different parties run the protocol any number of times. It turns out that any protocol that is secure for a single invocation between two parties (i.e., as in Definitions 2.4 and 2.5), is secure in the multi-party and sequential invocation case.

2.5.1 Sequential executions for one pair of parties

Let A and B be parties who invoke t sequential executions of a session-key generation protocol. Given that an adversary may gain a password guess per each invocation, the "security loss" for t invocations should be $O(t\epsilon)$. That is, we consider ideal and real distributions consisting of the outputs from all t executions. Then, we require that these distributions be $(1 - O(t\epsilon))$-indistinguishable. Below, we prove that *any* secure protocol for password-based authenticated session-key generation maintains $O(t\epsilon)$ security after t sequential invocations.

Sequential vs concurrent executions for two parties: Our solution is proven secure only if A and B do not invoke *concurrent* executions of the session-key generation protocol with the same password. Here and below, we treat a pair of parties that share several independently distributed passwords as different pairs of parties. We stress that security is not guaranteed in a scenario where the adversary invokes B twice or more (using the same password) during a single execution with A

(or vice versa). Therefore, in order to actually use our protocol, some mechanism must be used to ensure that such concurrent executions do not take place. This can be achieved by having A and B wait Δ units of time between protocol executions, where Δ is greater than the time taken to run a single execution. We note that when parties do not come "under attack", this delay of Δ will usually not affect them (since they will usually not execute two successful session-key generation protocols immediately one after the other).

We remark that this limitation on concurrent executions does *not* prevent the parties from opening a number of different (independently-keyed) communication lines. They may do this by running the session-key protocol *sequentially*, once for each desired communication line. However, in this case, they incur a delay of Δ units of time between each execution. Alternatively, they may run the protocol once and obtain a $(1-O(\epsilon))$-pseudorandom session-key. By applying a pseudorandom generator to this key, any polynomial number of computationally independent $(1-O(\epsilon))$-pseudorandom session-keys can be derived.

Proof of security for sequential executions: We prove the sequential composition of secure password-based session-key protocols for the basic definition (Definition 2.4). The proof for the augmented definition (Definition 2.5) is almost identical. We begin with the following notation. Let $R_C(w,\sigma) \stackrel{\rm def}{=} (\text{output}(A), \text{output}(B), \text{output}(C^{A(w),B(w)}(\sigma)))$. That is, $R_C(w,\sigma)$ equals the *outputs* of A, B and C from a real execution where the joint password equals w and C's auxiliary input is σ (and thus $\text{REAL}_C(\mathcal{D},\sigma) = (w, R_C(w,\sigma))$ for $w \in_R \mathcal{D}$). Next, we present the equivalent notation $I_{\hat{C}}(\sigma)$ for the ideal-model as follows:

$$I_{\hat{C}}(\sigma) = \begin{cases} (U_n, U_n, \text{output}(\hat{C}(\sigma))) & \text{if } \mathsf{send}(\hat{C}(\sigma)) = 1, \\ (U_n, \perp, \text{output}(\hat{C}(\sigma))) & \text{otherwise.} \end{cases}$$

Thus, $\text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma) = (w, I_{\hat{C}}(\sigma))$ for $w \in_R \mathcal{D}$. (Recall that $\mathsf{send}(\hat{C}(\sigma))$ denotes the input-bit sent by $\hat{C}$ to the trusted party upon auxiliary input σ, and $\text{output}(\hat{C}(\sigma))$ denotes its final output.) We stress that $I_{\hat{C}}(\sigma)$ is *independent* of the specific dictionary $\mathcal{D}$ and the password w (and for this reason $\mathcal{D}$ does not appear in the notation). This is equivalent to the definition of $\text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)$ in Section 2.3.2 because the password plays no role in the choice of the session-key or in $\hat{C}$'s decision to send 0 or 1 to the trusted party. Furthermore, as mentioned in Footnote 12, since $\hat{A}$ and $\hat{B}$ are always honest, there is no need to have them receive any password for input or send any message whatsoever to the trusted party.

We now define the distribution $\text{REAL}^t_C(\mathcal{D},\sigma_0)$, representing t sequential executions, as follows:

$$\text{REAL}^t_C(\mathcal{D},\sigma_0) \stackrel{\rm def}{=} (w, \sigma_1 = R_C(w,\sigma_0), \sigma_2 = R_C(w,\sigma_1), \cdots, \sigma_t = R_C(w,\sigma_{t-1}))$$

where σ_0 is some arbitrary auxiliary input to C and $w \in_R \mathcal{D}$. Likewise, the distribution $\text{IDEAL}^t_{\hat{C}}(\mathcal{D},\sigma_0)$ is defined by:

$$\text{IDEAL}^t_{\hat{C}}(\mathcal{D},\sigma_0) \stackrel{\rm def}{=} (w, \sigma_1 = I_{\hat{C}}(\sigma_0), \sigma_2 = I_{\hat{C}}(\sigma_1), \cdots, \sigma_t = I_{\hat{C}}(\sigma_{t-1}))$$

where σ_0 is some arbitrary auxiliary input to C and $w \in_R \mathcal{D}$. (For the sake of brevity, we will sometimes omit the explicit assignments $\sigma_i = I_{\hat{C}}(\sigma_{i-1})$ and will just write $(w, I_{\hat{C}}(\sigma_0), I_{\hat{C}}(\sigma_1), \ldots, I_{\hat{C}}(\sigma_{t_1}))$.)

Notice that in the i'th session, C receives *all* the parties' outputs from the previous session (i.e., including previous session-keys), rather than just its own output (or state information) as one may expect. This models the fact that information about previous session-keys may be leaked from protocols who use them. Therefore, the security of future session keys holds, even if previous session-keys are revealed to the adversary.

By the above notation, $\text{REAL}^1_C(\mathcal{D},\sigma_0) = \text{REAL}_C(\mathcal{D},\sigma_0)$ and $\text{IDEAL}^1_{\hat{C}} = \text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma_0)$ and thus by the definition it holds that $\text{REAL}^1_C(\mathcal{D},\sigma_0)$ and $\text{IDEAL}^1_{\hat{C}}(\mathcal{D},\sigma_0)$ are $(1-O(\epsilon))$-indistinguishable. We now show that for *any* polynomial function of the security parameter $t=t(n)$, the distributions $\text{IDEAL}^t_{\hat{C}}(\mathcal{D},\sigma_0)$ and $\text{REAL}^t_C(\mathcal{D},\sigma_0)$ are $(1-O(t\epsilon))$-indistinguishable.

Proposition 2.8 *Consider a secure protocol for password-based authenticated session-key generation. Then, for every ppt real-model adversary C there exists a ppt ideal-model adversary $\hat{C}$ such that for every polynomial function $t=t(n)$*

$$\left\{\text{IDEAL}^t_{\hat{C}}(\mathcal{D},\sigma_0)\right\}_{n,\mathcal{D},\sigma_0} \stackrel{\mathrm{O(t\epsilon)}}{\equiv} \left\{\text{REAL}^t_C(\mathcal{D},\sigma_0)\right\}_{n,\mathcal{D},\sigma_0}$$

Proof: We prove the proposition using a hybrid argument. Specifically, define the following hybrid distribution between $\text{IDEAL}^t_{\hat{C}}$ and REAL^t_C:

$$H_i(\mathcal{D},\sigma_0) = (w, \sigma_1 = I_{\hat{C}}(\sigma_0), \ldots, \sigma_i = I_{\hat{C}}(\sigma_{i-1}), \sigma_{i+1} = R_C(w,\sigma_i), \ldots, \sigma_t = R_C(w,\sigma_{t-1}))$$

Thus, $H_0(\mathcal{D},\sigma_0) = \text{REAL}^t_C(\mathcal{D},\sigma_0)$ and $H_t(\mathcal{D},\sigma_0) = \text{IDEAL}^t_{\hat{C}}(\mathcal{D},\sigma_0)$. We begin by proving that for every i and for every ppt real-model adversary C, there exists a ppt ideal-model adversary $\hat{C}$ such that for every ppt samplable dictionary $\mathcal{D} \subseteq \{0,1\}^n$ and every auxiliary input $\sigma_0 \in \{0,1\}^{\mathrm{poly}(n)}$,

$$\{H_i(\mathcal{D},\sigma_0)\} \stackrel{\mathrm{O(\epsilon)}}{\equiv} \{H_{i+1}(\mathcal{D},\sigma_0)\}$$

Furthermore, the running time of $\hat{C}$ exceeds that of C by a fixed polynomial additive factor only. Intuitively, the fact that H_i and H_{i+1} can be distinguished with probability at most $O(\epsilon)$ holds because the only difference between the distributions is in the $(i+1)^{\text{th}}$ position which in H_i equals $R_C(w,\sigma_i)$ and in H_{i+1} equals $I_{\hat{C}}(\sigma_i)$. More formally, first notice that for the series $(\sigma_1 = I_{\hat{C}}(\sigma_0), \ldots, \sigma_i = I_{\hat{C}}(\sigma_{i-1}))$, the value σ_i is *independent* of the password w. This means that the adversary can generate this series and σ_i by itself. Therefore, by the security of the protocol for the basic definition (of one execution), it follows that:

$$\{(w, I_{\hat{C}}(\sigma_0), \ldots, I_{\hat{C}}(\sigma_{i-1}), R_C(w,\sigma_i))\} \stackrel{\mathrm{O(\epsilon)}}{\equiv} \{(w, I_{\hat{C}}(\sigma_0), \ldots, I_{\hat{C}}(\sigma_{i-1}), I_{\hat{C}}(\sigma_i))\} \qquad (5)$$

Next, notice that the two distributions in Eq. (5) both explicitly contain the password w. Thus, given one of these distributions, the series $(R_C(w,\sigma_{i+1}), \ldots, R_C(w,\sigma_{t-1}))$ can be generated in probabilistic polynomial-time, for any polynomial function $t=t(n)$. Therefore,

$$\begin{aligned}&\{(w, I_{\hat{C}}(\sigma_0), \ldots, I_{\hat{C}}(\sigma_{i-1}), R_C(w,\sigma_i), R_C(w,\sigma_{i+1}), \ldots, R_C(w,\sigma_{t-1}))\}\\ \stackrel{\mathrm{O(\epsilon)}}{\equiv}\ &\{(w, I_{\hat{C}}(\sigma_0), \ldots, I_{\hat{C}}(\sigma_{i-1}), I_{\hat{C}}(\sigma_i), R_C(w,\sigma_{i+1}), \ldots, R_C(w,\sigma_{t-1}))\}\end{aligned}$$

which is equivalent to saying that

$$\{H_i(\mathcal{D},\sigma_0)\} \stackrel{\mathrm{O(\epsilon)}}{\equiv} \{H_{i+1}(\mathcal{D},\sigma_0)\}$$

By summing over the difference between all pairs of hybrids H_i and H_{i+1}, we obtain that:

$$\{H_0(\mathcal{D},\sigma_0)\} \stackrel{\mathrm{O(t\epsilon)}}{\equiv} \{H_t(\mathcal{D},\sigma_0)\}$$

Furthermore, since for each step of the hybrid summation, the running time of the adversary is increased by an additive polynomial factor only, the resulting ideal adversary is still polynomial-time. This completes the proof of the proposition. ■

2.5.2 Concurrent executions for many pairs of parties

We now show the generalization to the case where many different parties execute the session-key protocol simultaneously. This includes the case that the adversarial channel controls any number of the legitimate parties.[20] Specifically, we claim that for m invocations of the protocol (which must be sequential for the same pair of parties using the same password and may be *concurrent* otherwise), the difference between the ideal and real executions is at most $O(m\epsilon)$.

We show this in the case of m different pairs, each pair executing a single invocation (the general case is similar). That is, there are m pairs of parties $(A_1, B_1), \ldots, (A_m, B_m)$, and each pair shares a secret password $w_i \in_R \mathcal{D}$. (We do not assume that the A's and B's are distinct, yet do assume that for each $i \neq j$, the passwords w_i and w_j are independently chosen.) Then, the formal definition of security in this scenario is obtained by the natural generalization of the basic definition. That is, an *ideal* execution is defined where the trusted party works with each pair independently and each execution is the same as in the basic single-execution case. We note that this means that the adversary can choose its input bit b differently in each of the m executions. The *real* execution is also defined in the natural way, where m different pairs of parties with independent passwords run concurrently in the network. We denote by $\text{MULTI-REAL}^m_{\hat{C}}(\mathcal{D},\sigma)$ the output of such a real execution with m pairs of parties. That is, let w_i be the shared password of (A_i, B_i). Then,

$$\text{MULTI-REAL}^m_{\hat{C}}(\mathcal{D},\sigma) = \Big((w_1, \text{output}(A_1), \text{output}(B_1)), \ldots, (w_m, \text{output}(A_m), \text{output}(B_m)), \text{output}(C(\sigma))\Big)$$

where output$(C(\sigma))$ denotes the output of C after all m concurrent executions. The distribution $\text{MULTI-IDEAL}^m_{C}(\mathcal{D},\sigma)$ is defined in an analogous way. In this section, we prove the following proposition:

Proposition 2.9 *Consider a protocol for password-based authenticated session-key generation that is secure according to Definition 2.4. Then, for every ppt real-model adversary C playing concurrent executions (as described above), there exists a ppt ideal-model adversary $\hat{C}$, such that for every polynomial function $m = m(n)$,*

$$\left\{\text{MULTI-IDEAL}^m_{\hat{C}}(\mathcal{D},\sigma)\right\}_{n,\mathcal{D},\sigma} \stackrel{O(m\epsilon)}{\equiv} \left\{\text{MULTI-REAL}^m_{C}(\mathcal{D},\sigma)\right\}_{n,\mathcal{D},\sigma}$$

In order to prove Proposition 2.9, we first show that any secure session-key generation protocol has a *canonical* ideal-model adversary (simulation). This canonical adversary will be such that it can be used to simulate in the multi-execution case.

We first define the following distribution, which is a mental experiment:

$$\text{MENTAL}_C(\mathcal{D},\sigma) \stackrel{\text{def}}{=} (w, \text{output}(A), \text{output}(B), \text{output}(C^{A(w'),B(w')}(\sigma)))$$

where $w, w' \in_R \mathcal{D}$ are passwords that are uniformly and independently chosen from the dictionary. That is, the difference between REAL and MENTAL is whether or not the parties A and B use the same password that appears as the first element in the distribution, or an independently chosen password. We stress that in the MENTAL distribution, output(A) and output(B) relate to the parties' outputs from the execution $C^{A(w'),B(w')}$, where they use the password w' (and not w). Intuitively, MENTAL and REAL can be distinguished with probability at most $O(\epsilon)$; otherwise, the protocol reveals more than "$O(\epsilon)$ information" about the password. However, we are actually interested in the probability of distinguishing MENTAL and IDEAL.

[20]The importance of this extension was pointed out by Boyarsky [13].

Claim 2.10 *Let C be a ppt real-model adversary and let $\hat{C}$ be the ppt ideal-model adversary such that*

$$\left\{\text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)\right\}_{n,\mathcal{D},\sigma} \stackrel{O(\epsilon)}{\equiv} \left\{\text{REAL}_{C}(\mathcal{D},\sigma)\right\}_{n,\mathcal{D},\sigma}$$

as guaranteed by Definition 2.4. Then,

$$\left\{\text{MENTAL}_{C}(\mathcal{D},\sigma)\right\}_{n,\mathcal{D},\sigma} \stackrel{O(\epsilon)}{\equiv} \left\{\text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)\right\}_{n,\mathcal{D},\sigma}$$

Proof: Intuitively, since MENTAL and REAL cannot be distinguished and IDEAL and REAL cannot be distinguished, the same holds for MENTAL and IDEAL. Formally, let DIST be a distribution which is either sampled from $\text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)$ or $\text{REAL}_{C}(\mathcal{D},\sigma)$. Then, consider the transformation obtained by replacing the *first* element in DIST with a random password $\tilde{w} \in_R \mathcal{D}$. Now, if DIST was sampled from $\text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)$, then the result is still a random sample from $\text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)$. This holds because the password w in the IDEAL distribution is chosen independently of everything else. However, if DIST was sampled from $\text{REAL}_{C}(\mathcal{D},\sigma)$, then the result is exactly a random sample from $\text{MENTAL}_{C}(\mathcal{D},\sigma)$. This is because the only difference between the REAL and MENTAL distribution is whether the parties' use the same password that appears as the first element, or an independent one.

We conclude that any machine distinguishing MENTAL and IDEAL with probability p can be used to distinguish between REAL and IDEAL with probability p. Since, by the security of the protocol, REAL and IDEAL can be distinguished with probability at most $O(\epsilon)$, it holds that MENTAL and IDEAL can also be distinguished with probability at most $O(\epsilon)$. ■

The above claim yields the following canonical ideal-model adversary:

Canonical ideal-model adversary $\hat{C}_{\text{CAN}}$. Let C be a real-model adversary. Then, upon auxiliary input σ, the ideal-model adversary $\hat{C}_{\text{CAN}}$ works by choosing a random password $w' \in_R \mathcal{D}$, and perfectly emulating $C^{A(w'),B(w')}(\sigma)$. That is, $\hat{C}_{\text{CAN}}$ invokes the real-model adversary C with input σ, and plays the roles of A and B with input w' in their interaction with C. At the end of this emulation, $\hat{C}_{\text{CAN}}$ must send a bit to the trusted party (to state whether B should output $\perp$ or receive the same uniform key as A). $\hat{C}_{\text{CAN}}$ checks whether B accepts or rejects in the emulation that it ran. Then, $\hat{C}_{\text{CAN}}$ sends 1 to the trusted party if and only if B accepts. (We note that this accept/reject also appears in C's view, by the definition of REAL.) Following this, $\hat{C}_{\text{CAN}}$ outputs the view of C and halts. We stress that $\hat{C}_{\text{CAN}}$ uses only black-box access to C and does not rewind C at all (this has sometimes been called "straight-line black-box simulation").

We now prove that the distribution generated by an ideal execution with adversary $\hat{C}_{\text{CAN}}$ is at most $O(\epsilon)$ far from the distribution generated by an ideal execution with the adversary $\hat{C}$ that is guaranteed to exist for C (by the security of the protocol). That is,

Claim 2.11 *Let C and $\hat{C}$ be as in Claim 2.10, and let $\hat{C}_{\text{CAN}}$ be as defined above. Then,*

$$\left\{\text{IDEAL}_{\hat{C}_{\text{CAN}}}(\mathcal{D},\sigma)\right\}_{n,\mathcal{D},\sigma} \stackrel{O(\epsilon)}{\equiv} \left\{\text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)\right\}_{n,\mathcal{D},\sigma}$$

Proof: Notice that the canonical ideal-model adversary generates a distribution very similar to that of MENTAL. Therefore, by Claim 2.10, the resulting distribution should be close to IDEAL.

The only difference is with respect to the outputs of A and B (i.e., A's output is U_n instead of its output from the protocol and B's output depends on the bit sent by $\hat{C}_{\text{CAN}}$). Intuitively, due to the security of the protocol, this also makes no difference.

Formally, let DIST be a distribution that is either sampled according to $\text{MENTAL}_C(\mathcal{D},\sigma)$ or $\text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)$. Then, consider the transformation obtained by replacing the second element (i.e., A's output) by a uniformly distributed string $k \leftarrow U_n$, and the third element (i.e., B's output) by either k or $\perp$, depending on the accept/reject bit that appears in C's view (as found in the fourth element of the distribution).

If DIST is sampled according to $\text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)$, then the result of the transformation can be distinguished from IDEAL with probability at most $O(\epsilon)$. This can be seen as follows. First, the output of $\hat{A}$ is identically distributed in both cases. It remains to show that the output of $\hat{B}$ differs with probability at most $O(\epsilon)$. The only time that a difference can occur is if the view output by $\hat{C}$ in the distribution has that $\hat{B}$ accepted, and yet $\hat{B}$'s output is $\perp$, or vice versa. However, the real model adversary's view always contains the correct accept/reject bit of B. Therefore, in the ideal model, this bit in the view output by $\hat{C}$ can be inconsistent with $\hat{B}$'s actual accept/reject bit with probability at most $O(\epsilon)$; otherwise, the real and ideal models can be distinguished with probability greater than $O(\epsilon)$. We conclude that when DIST is sampled according to $\text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)$, the result of the transformation can be distinguished from $\text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)$ with probability at most $O(\epsilon)$.

However, if DIST is sampled according to MENTAL, then the result of the transformation is distributed *exactly* according to $\text{IDEAL}_{\hat{C}_{\text{CAN}}}(\mathcal{D},\sigma)$. This follows directly from the definition of the ideal model and the ideal-model adversary $\hat{C}_{\text{CAN}}$. (Recall that $\hat{C}_{\text{CAN}}$ sends 1 to the trusted party if and only if B accepts in its emulation. Further recall that the accept/reject bit of B correctly appears in C's view as output by $\hat{C}_{\text{CAN}}$ in the emulation.)

Combining the above, we have that any machine distinguishing $\text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)$ and $\text{IDEAL}_{\hat{C}_{\text{CAN}}}(\mathcal{D},\sigma)$ with probability p can be used to distinguish between MENTAL and IDEAL with probability at least $p - O(\epsilon)$. Since, by Claim 2.10, MENTAL and IDEAL can be distinguished with probability at most $O(\epsilon)$, it holds that p (i.e., the probability of distinguishing $\text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)$ and $\text{IDEAL}_{\hat{C}_{\text{CAN}}}(\mathcal{D},\sigma)$) is bound by $O(\epsilon) + O(\epsilon) = O(\epsilon)$. ■

Now, let C be a real-model adversary and let $\hat{C}$ be the ideal-model adversary that is guaranteed to exist by the security of the protocol. By Claim 2.11, we have that the distribution generated by $\hat{C}_{\text{CAN}}$ is only $O(\epsilon)$ far from $\text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)$. Furthermore, by the security of the protocol, we have that

$$\left\{\text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)\right\}_{n,\mathcal{D},\sigma} \stackrel{O(\epsilon)}{\equiv} \left\{\text{REAL}_C(\mathcal{D},\sigma)\right\}_{n,\mathcal{D},\sigma}$$

We conclude that $\text{IDEAL}_{\hat{C}_{\text{CAN}}}(\mathcal{D},\sigma)$ is at most $O(\epsilon)$ far from $\text{REAL}_C(\mathcal{D},\sigma)$. That is,

Lemma 2.12 *Consider any protocol for password-based authenticated session-key generation that is secure according to Definition 2.4, and let C be any real-model adversary. Then, for the ideal-model adversary $\hat{C}_{\text{CAN}}$ as defined above,*

$$\left\{\text{IDEAL}_{\hat{C}_{\text{CAN}}}(\mathcal{D},\sigma)\right\}_{n,\mathcal{D},\sigma} \stackrel{O(\epsilon)}{\equiv} \left\{\text{REAL}_C(\mathcal{D},\sigma)\right\}_{n,\mathcal{D},\sigma}$$

Concurrent executions for many parties. Given the canonical simulator of Lemma 2.12, we can prove Proposition 2.9. The proposition is proved using a standard hybrid argument. The main point is that for any fixed i, the passwords in all other executions are independent of the password in the i^{th} execution. Thus, an adversary can choose the passwords for all other executions by itself, thereby enabling it to internally (and perfectly) simulate these executions (even if they are concurrent to the i^{th} execution).

Proof of Proposition 2.9: Let C be a real-model adversary and let m be the number of concurrent executions. Then, construct a multi-execution ideal-model adversary $\hat{C}$ using the canonical single-execution adversary $\hat{C}_{\text{CAN}}$. Adversary $\hat{C}$ follows the strategy of $\hat{C}_{\text{CAN}}$ independently for each execution. More formally, $\hat{C}$ invokes C with auxiliary input σ and chooses uniform passwords $w_1, \ldots, w_m \in_R \mathcal{D}$ for each pair $(A_1, B_1), \ldots, (A_m, B_m)$. Then, in the i^{th} execution, $\hat{C}$ plays the roles of A_i and B_i with shared password w_i, forwarding all messages between the parties and C. At the conclusion of the execution of (A_i, B_i), adversary $\hat{C}$ sends 1 to the trusted party for the i^{th} execution if and only if B_i accepts. At the conclusion of all executions, $\hat{C}$ outputs the view of C and halts. We now prove that

$$\left\{\text{MULTI-IDEAL}^m_{\hat{C}}(\mathcal{D}, \sigma)\right\}_{n,\mathcal{D},\sigma} \overset{O(m\epsilon)}{\equiv} \left\{\text{MULTI-REAL}^m_{C}(\mathcal{D}, \sigma)\right\}_{n,\mathcal{D},\sigma} \tag{6}$$

In order to prove Eq. (6), we define a hybrid distribution in which the outputs of the first i executions are generated from real executions with C and the outputs of the last $m-i$ executions are generated from ideal executions with $\hat{C}_{\text{CAN}}$ applied to C. That is, the first i executions are pure real executions with C. However, the last $m-i$ executions are ideal executions with adversary $\hat{C}_{\text{CAN}}$. As in a regular ideal execution, $\hat{C}_{\text{CAN}}$ runs (A_j, B_j) and forwards all messages to C. Furthermore, $\hat{C}_{\text{CAN}}$ sends 1 to the trusted party if and only if B_j accepts. We denote this hybrid execution by $\text{HYBRID}^i_{C,\hat{C}_{\text{CAN}}}(\mathcal{D}, \sigma)$.

By the above definition, $\text{HYBRID}^0_{C,\hat{C}_{\text{CAN}}}(\mathcal{D}, \sigma) = \text{MULTI-IDEAL}_{\hat{C}}(\mathcal{D}, \sigma)$ and $\text{HYBRID}^m_{C,\hat{C}_{\text{CAN}}}(\mathcal{D}, \sigma) = \text{MULTI-REAL}_{C}(\mathcal{D}, \sigma)$. It therefore remains to show that for every i,

$$\left\{\text{HYBRID}^i_{C,\hat{C}_{\text{CAN}}}(\mathcal{D}, \sigma)\right\}_{n,\ D,\sigma} \overset{O(\epsilon)}{\equiv} \left\{\text{HYBRID}^{i+1}_{C,\hat{C}_{\text{CAN}}}(\mathcal{D}, \sigma)\right\}_{n,\ D,\sigma} \tag{7}$$

This follows from the fact that $\hat{C}_{\text{CAN}}$ is a "good" ideal-model adversary for a single execution, as shown in Lemma 2.12. That is, construct a single execution adversary C_i who works as follows. C_i internally emulates real executions of C with $(A_1, B_1), \ldots, (A_i, B_i)$ and ideal executions of $\hat{C}_{\text{CAN}}$ (applied to C) with $(A_{i+2}, B_{i+2}), \ldots, (A_m, B_m)$. In contrast, the messages of the $i+1^{\text{th}}$ execution are sent externally. By the definition of the canonical ideal-model adversary, when $\hat{C}_{\text{CAN}}$ is applied to C_i for a single ideal execution, the resulting distribution is exactly $\text{HYBRID}^i_{C,\hat{C}_{\text{CAN}}}(\mathcal{D}, \sigma)$. On the other hand, the result of a real execution with C_i is exactly $\text{HYBRID}^{i+1}_{C,\hat{C}_{\text{CAN}}}(\mathcal{D}, \sigma)$. Therefore, Eq. (7) follows from Lemma 2.12. Applying the hybrid argument m times, we obtain Eq. (6), completing the proof. ■

3 Our Session-Key Generation Protocol

Preliminaries: All arithmetic below is over the finite field $\text{GF}(2^n)$ which is identified with $\{0,1\}^n$. For a review of cryptographic tools used and some relevant notations, see Appendix A.

In our protocol, we use a secure protocol for evaluating *non-constant, linear polynomials* (actually, we could use any family of 1–1 Universal$_2$ hash functions). This protocol involves two parties

A and B; party A has a non-constant, linear polynomial $Q(\cdot) \in \{0,1\}^{2n}$ and party B has a string $x \in \{0,1\}^n$. The functionality is defined by $(Q, x) \mapsto (\lambda, Q(x))$; that is, A receives nothing and B receives the value $Q(x)$ (and nothing else). The postulate that A is supposed to input a non-constant, linear polynomial can be enforced by simply mapping all possible input strings to the set of such polynomials (this convention is used for all references to polynomials from here on). We actually augment this functionality by having A also input a commitment to the polynomial Q (i.e., $c_A \in \text{Commit}(Q)$) and its corresponding decommitment r (i.e., $c_A = C(Q, r)$). Furthermore, B also inputs a commitment value c_B. The augmentation is such that if $c_A \neq c_B$ (or $c_A \notin \text{Commit}(Q)$), then B receives a special failure symbol. This is needed in order to tie the polynomial evaluation to a value previously committed to in the main (higher level) protocol. The functionality is defined as follows:

Definition 3.1 (augmented polynomial evaluation):

- **Inputs:** *The input of Party A consists of a linear, non-constant polynomial Q over* $\text{GF}(2^n)$, *a commitment c_A to this Q, and a corresponding decommitment r. The input of Party B consists of a commitment c_B and a value $x \in \text{GF}(2^n)$.*
- **Outputs:**
  1. Correct Input Case: *If $c_A = c_B$ and $c_A = C(Q, r)$, then B receives $Q(x)$.*
  2. Incorrect Input Case: *If $c_A \neq c_B$ or $c_A \neq C(Q, r)$, then B receives a special failure symbol, denoted $\perp$.*

  In both cases, A receives nothing.

We stress that the relevant input case can be determined in polynomial time because A provides both Q and r. Therefore, the augmented polynomial evaluation functionality is probabilistic polynomial-time and by [53, 30], it can be securely computed. (See Appendix A.1 for the definitions of secure computation.)

3.1 The Protocol

Let f be a one-way permutation and b a hard-core bit of f. A schematic diagram of Protocol 3.2, is provided in Figure 1 (below).

Protocol 3.2 (password-based authenticated session-key generation)

Inputs: Parties A and B start with a joint password w, which is supposed to be uniformly distributed in $\mathcal{D}$.

Outputs: A and B each output an accept/reject bit as well as a session-key, denoted k_A and k_B respectively. The accept/reject bit is a public output, whereas the session-key is a local output.

(In normal operation $k_A = k_B$ and both parties accept. As can be seen below, the public output bit of A will always be accept. We will show that in case $k_A \neq k_B$ the public output bit of B is unlikely to be accept.)

Operation: The protocol proceeds in four stages.

1. **Stage 1: (Non-Malleable) Commit**
   (a) A chooses a random, linear, non-constant polynomial Q over $\text{GF}(2^n)$.

(b) A and B engage in a *non-malleable* (perfectly binding) commitment protocol in which A commits to the string $(Q, w) \in \{0,1\}^{3n}$. Denote the random coins used by B in the commitment protocol (where he plays the role of the receiver) by r_B, and denote B's view of the execution of the commitment protocol by $NMC(Q, w)$.[21]

After the commitment protocol terminates, B sends (the receiver's coins) r_B to A. (This has no effect on neither the hiding property (which refers to what B can learn) nor to the binding property (because the commitment phase is perfectly binding and its execution has already terminated).)

2. **Stage 2: Pre-Key Exchange** – In this stage the parties generates strings π_A and π_B, from which the output session-keys (as well as validation checks performed below) are *derived*. Thus, π_A and π_B are called pre-keys, and the process of generating them is referred to as "pre-key exchange".

   (a) A sends B a (perfectly binding) commitment $c = C(Q, r)$, for a randomly chosen r.[22]

   (b) A and B engage in an augmented polynomial evaluation protocol. A inputs the polynomial Q as well as (c, r); whereas B inputs the password w (viewed as an element of $\mathrm{GF}(2^n)$) as well as c.

   (If indeed both parties input the same commitment value $c = C(Q, r)$ (as well as (Q, r) and w respectively) then B gets the output $Q(w)$.)

   (c) We denote B's output by π_B. (Note that π_B is supposed to equal $Q(w)$.)

   (d) A internally computes $\pi_A = Q(w)$.

3. **Stage 3: Validation**

   (a) A sends the string $y = f^{2n}(\pi_A)$ to B.

   (b) A proves to B in zero-knowledge that she has input the same polynomial in the non-malleable commitment (performed in Stage 1) and in the ordinary commitment (performed in Stage 2(a)), and that the value y is "consistent" with the non-malleable commitment. Formally, A proves the following NP-statement:

   There exists a pair $(X_1, x_2) \in \{0,1\}^{2n} \times \{0,1\}^n$ (where supposedly $X_1 = Q$ and $x_2 = w$) and random coins $r_{A,1}, r_{A,2}$ (where supposedly $r_{A,1}$ and $r_{A,2}$ are A's random coins in the non-malleable and ordinary commitments, respectively) such that the following three conditions hold

   i. B's view of the non-malleable commitment stage (denoted above by $NMC(Q, w)$) is identical to the receiver's view of a non-malleable commitment to (X_1, x_2), where the sender and receiver's respective random coins are $r_{A,1}$ and r_B. (Recall that r_B denotes B's random coins in the non-malleable commitment, and that it has been sent to A at the end of Stage 1(b).)[23]

   ii. $c = C(X_1, r_{A,2})$

   iii. $y = f^{2n}(X_1(x_2))$

   The zero-knowledge proof used here is the *specific* zero-knowledge proof of Richardson and Kilian [47], with a *specific* setting of parameters. Specifically, we refer to the

[21] Recall that B's view consists of his random coins and all messages received during the commitment protocol execution.

[22] The purpose of this additional commitment to the polynomial Q is explained in Footnote 29.

[23] The view of a protocol execution is a function of the parties' respective inputs and random strings. Therefore, the sender's input (X_1, x_2), and the party's coins $r_{A,1}$ and r_B determine a unique possible view. Recall that B sent r_B to A following the commitment protocol. Thus, A has $NMC(Q, w)$ (which includes r_B), the committed-to value (Q, w) and her own coins $r_{A,1}$, enabling her to efficiently prove the above statement.

setting of the number of iterations, denoted m, in the first part of the Richardson-Kilian proof system. We set m to equal the number of rounds in Stages 1 and 2 of our protocol *plus* any non-constant function of the security parameter. For further details, see Appendix A.4.

(c) Let t_A be the session transcript so far as seen by A (i.e., the sequence of all messages sent and received by A so far), and let MAC_k be a message authentication code, keyed by k. Then, A computes $k_1(\pi_A) \stackrel{\text{def}}{=} b(\pi_A)\cdots b(f^{n-1}(\pi_A))$, where b is a hard-core bit of f, and sends the value $m = MAC_{k_1(\pi_A)}(t_A)$ to B.

4. **Stage 4: Decision**

(a) A always accepts and outputs $k_2(\pi_A) \stackrel{\text{def}}{=} b(f^n(\pi_A))\cdots b(f^{2n-1}(\pi_A))$.

(b) B accepts if and only if the following three conditions are satisfied:

i. $y = f^{2n}(\pi_B)$, where y is the string sent by A to B in Step 3(a), and π_B is B's output from the polynomial evaluation (as defined in Stage 2(c)).
(We stress that if $\pi_B = \perp$ then no y fulfills this equality, and B always rejects.)

ii. B accepts the zero-knowledge proof in Step 3(b).

iii. The MAC value received in Stage 3(c) passes as a valid authentication tag for the session-transcript as seen by B, with respect to the MAC-key $k_1(\pi_B) = b(\pi_B)\cdots b(f^{n-1}(\pi_B))$. That is, $\text{Verify}_{k_1(\pi_B)}(t_B, m) = 1$, where t_B is the transcript of Stages 1 through 3(b) as seen by B, the string m is the alleged MAC-tag that B received in Stage 3(c), and MAC-verification is with respect to the MAC-key $k_1(\pi_B) = b(\pi_B)\cdots b(f^{n-1}(\pi_B))$.

In case B accepts, he outputs the session-key $k_2(\pi_B) = b(f^n(\pi_B))\cdots b(f^{2n-1}(\pi_B))$, otherwise he outputs $\perp$. (Recall that the accept/reject decision bit is considered a public output.)

We stress that A and B always accept or reject based solely on these criteria, and that they do not halt (before this stage) even if they detect malicious behavior.

In our description of the protocol, we have referred only to parties A and B. That is, we have ignored the existence (and possible impact) of the channel C. That is, when A sends a string z to B, we "pretend" that B actually received z and not something else. In a real execution, this may not be the case at all. In the actual analysis we will subscript every value by its owner, as we have done for π_A and π_B in the protocol. For example, we shall say that in Step 3(a), A sends a string y_A and the string received by B is y_B.

3.2 Motivation for the Protocol

We suggest to the reader to start by considering the schematic diagram of Protocol 3.2, as provided in Figure 1.

3.2.1 On the general structure of the protocol

The central module of Protocol 3.2 is the secure polynomial evaluation. As suggested by Naor and Pinkas [44], this module (by itself) suffices for achieving security against *passive* channels (but not against *active* ones). Specifically, consider the following protocol. Party A chooses a random, linear polynomial Q and inputs it into a secure polynomial evaluation with party B who inputs the joint password w. By the definition of the polynomial evaluation, B receives $Q(w)$ and A receives

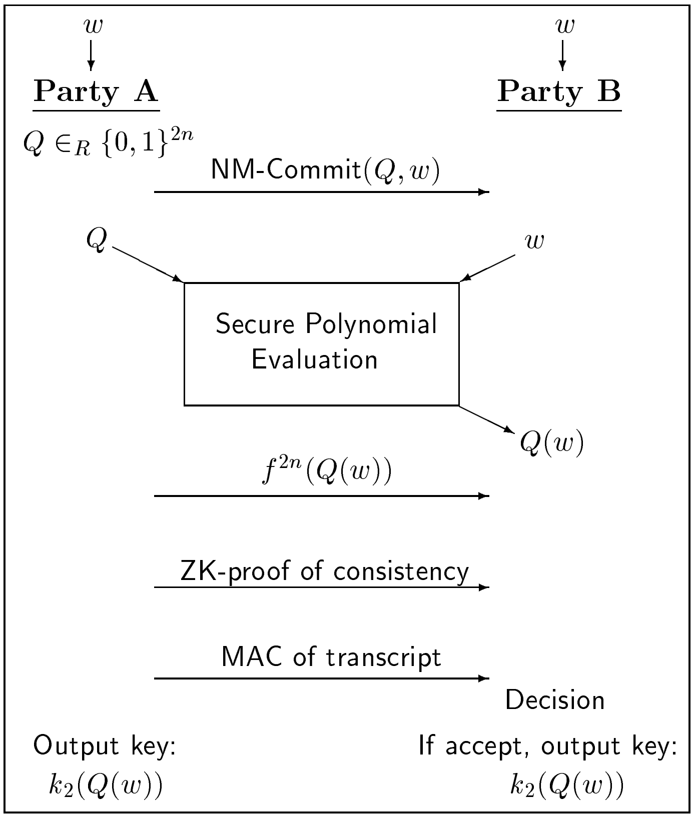

Figure 1: Schematic Diagram of the Protocol.

nothing. Next, A internally computes $Q(w)$ (she can do this as she knows both Q and w), and both parties use this value as the session-key. The key is uniformly distributed (since Q is random and linear) and due to the secrecy requirements of the polynomial evaluation, the protocol reveals nothing of w or $Q(w)$ to a passive eavesdropper C (since otherwise this would also be revealed to party A who should learn nothing from the evaluation). generates

One key problem in extending the above argument to the *active* channel setting is that the standard security definitions of two-party computation (which refer to the stand-alone setting) guarantee nothing about what happens in the concurrent setting. In fact, one can show that the simplified protocol (as outlined in the previous paragraph) is not secure against an active adversary. We now provide some intuition as to why our protocol, which is derived via significant augmentations of the simplified protocol, is nevertheless secure.

First, assume that the MAC-value sent by A at the conclusion of the protocol is such that unless C "behaved passively" (i.e., relayed all message without modification), then B rejects (with some high probability). here MAC Now, if C behaves passively, then B clearly accepts (as in the case of honest parties A and B that execute the protocol without any interference). On the other hand, if C does not behave passively, then (by our assumption regarding the MAC-value) B rejects. However, C itself knows whether or not it behaved passively and therefore can predict whether or not B will reject. In other words, the accept/reject bit output by B is simulatable (by C itself). We proceed by observing that this bit is the only meaningful message sent by B during the protocol: apart from in the polynomial evaluation, the only messages sent by B are as the *receiver* in a non-malleable commitment protocol and the *verifier* in a zero-knowledge proof (clearly, no knowledge of the password w is used by B in these protocols). Furthermore, the polynomial evaluation is such that only B receives output. Therefore, intuitively, the input used by B is not revealed by the execution; equivalently, the view of C is (computationally) independent of B's input w (this can be shown to hold even in our concurrent setting). We conclude that all messages sent by B during the execution can be simulated without knowledge of w. Therefore, by indeed simulating B, we can reduce the *concurrent* scenario involving A, C and B to a (standard) two-party setting between A and C. In this (standard) setting, we can apply standard tools and techniques for simulating C's view in its interaction with A, and conclude that the entire real execution is simulatable in the ideal model. Hence, assuming that the MAC-value is accepted if and only if C "behaves passively", the protocol is secure.

Thus, the basis for simulating C's view (which means security of our protocol) lies in the security of the MAC in our scenario. Indeed, the MAC is secure when the parties using it (a priori) share a random MAC-key; but in our case the parties establish the MAC-key during the protocol itself, and it is not clear that this key is random nor the same in the view of both parties. In order to justify the security of the MAC (in our setting), two properties must be shown to hold. Firstly, we must show that with high probability either A and B hold the *same* MAC-key or B is going to reject anyhow (and C knows this). Secondly, we need to show that this (identical) MAC-key held by A and B has "sufficient pseudorandomness" to prevent C from successfully forging a MAC-value.

In some sense, we are back to the original problem (of generating a shared secret key). However, notice that the security of the MAC-key only needs to hold *before* B concludes and outputs accept or reject. Since proving the security of the MAC-key does not require the simulation of B's output accept/reject bit, it is easier to prove than the security of the shared session-key.[24] Nevertheless, this part of the proof is still the hardest.

3.2.2 On some specific choices

Using a pseudorandom generator. In the protocol, we implicitly use a pseudorandom generator defined by $G(s) = b(s) \cdots b(f^{2n-1}(s)) \cdot f^{2n}(s)$. As discussed in Appendix A.5, this is a "seed-committing" pseudorandom generator (i.e., $f^{2n}(s)$ uniquely determines s). To see why this type of a pseudorandom generator is relevant to us, recall that as part of the validation stage, some function F of π_B is sent by A to B, whereas another function k_1 (of π_B) is used to derive the MAC-key, and yet another function (i.e., k_2) is used to derive the output session-key. The properties required from F are that firstly it be 1–1 (so that $F(\pi_B)$ uniquely determines π_B), and secondly that the MAC-key and the output session-key (also derived from π_B) be pseudorandom, even though the adversary is given $F(\pi_B)$. Viewed in this light, using a seed-committed pseudorandom generator (while setting $F(\cdot) = f^{2n}(\cdot)$ and $G(\cdot) = k_1(\cdot)k_2(\cdot)f^{2n}(\cdot))$ is a natural choice.

On the use of linear polynomials. The pre-keys are generated by applying a random, linear, non-constant polynomial on the password. Such a polynomial is used for the following reasons. Firstly, we need "random 1–1 functions" that map each dictionary entry to a uniformly distributed n-bit string. The 1–1 property is used in saying that Q and π uniquely determine w such that $Q(w) = \pi$.[25] Secondly, we desire that for $w' \neq w$, the values $Q(w')$ and $Q(w)$ be (almost) independent. This ensures that if C guesses the wrong password and obtains $Q(w')$, it will gain

[24] Indeed, the fact that B has not yet output its accept/reject bit is crucial in our proof of the unforgeability of the MAC.

[25] In particular, if a constant polynomial is allowed then C could choose a constant Q' and run the entire protocol with B using Q'. Since Q' is constant, $\pi_B = Q'(w)$ is a fixed value and is thus KNOWN to C. Furthermore, C

no information on the actual key $Q(w)$. (Essentially, any family of 1–1 Universal$_2$ hash functions would be appropriate.)

3.3 Properties of Protocol 3.2

The main properties of Protocol 3.2 are captured by the following theorem.

Theorem 3.3 *Suppose that all the cryptographic tools used in Protocol 3.2 satisfy their stated properties. Then Protocol 3.2 constitutes a secure protocol for* (augmented) *password-based authenticated session-key generation* (as defined in Definition 2.5).

As we have mentioned above, Protocol 3.2 also fulfills the additional property of intrusion detection.

Protocol 3.2 as a feasibility result: All the cryptographic tools used in Protocol 3.2 can be securely implemented assuming the existence of *trapdoor permutations*. Thus, at the very least, Theorem 3.3 implies the feasibility result captured by Theorem 2.7. (We note that the zero-knowledge proof of [47] can be implemented using one-way permutations only; see Appendix A.4.)

Protocol 3.2 as a basis for efficient solutions: We now briefly discuss the efficiency of our protocol. From this perspective, the most problematic modules of the protocol are the non-malleable commitment, the secure (augmented) polynomial evaluation, and the zero-knowledge proof of [47]. Focusing on round complexity, we make the following observations: First, by a recent general result of Lindell [38] (building on [53, 30]), the secure (augmented) polynomial evaluation can be implemented in a constant number of rounds. Second, by using a round-efficient version of the zero-knowledge proof of [47] based on claw-free functions (see Appendix A.4), the number of rounds of communication required for this proof is $m + O(1)$, where m equals the number of rounds in the first two stages of our protocol *plus* some non-constant function in the security parameter (say $\log \log n$). In fact, as discussed in Section 6.1.1, this can be reduced to a single additional round. We thus conclude that the main bottleneck with respect to the number of rounds of communication is due to the non-malleable commitment. In a recent result by Barak [1], it was shown (under stronger complexity assumptions) that non-malleable commitments can be achieved for a constant number of rounds. Thus, using this non-malleable commitment, we indeed obtain a constant-round protocol.

Turning to the bandwidth (i.e., length of messages) and the computational complexity of our protocol, we admit that both are large, but this is due to the corresponding complexities of the problematic modules mentioned above. Any improvement in the efficiency of these modules (which is, fortunately, an important open problem) would yield a corresponding improvement in the efficiency of our protocol.

4 Analysis of Protocol 3.2: Proof Sketches

Regrettably, due to reasons mentioned in the introduction and further discussed below, the analysis of Protocol 3.2 is quite involved. In order to focus on the main ideas of this analysis, we provide its essence in the current section, while deferring some details to subsequent sections.

can execute the zero-knowledge proof in the validation stage correctly, because $y = f^{2n}(Q'(w)) = f^{2n}(Q'(w'))$ is consistent with $NMC(Q', w')$ for *every* w' (rather than only for $w' = w$). We conclude that (under such a flawed modification) B accepts with a session key known to C, in contradiction to the session-key secrecy requirement.

4.1 Preliminaries

Recall that the (adversarial) channel (or adversary) C may omit, insert and modify any message sent between A and B. Thus, in a sense C conducts two *separate* executions: one with A in which C impersonates B (called the (A, C) execution), and one with B in which C impersonates A (called the (C, B) execution). These two executions are carried out *concurrently* (by C), and there is no explicit execution between A and B. Furthermore, C has full control of the *scheduling* of the (A, C) and (C, B) executions (i.e., C may maliciously decide when to pause one execution and continue the other). For this reason, throughout the proof we make statements to the effect of: "when A executes X in her protocol with C then...". This reflects the fact that the separate (A, C) and (C, B) executions may be at very different stages.

We note that there are currently no tools for dealing with (general) *concurrent* computation in the two-party case.[26] Our solution is therefore based on an ad-hoc analysis of (two) concurrent executions of specific two-party protocols that are secure as stand-alone (i.e., when only two parties are involved and they conduct a single execution over a direct communication line). Our analysis of these executions proceeds by using specific properties to remove the concurrency and obtain a reduction to the stand-alone setting. That is, we show how an adversarial success in the concurrent setting can be translated into a related adversarial success in the stand-alone setting. This enables us to analyze the adversary's capability in the concurrent setting, based on the security of two-party stand-alone protocols.

We stress that we make no attempt to minimize the constants (in $O(\epsilon)$ terms) in our proofs. In fact, some of our proofs are clearly wasteful in this sense and the results we obtain are not tight. Our main objective is to make our (regrettably complex) proofs as modular and simple as possible.

Channel's output and view: We will assume, without loss of generality, that the adversary's output always includes its view of the execution (because the adversary can always be modified so that this holds). In fact, the reader may assume (also without loss of generality) that the adversary's output always equals its view (because the output is always efficiently computable from the view).

Reliable channels: For the proof, we define the concept of a *reliable* channel. We say that a channel C is *reliable in a given protocol execution* if C runs the (A, C) and (C, B) executions in a synchronized manner and does not modify any message sent by A or B. That is, any message sent by A is immediately forwarded to B (without modification), and vice versa. This property is purely syntactic and relates only to the bits of the messages sent in a *given* execution of the protocol. In essence, an execution for which C is reliable looks like an execution via a passive adversary. However, C may decide at any time during the protocol execution to cease being reliable (this decision is possibly based on its current view and may be probabilistic). This is in contrast to a passive adversary who, by definition, only eavesdrops on the communication.

Notation: We present some notation that is used throughout the proof. As we have seen, $C^{A(w),B(w)}$ denotes an execution of C with A and B, where the parties' joint password is w. Likewise, denote by $C^{A(Q,w),B(w)}$ an execution of C with A and B, where A is modified so that she receives the random (non-constant, linear) polynomial Q to be used in the protocol as additional input (recall that A's input in the protocol is defined to be the password w only). We note that

[26] There is work relating to concurrently-secure honest-majority computation (cf. [16]). However, this does not apply to the two-party case and furthermore assumes the existence of direct (and reliable) communication channels between all parties.

such a modification makes no difference to the outcome since in the protocol definition, party A begins by choosing such a random polynomial Q. This modification is made for the sake of the analysis and enables us to refer explicitly to Q when, for example, relating to the session-key output by A, which is defined as $k_2(Q(w))$ in the protocol. Sometimes in the proof, we refer to stand-alone executions of an adversary with A or B. In such a case, we denote by $C^{A(Q,w)}$ (resp., $C^{B(w)}$) a stand-alone execution of the protocol between C and A (resp., B).

We note that, for the sake of simplicity, we often omit explicit reference to C's auxiliary input σ, and therefore write $C^{A(Q,w),B(w)}$ rather than $C^{A(Q,w),B(w)}(\sigma)$. All our proofs do, however, hold with respect to such an auxiliary input.

Throughout our proof, it is often important to consider the accept/reject decision-bit output by B (recall that this bit is public and therefore known to C). We denote by "$\mathrm{dec}_B = \mathsf{acc}$" the case that B outputs accept, and likewise "$\mathrm{dec}_B = \mathsf{rej}$" the case that B outputs reject. We also often refer to the event that C is reliable or not. Thus, we denote "$\mathrm{reliable}_C = \mathsf{true}$" if C was reliable in the given execution, and "$\mathrm{reliable}_C = \mathsf{false}$" otherwise.

The basic and augmented definitions of security: We prove that Protocol 3.2 is a secure password-based authenticated session-key generation protocol with respect to the basic definition (i.e., Definition 2.4). The proof of security with respect to the augmented definition (i.e., Definition 2.5) is obtained by minor modifications, which are noted where relevant. Our choice of presenting the proof with respect to Definition 2.4 is due to the desire to avoid the more cumbersome formalism of Definition 2.5, while realizing that the main issues of security arise already in Definition 2.4.

4.2 Organization and an Outline of the Proof

Due to the length and complexity of our proof, we leave the full proofs of the central lemmas to later sections. Instead, intuitive proof sketches are provided in-place. Unless otherwise stated, the sketches are quite precise and the full proofs are derived from them in a straightforward manner.

The cases of *passive* and *active* adversaries (i.e., Parts 1 and 2 of Definition 2.4) are dealt with separately. The proof of security against passive adversaries can be found in Section 4.3 (with further details in Section 5). On the other hand, the proof sketches for the case of active adversaries span Sections 4.4 to 4.7, with the full proofs presented in Sections 6 to 8.

We now outline the high-level structure of the proof of security against *active* adversaries. Conceptually, our proof works by first simulating B's role in the (C, B)-execution, and thus reducing the entire analysis to one of a stand-alone (A, C)-execution. However, in order to do this simulation, we need to show how B's accept/reject bit can be simulated (see the motivating discussion in Section 3.2.1). The main property needed for this task is what we call *key-match.* This property states that the probability that B accepts and yet the pre-keys are different (i.e., $\pi_B \neq \pi_A$) is at most $O(\epsilon) + \mu(n)$. (Recall that the pre-key π_B is B's output from the polynomial evaluation and $\pi_A = Q(w)$.) Then, given the key-match property, we are able to show the simulatability of B's accept/reject bit, and thus the simulatability of the entire (C, B) execution. Specifically, we show that for every C interacting with A and B, there exists an adversary C' interacting with A only, such that

$$\{w, k_2(Q(w)), \mathrm{output}(C^{A(Q,w),B(w)})\} \overset{O(\epsilon)}{\equiv} \{w, k_2(Q(w)), \mathrm{output}(C'^{A(Q,w)})\} \qquad (8)$$

Then, we continue by proving that C''s view in this two-party (stand-alone) setting with A only, can also be simulated. Specifically, we show that for every C' interacting with A only, there exists a non-interactive machine C'' such that

$$\{w, k_2(Q(w)), \text{output}(C'^{A(Q,w)})\} \overset{O(\epsilon)}{\equiv} \{w, U_n, \text{output}(C'')\} \tag{9}$$

Combining Equations (8) and (9) brings us quite close to proving that the IDEAL and REAL distributions are $(1 - O(\epsilon))$-indistinguishable. To see this, recall that in the real-model A always outputs $k_2(Q(w))$, and in the ideal-model $\hat{A}$ always outputs U_n. Thus, the above equations imply the existence of a non-interactive machine $\hat{C}$ (similar to the ideal-model machine) for which

$$\{w, \text{output}(A), \text{output}(C^{A(w),B(w)})\} \overset{O(\epsilon)}{\equiv} \{w, \text{output}(\hat{A}), \text{output}(\hat{C})\} \tag{10}$$

However, this is not enough since the IDEAL and REAL distributions also include B's output. Therefore, Eq. (10) must be "extended" to include B's output as well. This is achieved by using (a consequence of) the key-match property described above.

The key-match property is proven in Sections 4.4 and Section 6. Next, the proof of Eq. (9) is presented (in Sections 4.5 and 7). (Conceptually, this proof should come after the proof of Eq. (8); however, significant parts of it are used in order to prove Eq. (8) and thus the order is reversed.) Finally, Eq. (8) is proven in Sections 4.6 and 8, and the "extension" of Eq. (10) to complete the proof is shown in Section 4.7. These dependencies are shown in Figure 2.

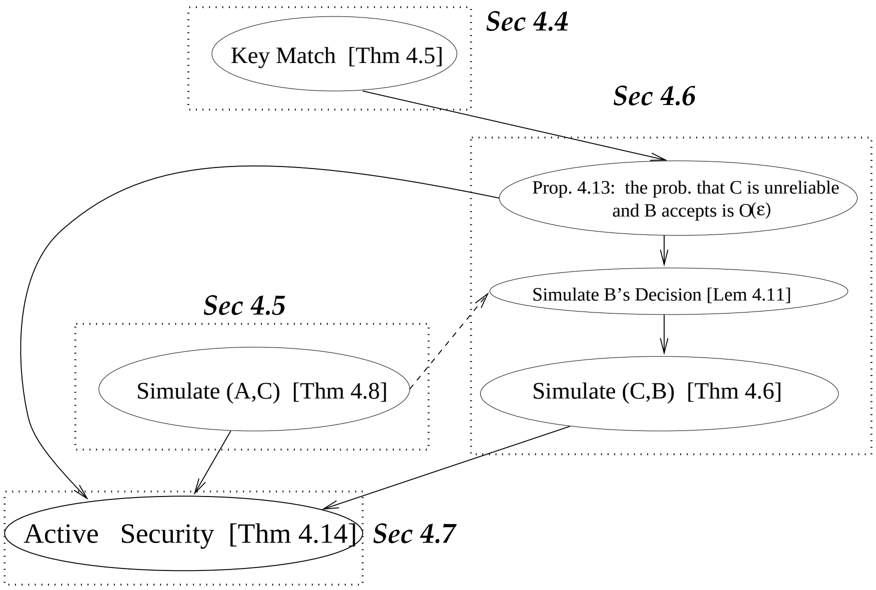

Figure 2: The structure of the proof of security for active adversaries. Solid arrows show direct applications of results, whereas dashed arrows show adaptation of proof techniques.

We remark that while proving the key-match property, we show how (and under what circumstances) A's zero-knowledge proof can be simulated in our concurrent setting (Section 6.1.1). We do not know how to show this using any zero-knowledge proof; rather our simulation utilizes properties of the *specific* proof system of Richardson and Kilian. Furthermore, the "zero-knowledge property" of the proof system in our setting is *not* derived merely from the fact that the Richardson and Kilian system proof is concurrent zero-knowledge, but rather from its specific structure (which is the key to its being concurrent zero-knowledge). Note that concurrent zero-knowledge only refers to a setting where many instances of the *same proof system* are run concurrently, but says nothing about a setting (such as ours) in which the proof system is run concurrently with other protocols. Furthermore, it does not cover settings where the adversary controls a prover in one instance and a verifier in another (i.e., man-in-the-middle attacks).

4.3 The Security of Protocol 3.2 for Passive Adversaries

In this section, we consider the case of a passive adversarial channel. In this case, the IDEAL and REAL distributions are required to be computationally indistinguishable (rather than being just $(1 - O(\epsilon))$-indistinguishable).

Notice that in the case that the channel C is passive, the setting is actually that of standard two-party computation, in which both parties are honest and the adversary can only eavesdrop on their communication. Despite this, the definitions of multi-party computation do not immediately imply that C cannot learn anything in this context. This is because the definitions relate to an adversary C who "corrupts" one or more parties. However, here we are dealing with the case that C corrupts *zero* parties and we must show that in this case, C learns nothing about *any* party's inputs or outputs.

Theorem 4.1 (passive executions): *Protocol 3.2 satisfies Condition 1 in Definition 2.4. That is, for every ppt real-model* passive *adversary C there exists a ppt ideal-model adversary $\hat{C}$ that always sends 1 to the trusted party such that*

$$\left\{\mathrm{IDEAL}_{\hat{C}}(\mathcal{D}, \sigma)\right\}_{n,\mathcal{D},\sigma} \stackrel{\mathrm{c}}{\equiv} \left\{\mathrm{REAL}_{C}(\mathcal{D}, \sigma)\right\}_{n,\mathcal{D},\sigma}$$

where $\mathcal{D} \subseteq \{0,1\}^n$ is any ppt samplable dictionary and $\sigma \in \{0,1\}^{\mathrm{poly}(n)}$ is the auxiliary input for the adversary.

Proof: We first note that in this case parties A and B both output the same session-key, $k_2(Q(w))$, and they both accept. Thus, it is enough to prove the following lemma:

Lemma 4.2 *For every* passive *ppt channel C,*

$$\left\{w, k_2(Q(w)), \mathrm{output}(C^{A(Q,w),B(w)}(\sigma))\right\} \stackrel{\mathrm{c}}{\equiv} \left\{w, U_n, \mathrm{output}(C^{A(Q,\tilde{w}),B(\tilde{w})}(\sigma))\right\}$$

where Q is a random non-constant linear polynomial, and w and $\tilde{w}$ are independently and uniformly distributed in $\mathcal{D}$.

This lemma implies the theorem because the ideal-model adversary $\hat{C}$ can simulate an execution for the real-model adversary C by choosing Q and $\tilde{w}$ and invoking $C^{A(Q,\tilde{w}),B(\tilde{w})}(\sigma)$. Furthermore, since C is passive, A and B's outputs are always identical, and equal to $k_2(Q(w))$ in the real model and U_n in the ideal model. The full REAL and IDEAL distributions can thus be derived from the distributions in the lemma by simply repeating the second element twice. The theorem therefore follows directly from Lemma 4.2.

The proof of Lemma 4.2 can be found in Section 5. Since C is a passive adversary (in this case), the proof is relatively straightforward and is based on the security of two-party protocols. ■

Security for executions in which C is reliable: We now strengthen the "passive adversary" requirement to include executions in which an active adversary C is reliable. Loosely speaking, we show that in real executions for which C is reliable, the output distribution is computationally distinguishable from in the ideal model. This is a stronger result because a passive channel is always reliable, but an active channel may sometime be reliable and sometimes not. Furthermore, such an active channel may dynamically decide whether or not to be reliable, possibly depending on what occurs during the protocol execution. Despite this, we show that in a given execution for which the channel is reliable, it can learn no more than if it was passive.

Proposition 4.3 *For every ppt real-model adversary C there exists an ideal adversary $\hat{C}$ such that for every ppt distinguisher D, for every polynomial $p(\cdot)$, all sufficiently large n's and all auxiliary input $\sigma \in \{0,1\}^{\mathrm{poly}(n)}$,*

$$\begin{aligned}|\Pr[D(\text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)) = 1 \ \&\ \text{reliable}_C = \mathsf{true}]& \\ -\Pr[D(\text{REAL}_C(\mathcal{D},\sigma)) = 1 \ \&\ \text{reliable}_C = \mathsf{true}]| &< \frac{1}{p(n)}\end{aligned}$$

where, in the first probability, reliable$_C$ *refers to whether or not the execution view of the channel C as included in the output of $\hat{C}$ indicates that C is reliable in the said execution.*[27]

Proof: As in Theorem 4.1, in executions for which C is reliable, both A and B output $k_2(Q(w))$ and both accept. Thus it is enough to show an equivalent of Lemma 4.2 for a reliable channel (rather than a passive channel). This is shown using the following claim:

Claim 4.4 *For every ppt* active *channel C there exists a* passive *channel C' such that for every ppt distinguisher D and every randomized process $z = Z(Q,w)$*

$$\Pr[D'(z, \text{output}(C'^{A(Q,w),B(w)})) = 1] = \Pr[D(z, \text{output}(C^{A(Q,w),B(w)})) = 1 \ \&\ \text{reliable}_C = \mathsf{true}]$$

where $D'(z,0) \stackrel{\text{def}}{=} 0$ and $D'(z,\gamma) \stackrel{\text{def}}{=} D(z,\gamma)$ otherwise (i.e., for any $\gamma \neq 0$).

Proof Sketch: The proof is based on having C' emulate an execution for C. Since C' is passive, it receives a message transcript of messages sent between A and B. Channel C''s emulation involves passing the messages of the transcript (in order) to C, and checking whether or not C is reliable (i.e., forwards all messages immediately and unchanged to their intended receiver). If C is not reliable (and thus C' cannot continue the emulation), then C' halts and outputs 0. On the other hand, if C is reliable throughout the entire execution, then C' outputs whatever C does from the experiment. The equality is obtained by considering the following two cases: In case C is reliable, C''s emulation is perfect and the output of D' equals the output of D (because the output of C', which equals the output of a reliable C, is definitely not 0). On the other hand, in case C is unreliable, C' outputs 0 and so does D'. ■

Using Claim 4.4, we obtain the analogous result of Lemma 4.2 for reliable channels. That is, we prove that for every ppt distinguisher D

$$\left|\Pr\left[\begin{array}{r} D(w, k_2(Q(w)), \text{out}(C^{A(Q,w),B(w)})) = 1 \\ \&\ \text{reliable}_C = \mathsf{true}\end{array}\right] - \Pr\left[\begin{array}{r} D(w, U_n, \text{out}(C^{A(Q,\tilde{w}),B(\tilde{w})})) = 1 \\ \&\ \text{reliable}_C = \mathsf{true}\end{array}\right]\right| = \mu(n) \tag{11}$$

[27]Recall that the output of C, included in REAL$_C$, contains the view of C. Thus, it is natural to assume that the output of $\hat{C}$, included in IDEAL$_{\hat{C}}$, also contains such a view. Formally, we may use a parsing rule that applied to $\hat{C}$'s output (included in IDEAL$_{\hat{C}}$), always yields some legal view of C. Alternatively, if $\hat{C}$'s output (included in IDEAL$_{\hat{C}}$) does not contain such a legal view, we define reliable$_C$ to be false.

where out$(\cdot)$ is shorthand for output$(\cdot)$. Eq. (11) follows by applying Claim 4.4 to each of the two probabilities in the equation (once setting $z = (w, k_2(Q(w)))$ and once setting $z = (w, U_n)$ (while switching the roles of w and $\tilde{w}$)), and applying Lemma 4.2 to the result. Using Eq. (11), the proposition follows (analogously to the way Theorem 4.1 follows from Lemma 4.2). ■

Security for the augmented definition: In the case that C is passive (or reliable), the session-key challenge received (in the augmented setting) after the first party terminates is of no consequence. This is because C (being passive or just reliable) makes no use of this message (it simply becomes a part of its view). Therefore, the distinguisher (in the basic setting), who always receives the session-key (since it is part of the output distribution), can generate the output distribution of the augmented setting. Thus, in case C is passive, the basic definition implies the augmented one.

4.4 The Key-Match Property

We now prove the *key-match* property, which states that the probability that A and B both accept but have different pre-keys is at most $O(\epsilon)$. This specific property will be used to establish the security of the entire protocol. Recall that the pre-keys are $\pi_A \stackrel{\text{def}}{=} Q(w)$ and π_B, where π_B is B's output from the polynomial evaluation (conducted in Stage 2).

Theorem 4.5 (key-match): *For every ppt adversarial channel C, every polynomial $p(\cdot)$ and all sufficiently large n's*

$$\Pr[\mathrm{dec}_B = \mathsf{acc}\ \&\ \pi_A \neq \pi_B] < 3\epsilon + \frac{1}{p(n)}$$

Proof outline and rough sketch: The analysis is partitioned into two *complementary* subcases related to the scheduling of the two executions (i.e., C's execution with A and C's execution with B). The scheduling of these two executions may be crucial with respect to the non-malleable commitments. This is because the definition of non-malleability (only) states that a commitment is non-malleable when executed concurrently with another commitment.[28] In an execution of our protocol, the commitment from C to B may be executed concurrently with the polynomial evaluation and/or validation stage of the (A, C) execution. In this case, it is not clear whether or not the non-malleable property holds.

We therefore prove the theorem by considering two possible strategies for C with respect to the scheduling of the (A, C) and (C, B) executions. In the first case, hereafter referred to as the *unsynchronized case*, we consider what happens if C completes the polynomial evaluation with A **before** completing the non-malleable commitment with B. In this case, the entire (A, C) execution may be interleaved with the (C, B) non-malleable commitment. However, according to this scheduling, we are ensured that the (A, C) and (C, B) polynomial evaluation stages are run at different times (with no overlap). Loosely speaking, this means that the polynomial Q_C input by C into the (C, B) evaluation is sufficiently *independent* of the polynomial Q input by A in the (A, C) evaluation. Recall that in the (A, C) execution, C only learns the value of $Q(\cdot)$ at a single point, which we denote w_C. Therefore, for every $w' \neq w_C$, the values $Q_C(w')$ and $Q(w')$ are independently distributed. In particular, unless $w_C = w$ (which occurs with probability at most ϵ), the random variables $Q(w)$ and $Q_C(w)$ are independently distributed, and so C has no clue regarding the value

[28] In fact, by the definition, non-malleability is only guaranteed if the commitments are of the same scheme. Two different non-malleable commitment schemes are *not* guaranteed to be non-malleable if executed concurrently.

of $Q_C(w)$ (even if its view is augmented by the value $Q(w)$). Therefore, the probability that the "y value" sent by C to B will match $f^{2n}(Q_C(w))$ is at most ϵ. We conclude that B will reject with probability at least $1 - O(\epsilon)$.

In the other possible scheduling, C completes the polynomial evaluation with A **after** completing the non-malleable commitment with B. In this case, hereafter referred to as the *synchronized case*, only the first two stages of the (A, C) execution may be run concurrently with the non-malleable commitment of the (C, B) execution (and so these executions are more synchronized than in the previous case). In this case we show how the (A, C) pre-key exchange can be simulated, and we thus remain with a concurrent execution containing two non-malleable commitments only. Non-malleability now holds and this prevents C from modifying the commitment sent by A, if B is to accept. This yields the key-match property. □

Further details on the proof of Theorem 4.5: We prove that for each of the two scheduling cases, the probability that this case holds and the event referred to in Theorem 4.5 occurs (i.e., B accepts and there is a key *mismatch*) is at most $O(\epsilon)$. A schematic description of the two cases is given in Figure 3. Using the Union Bound, Theorem 4.5 follows. That is, Theorem 4.5 is obtained by combining the following Lemmas 4.6 and Lemma 4.7, which refer to the two corresponding scheduling cases.

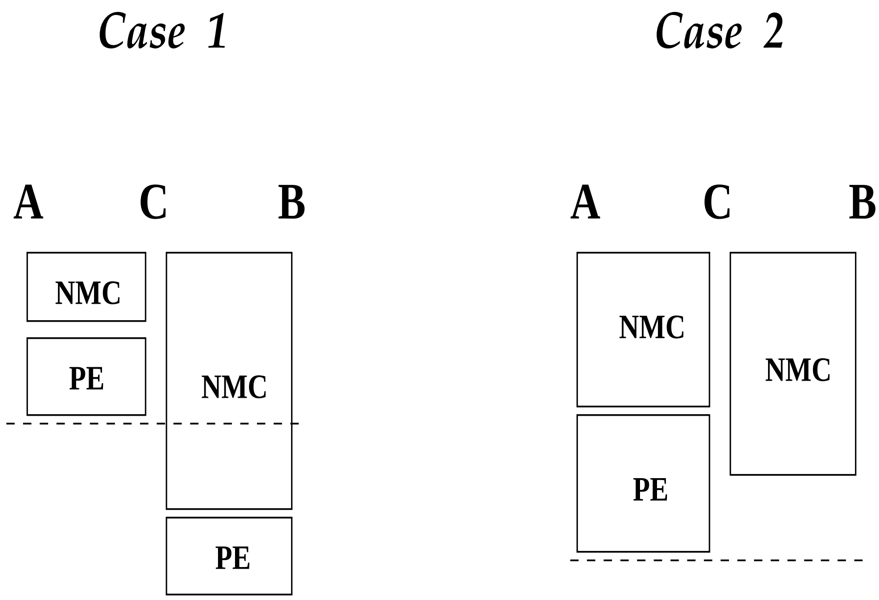

Figure 3: The two scheduling cases. NMC stands for non-malleable commitment, and PE stands for polynomial evaluation.

4.4.1 Case (1) – The unsynchronized case

In this case, C completes the polynomial evaluation with A **before** completing the non-malleable commitment with B. We actually prove a stronger claim here. We prove that according to this scheduling, B accepts with probability less than $2\epsilon + \mu(n)$ irrelevant of the values of π_A and π_B. This is enough because

$$\Pr[\mathrm{dec}_B = \mathsf{acc} \ \&\ \pi_A \neq \pi_B \ \&\ \text{Case 1}] \leq \Pr[\mathrm{dec}_B = \mathsf{acc} \ \&\ \text{Case 1}]$$

Lemma 4.6 (Case 1 – unsynchronized): *Let C be a ppt channel and define* Case 1 *to be a scheduling of the protocol execution by which C completes the polynomial evaluation with A* before *concluding the non-malleable commitment with B. Then for every polynomial $p(\cdot)$ and all sufficiently large n's*

$$\Pr[\mathrm{dec}_B = \mathsf{acc} \ \&\ \text{Case 1}] < 2\epsilon + \frac{1}{p(n)}$$

Proof Sketch: In this case, the (C, B) polynomial evaluation stage is run strictly after the (A, C) polynomial evaluation stage, and the executions are thus "independent" of each other. That is, the polynomial evaluations are executed *sequentially* and not concurrently. For the sake of simplicity, assume that the entire protocol consists of a single polynomial evaluation between A and C and a single polynomial evaluation between C and B. Then, since the evaluations are run sequentially, a party P can interact with C and play A's role in the (A, C) execution and B's role in the (C, B) execution. Thus, we can reduce our concurrent setting to a two-party setting between C and P. In this setting, C and P run two sequential polynomial evaluations: in the first polynomial evaluation the party P (playing A's role) inputs a polynomial Q and the party C inputs some w_C, whereas in the second polynomial evaluation C inputs a polynomial Q_C and P (playing B's role) inputs w. In the first polynomial evaluation C is supposed to obtain the output $Q(w_C)$, whereas in the second polynomial evaluation C is supposed to get nothing and B is supposed to get $Q_C(w)$. The channel C "succeeds" if it can guess $Q_C(w)$ (this is "comparable" to C successfully causing B to accept by sending the correct value for $y = f^{2n}(Q_C(w))$). In this (simplified) two party setting, it can be shown that P can only succeed with probability ϵ (since C learns nothing about w from the execution).

The actual reduction is more involved, since the (A, C) and (C, B) protocols involve other steps beyond polynomial evaluation. Furthermore, some of these steps may be run concurrently (unlike the polynomial evaluations which are executed sequentially according to this scheduling). do Therefore, the main difficulty in the proof is in defining a two-party protocol between C and P that correctly emulates the concurrent execution of our *entire* protocol (subject to the two polynomial evaluations remaining sequential). Among other things, our proof utilizes properties of the specific zero-knowledge proof of Richardson and Kilian [47]. We note that the way we solve this problem in the full proof also handles the issues arising in connection with the augmented definition of security (Def. 2.5). ∎

The full proof of Lemma 4.6 is presented in Section 6.1, and (as hinted above) is far more complex than the above proof sketch.

4.4.2 Case (2) – The synchronized case

We now show that the probability that C completes the polynomial evaluation with A **after** completing the non-malleable commitment with B **and** the bad event referred to in Theorem 4.5 occurs (i.e., B accepts and there is a pre-key *mismatch*) is less than $\epsilon + \mu(n)$.

Lemma 4.7 (Case 2 – synchronized): *Let C be a ppt channel and define* Case 2 *to be a scheduling of the protocol by which C completes the polynomial evaluation with A* after *completing the non-malleable commitment with B. Then for every polynomial $p(\cdot)$ and for all sufficiently large n's,*

$$\Pr[\mathrm{dec}_B = \mathsf{acc} \ \&\ \pi_A \neq \pi_B \ \&\ \text{Case 2}] \ < \ \epsilon + \frac{1}{p(n)}$$

Proof Sketch: As we have mentioned, in this scheduling case we can show that the non-malleability property holds with respect to A's commitment to the pair (Q, w). This is because the (C, B) non-malleable commitment is run concurrently only with the (A, C) non-malleable commitment and pre-key exchange. The main point is that the pre-key exchange can be simulated using only knowledge of Q (and without w).[29] Therefore, we can give C the polynomial Q and reduce the scenario to one where the only protocols running are non-malleable commitments (as required by the definition of non-malleability). Given this, we have that the probability that C will commit to a pair (Q', w') where $w' = w$ is essentially the same as the probability of it guessing the password w outright. (Recall that by the non-malleability of the commitment scheme, A's commitment including the password w does not help C in generating a commitment that includes the same w. This holds unless C simply copies A's commitment unmodified; however, then we can show that B rejects unless $\pi_A = \pi_B$, in which case key-match holds). We denote C's non-malleable commitment by (Q', w').

We first consider the probability that $w' = w$ (i.e., that the second element in the pair committed to by C equals the shared secret password of A and B), and $Q' \neq Q$.[30] Since A's commitment does not help C in generating this commitment, and since w is uniformly distributed in $\mathcal{D}$ with respect to C's view, the probability that C generates such a commitment (i.e., that $w' = w$) is at most negligibly more than ϵ. (Indeed, if C generates a commitment with $w' = w$, then it may cause B to accept, even when $\pi_A \neq \pi_B$.)

Next, if B receives a non-malleable commitment to (Q', w') where $w' \neq w$, then the validation stage ensures that B will reject. Essentially, this is because the (C, B) validation stage enforces that B's output from the polynomial evaluation be consistent with the non-malleable commitment that B received. That is, it ensures that B will reject unless he receives $Q'(w')$ from the (C, B)-polynomial evaluation (i.e., $\pi_B = Q'(w')$). On the other hand, the validation stage also enforces that the polynomial input by C into the (C, B)-polynomial evaluation is Q' (i.e., $Q_C = Q'$). Thus, the respective inputs of C and B into the (C, B)-polynomial evaluation are Q' and w, and so B receives $Q'(w)$ as the output of this evaluation (by the evaluation's correctness).[31] Therefore, if B accepts, it must be the case that $Q'(w') = Q'(w)$, which implies that $w' = w$ (because Q' is a non-constant linear polynomial). Letting bad denote the event in the lemma's claim, we get

$$\begin{aligned}
\Pr[\mathsf{bad}] &= \Pr[\mathsf{bad} \;\&\; w'{=}w] + \Pr[\mathsf{bad} \;\&\; w'{\neq}w] \\
&\leq \Pr[\mathsf{bad} \;\&\; w'{=}w \;\&\; Q'{=}Q] + \Pr[\text{Case 2} \;\&\; w'{=}w \;\&\; Q'{\neq}Q] + \Pr[\mathrm{dec}_B{=}\mathsf{acc} \;\&\; w'{\neq}w] \\
&\leq \mu(n) \;+\; (\epsilon + \mu(n)) \;+\; \mu(n)
\end{aligned}$$

Referring to the augmented definition of security, we note that in this synchronization case, the session-key challenge received by C in the augmented setting is of no consequence. This is because C completes its non-malleable commitment to B before A terminates, and so the value of its commitment is determined before C receives the session-key challenge. Therefore, C's success in

[29] This is the reason that an additional commitment to the polynomial Q is included in the pre-key exchange. That is, were we to augment the polynomial evaluation by having the parties include their transcripts of the non-malleable commitment to Q and w (rather than just by a commitment to Q), then knowledge of w would be required for running A's role in the pre-key exchange. The argument for non-malleability here would then fall.

[30] As we have mentioned, in case $(Q', w') = (Q, w)$, we can show that party B rejects with overwhelming probability, unless $\pi_A = \pi_B$. This is because the validation stage essentially enforces that $\pi_B = Q'(w')$, and therefore in case $(Q', w') = (Q, w)$ it follows that $\pi_B = Q'(w') = Q(w) = \pi_A$. Thus, in this proof sketch, we only consider the case that $(Q', w') \neq (Q, w)$.

[31] Correctness, even in the concurrent setting, is implied by security in the stand-alone setting, because the latter holds even when the adversary knows the private input of the honest party, which in turn allows it to emulate the concurrent execution.

generating a related commitment is independent of the session-key challenge. Furthermore, as is shown in the full proof, if C has failed in generating a related commitment, then B rejects with overwhelming probability *even* if C is later given both Q and w (and not merely the session-key $k_2(Q(w))$). ■

The full proof of Lemma 4.7, which amounts to a careful implementation of the above proof sketch, can be found in Section 6.2.

4.5 Simulating the Stand-Alone (A, C) Execution

In this section, we show that C's view, when interacting with A only, can be simulated by a machine that interacts with nobody.[32] Actually, we show that the *joint* distribution of C's simulated view along with the password and a random string is $(1 - O(\epsilon))$-indistinguishable from C's real view along with the password and A's output session-key.

Theorem 4.8 *For every ppt channel C' interacting with A only, there exists a* non-interactive *machine C'', such that*

$$\left\{w, k_2(Q(w)), \text{output}(C'^{A(Q,w)}(\sigma))\right\}_{n,\mathcal{D},\sigma} \stackrel{2\epsilon}{\equiv} \{w, U_n, \text{output}(C''(\sigma))\}_{n,\mathcal{D},\sigma}$$

where Q is a random non-constant linear polynomial, $\mathcal{D} \subseteq \{0,1\}^n$ is any ppt samplable dictionary, $w \in_R \mathcal{D}$, and $\epsilon = \frac{1}{|\mathcal{D}|}$.

Proof Sketch: First, notice that it is enough to prove that for every ppt channel C',

$$\left\{w, k_2(Q(w)), \text{output}(C'^{A(Q,w)}(\sigma))\right\} \stackrel{2\epsilon}{\equiv} \left\{w, U_n, \text{output}(C'^{A(Q,\tilde{w})}(\sigma))\right\} \tag{12}$$

where $w, \tilde{w} \in_R \mathcal{D}$ are independently chosen passwords from $\mathcal{D}$. In order to see that Eq. (12) implies Theorem 4.8, define the following non-interactive machine C''. Machine C'' chooses a random (linear, non-constant) polynomial Q and a "password" $\tilde{w} \in_R \mathcal{D}$. Then, C'' perfectly emulates an execution of $C'^{A(Q,\tilde{w})(\sigma)}$ by playing A's role (C'' can do this because it knows Q and $\tilde{w}$). Finally, C'' outputs whatever C' does. The resulting output of C'' is distributed exactly like $\mathsf{output}(C'^{A(Q,\tilde{w})}(\sigma))$. Thus, it is enough to prove Eq. (12). Now, notice that the distributions $\{w, U_n, \text{output}(C'^{A(Q,\tilde{w})}(\sigma))\}$ and $\{\tilde{w}, U_n, \text{output}(C'^{A(Q,w)}(\sigma))\}$ are *equivalent*. We therefore proceed by proving that

$$\left\{w, k_2(Q(w)), \text{output}(C'^{A(Q,w)}(\sigma))\right\} \stackrel{2\epsilon}{\equiv} \left\{\tilde{w}, U_n, \text{output}(C'^{A(Q,w)}(\sigma))\right\} \tag{13}$$

First we show that at the conclusion of the polynomial evaluation, with respect to C''s view, the pair $(w, Q(w))$ is $(1-\epsilon)$-indistinguishable from $(\tilde{w}, U_n)$. The fact that w is indistinguishable from $\tilde{w}$ follows from the fact that in the first two stages of the protocol, A uses w only in the non-malleable commitment. Thus, by the hiding property of the non-malleable commitment scheme, w remains indistinguishable from $\tilde{w}$. It is therefore enough to show that after the polynomial evaluation the value of $Q(w)$ is $(1 - \epsilon)$-pseudorandom, with respect to C''s view.

[32]Recall that in the *next* subsection, we show that the (C, B)-execution can be simulated by C itself, while interacting with A. Thus put together, these two simulations provide the core of the proof of security of the entire protocol (for active adversaries). Our choice of the current order of the two simulations is due to the fact that we use elements in the analysis of the current simulation in the analysis of the next simulation.

Consider what C' can learn about $Q(w)$ from the first two stages of the protocol (i.e., until the end of the polynomial evaluation). Due to the hiding property of the two commitment schemes in use, the two commitment transcripts reveal nothing of Q or w, and so the only place that C' can learn something is from the polynomial evaluation itself. The security of the polynomial evaluation implies that the receiver (here played by C') can learn nothing beyond the value of $Q(\cdot)$ at a single point selected by C'. We denote this point by w_C. Thus, in the case that $w_C \neq w$, given $Q(w_C)$, the values $Q(w)$ and U_n are statistically close (recall that Q is a random, non-constant, linear polynomial and so we have "almost" pairwise independence). However, since w is uniformly distributed in $\mathcal{D}$ (and C' learned nothing about it so far), the probability that $w_C = w$ is at most ϵ. This means that at the conclusion of the polynomial evaluation, with respect to C''s view, $Q(w)$ can be distinguished from U_n with probability at most negligibly greater than ϵ.

We have shown that after the first two stages, with respect to C''s view, $(w, Q(w))$ is $(1-\epsilon)$-indistinguishable from $(\tilde{w}, U_n)$. We now consider the messages sent by A in the remaining two stages. Recall that A sends nothing in the last (i.e., fourth) stage, whereas the only messages sent by A in the third stage are the value $y = f^{2n}(Q(w))$, messages it sends as prover in the zero-knowledge proof, and a MAC of the entire message transcript keyed by $k_1(Q(w))$. The zero-knowledge proof reveals nothing because it can be simulated by C' itself (in the standard manner, since here we are considering a stand-alone setting between A and C'). Thus, it remains to deal with y and the MAC-value. We do so by showing that, even when given the value $y = f^{2n}(Q(w))$, the MAC-key $k_1(Q(w))$, and the view of C' at the end of Stage 2, the value $(w, Q(w))$ is $(1-2\epsilon)$-indistinguishable from $(\tilde{w}, U_n)$. The latter implies Eq. (13), and is proven by relying on the following two facts: (1) as established above, $\{w, Q(w)\}$ is $(1-\epsilon)$-indistinguishable from $\{\tilde{w}, U_n\}$, with respect to C''s view at the end of Stage 2 of the protocol, and (2) the string $y = f^{2n}(Q(w))$ along with the MAC-key $k_1(Q(w))$ and the session-key $k_2(Q(w))$ constitutes a pseudorandom generator. This concludes the proof of the theorem. ■

The full proof of Theorem 4.8, which amounts to a careful implementation of the above proof sketch, can be found in Section 7.

We note that since in this theorem we consider a stand-alone execution between A and C', the analogous claim for the augmented definition of security holds as well. This is because the session-key challenge is only given to C' after the entire execution has terminated. Therefore, it is equivalent to giving the session-key to the distinguisher. However, the distinguisher receives the actual session-key anyway, and can thus generate the challenge by itself.

4.6 Simulating the (C, B) Execution

In this section, we show how the entire (C, B) execution can be simulated (by C while interacting with A). That is, we consider the concurrent setting in which C interacts with both A and B. We claim that a channel interacting *only with* A can simulate C's view in the concurrent setting with A and B, so that C's simulated view is $(1-O(\epsilon))$-indistinguishable from its view in an execution with A and B. In fact, $(1-O(\epsilon))$-indistinguishability holds also for C's view combined with the password and the session-key. That is,

Theorem 4.9 (simulating the (C, B) execution): *For every ppt channel C interacting with A and B, there exists a ppt channel C' interacting only with A, such that*

$$\left\{w, k_2(Q(w)), \mathrm{output}({C'}^{A(Q,w)}(\sigma))\right\}_{n,\mathcal{D},\sigma} \stackrel{5\epsilon}{\equiv} \left\{w, k_2(Q(w)), \mathrm{output}(C^{A(Q,w),B(w)}(\sigma))\right\}_{n,\mathcal{D},\sigma}$$

where Q is a random non-constant linear polynomial, $\mathcal{D} \subseteq \{0,1\}^n$ is any ppt samplable dictionary, $w \in_R \mathcal{D}$, and $\epsilon = \frac{1}{|\mathcal{D}|}$.

Proof Outline: The theorem is proved in two steps. Conceptually, simulation of the (C, B) execution is demonstrated by separately showing how the first three stages of the (C, B) execution (i.e., everything except for B's $\mathsf{accept/reject}$ bit) can be simulated, and then showing how B's $\mathsf{accept/reject}$ bit itself can also be simulated. In order to implement this two-step process, we consider a modified party $B_{\not\mathsf{dec}}$ that behaves exactly as B, except that it does not output an $\mathsf{accept/reject}$ bit. Theorem 4.9 is obtained by combining the following Lemmas 4.10 and 4.11, which refer to the first and second steps, respectively. □

Further details on the proof of Theorem 4.9: As outlined above, Theorem 4.9 is obtained by combining Lemmas 4.10 and 4.11, which are stated in the following Subsections 4.6.1 and 4.6.2, respectively. In these subsections we also provide sketches for the proofs of these lemmas. The full proofs, to be found in Section 8, are merely careful implementations of the corresponding proof sketches.

4.6.1 Step 1: Simulating the $(C, B_{\not\mathsf{dec}})$ execution

We start by showing that C's interaction with A and the modified B (i.e., $B_{\not\mathsf{dec}}$, which has no public accept/reject output), can be simulated by a machine that only interacts with A.

Lemma 4.10 *Let $\tilde{C}$ be a ppt channel interacting with A and a modified party $B_{\not\mathsf{dec}}$ who does not output an* $\mathsf{accept/reject}$ *bit. Then, there exists a ppt channel C' interacting with A* only*, such that*

$$\left\{w, k_2(Q(w)), \mathrm{output}(C'^{A(Q,w)})\right\} \stackrel{c}{\equiv} \left\{w, k_2(Q(w)), \mathrm{output}(\tilde{C}^{A(Q,w), B_{\not\mathsf{dec}}(w)})\right\}$$

Proof Sketch: Since $k_2(Q(w))$ is a polynomial-time function of $Q(w)$, it suffices to prove the following equation:

$$\left\{w, Q(w), \mathrm{output}(C'^{A(Q,w)})\right\} \stackrel{c}{\equiv} \left\{w, Q(w), \mathrm{output}(\tilde{C}^{A(Q,w), B_{\not\mathsf{dec}}(w)})\right\} \qquad (14)$$

First notice that the only messages sent by $B_{\not\mathsf{dec}}$ in the validation stage are as an honest verifier in the zero-knowledge proof. These can therefore be easily simulated. Next, observe that in the remaining first two stages, the only place that $B_{\not\mathsf{dec}}$ uses w is in the $(\tilde{C}, B_{\not\mathsf{dec}})$ polynomial-evaluation. However, by the definition of the polynomial evaluation functionality, $\tilde{C}$ receives no output from this evaluation and thus nothing is revealed about w. This is trivial in a stand-alone setting; here we claim that it also holds in our concurrent setting. Formally, we show that if $B_{\not\mathsf{dec}}$ were to use some fixed $w' \in \mathcal{D}$ instead of the password w, then this is indistinguishable to $\tilde{C}$ (when also interacting concurrently with A). That is, we show that for every ppt $\tilde{C}$,

$$\left\{w, Q(w), \mathrm{output}(\tilde{C}^{A(Q,w), B_{\not\mathsf{dec}}(w)})\right\} \stackrel{c}{\equiv} \left\{w, Q(w), \mathrm{output}(\tilde{C}^{A(Q,w), B_{\not\mathsf{dec}}(w')})\right\} \qquad (15)$$

where $w \in_R \mathcal{D}$ is a random password and $w' \in \mathcal{D}$ is fixed. This is shown by reducing $\tilde{C}$'s concurrent execution with A and $B_{\not\mathsf{dec}}$ to a stand-alone two-party setting between $\tilde{C}$ and $B_{\not\mathsf{dec}}$ only.[33] The

[33] Indeed, the reader may consider this reduction (i.e., removing the $(A, \tilde{C})$ execution) odd, given that our final goal here is to remove the $(\tilde{C}, B_{\not\mathsf{dec}})$ execution (i.e., reduce $\tilde{C}$'s concurrent execution with A and $B_{\not\mathsf{dec}}$ to a stand-alone two-party setting between $\tilde{C}$ and A). Still, such a reduction enables us to establish Eq. (15), and once Eq. (15) is established we proceed to remove the $(\tilde{C}, B_{\not\mathsf{dec}})$ execution.

reduction is obtained by providing $\tilde{C}$ with the password w and the polynomial Q, which enables $\tilde{C}$ to perfectly emulate the entire $(A, \tilde{C})$ execution. As a result of this emulation, we are left with a stand-alone setting between $\tilde{C}$ and $B_{\not{d}\mathsf{ec}}$ in which $B_{\not{d}\mathsf{ec}}$ inputs either w or w' into the polynomial evaluation (and $\tilde{C}$ knows both w and w'). In this stand-alone setting, the security of the polynomial evaluation guarantees that $\tilde{C}$ can distinguish the input cases with at most negligible probability, even when given both w and w' (as well as Q). Eq. (15) follows.

We have established that $\tilde{C}$ cannot distinguish the case that $B_{\not{d}\mathsf{ec}}$ uses w from the case that $B_{\not{d}\mathsf{ec}}$ uses w'. This suggests to define the channel C' as follows: C' chooses an arbitrary $w' \in \mathcal{D}$ and emulates the $\tilde{C}^{A(Q,w),B_{\not{d}\mathsf{ec}}(w')}$ setting for $\tilde{C}$, while interacting with $A(Q,w)$ (and using w' in the emulation of $B_{\not{d}\mathsf{ec}}(w')$). At the end of the interaction, C' outputs whatever $\tilde{C}$ does. Channel $\tilde{C}$'s view in this simulation is indistinguishable from in a real execution with A and $B_{\not{d}\mathsf{ec}}$, establishing Eq. (14) and therefore proving the lemma.

We note that the above argument is unchanged when considering the augmented definition of security. This is true because Eq. (15) holds even if $\tilde{C}$ is explicitly given both Q and w (in which case $\tilde{C}$ can generate the session-key challenge by itself). ■

4.6.2 Step 2: Simulating B's decision bit

We now show how the accept/reject bit of B can be simulated (while interacting with A and $B_{\not{d}\mathsf{ec}}$).

Lemma 4.11 *Let $B_{\not{d}\mathsf{ec}}$ be a party who does not output an* accept/reject *bit. Then, for every ppt channel C interacting with A and B, there exists a ppt channel $\tilde{C}$ interacting with A and $B_{\not{d}\mathsf{ec}}$, such that*

$$\left\{w, k_2(Q(w)), \text{output}(\tilde{C}^{A(Q,w),B_{\not{d}\mathsf{ec}}(w)})\right\} \stackrel{5\epsilon}{\equiv} \left\{w, k_2(Q(w)), \text{output}(C^{A(Q,w),B(w)})\right\}$$

Proof Sketch: The proof of this claim relies heavily on the security of the MAC-value sent in the validation stage of the protocol. Recall that A sends a MAC of her entire session-transcript using $k_1(\pi_A) = k_1(Q(w))$ as the key. Furthermore, B verifies the MAC value that it receives using the key $k_1(\pi_B)$, where π_B is B's output from the polynomial evaluation. Intuitively, the MAC ensures that, except with probability $O(\epsilon)$, if C was not reliable, then B will detect its interference and will therefore reject. On the other hand, if C was reliable then B will surely accept. Loosely speaking, this means that C can learn "at most an $O(\epsilon)$ fraction of a bit of information" from B's accept/reject bit.

We begin by proving the security of the MAC value when keyed by $k_1(\pi_A)$. This is an important step in proving Lemma 4.11. We note that we need to show that the MAC is secure only *before* B outputs its accept/reject bit. Thus, we consider a scenario in which C interacts with A and the modified party $B_{\not{d}\mathsf{ec}}$. The security of the MAC is formally stated in the following claim. (For simplicity, we consider an implementation of a MAC by a pseudorandom function. However, our proof can be extended to any secure implementation of a MAC.)

Claim 4.12 *Let C be an arbitrary ppt channel interacting with A and a modified party $B_{\not{d}\mathsf{ec}}$ as in Lemma 4.11, and let t_A and t_B denote the transcripts from the (A, C) and $(C, B_{\not{d}\mathsf{ec}})$ executions, respectively. Then, if $t_A \neq t_B$, it holds that the value $MAC_{k_1(\pi_A)}(t_B)$ is $(1-2\epsilon)$-pseudorandom with respect to C's view. That is,*

$$\left\{MAC_{k_1(\pi_A)}(t_B), C^{A(Q,w),B_{\not{d}\mathsf{ec}}(w)}\right\} \stackrel{2\epsilon}{\equiv} \left\{U_n, C^{A(Q,w),B_{\not{d}\mathsf{ec}}(w)}\right\}$$

Proof Sketch: We first observe that we can ignore the entire $(C, B_{\not{\mathsf{d}}\mathsf{ec}})$ execution in proving the claim. A similar claim has already been shown in Lemma 4.10 (above). Loosely speaking, Eq. (14) in the proof of Lemma 4.10 states that C's view is essentially the same when interacting with A and $B_{\not{\mathsf{d}}\mathsf{ec}}$ or when interacting with A alone. Actually, Eq. (14) asserts that these two views are indistinguishable also when considered in conjunction with $(w, Q(w))$. Since $k_1(\cdot)$ is a polynomial-time function, it follows that these two views are indistinguishable also when considered in conjunction with $(w, k_1(Q(w))$, where $k_1(Q(w))$ is the MAC-key.

We now analyze the security of the MAC-key in a stand-alone setting between A and C. This proof is very similar to the proof of Theorem 4.8 (there $k_2(Q(w))$ is shown to be $(1 - O(\epsilon))$-pseudorandom; here a similar result is needed with respect to $k_1(Q(w))$). As in Theorem 4.8, we first establish that at the conclusion of the polynomial evaluation, the value $Q(w)$ is $(1 - O(\epsilon))$-pseudorandom to C. Next, recall that the only messages sent by A in the third stage of the protocol are $y = f^{2n}(Q(w))$, messages from a zero-knowledge proof and a MAC of the message transcript. The zero-knowledge proof can be simulated and so it reveals nothing. Then, since $G(s) = (f^{2n}(s), k_1(s))$ is a pseudorandom generator and $Q(w)$ is $(1 - O(\epsilon))$-pseudorandom at the end of Stage 2, it holds that the MAC-key $k_1(Q(w))$ remains $(1 - O(\epsilon))$-pseudorandom even *given* $y = f^{2n}(Q(w))$.

Having established that the MAC-key is $(1 - O(\epsilon))$-pseudorandom (with respect to C's view), we conclude by showing that this implies that the probability that C successfully forges the MAC is at most $O(\epsilon) + \mu(n)$. Now, since $k_1(Q(w))$ is $(1 - O(\epsilon))$-pseudorandom, a pseudorandom function keyed by $k_1(Q(w))$ is also $(1 - O(\epsilon))$-pseudorandom. Recall that the last message sent by A is $MAC_{k_1(Q(w))}(t_A)$ where t_A is A's message transcript. Therefore, by the properties of a $(1 - O(\epsilon))$-pseudorandom function, for every $t \neq t_A$ that C can produce, the value $MAC_{k_1(Q(w))}(t)$ is $(1 - O(\epsilon))$-pseudorandom given C's view. Since t_B is part of C's view, it follows that the value $MAC_{k_1(Q(w))}(t_B)$ is $(1 - O(\epsilon))$-pseudorandom given C's view. This concludes the proof of the claim.

We note that the above also holds for the augmented definition of security. This is because the MAC-key $k_1(Q(w))$ remains $(1 - O(\epsilon))$-pseudorandom even given *both* $y = f^{2n}(Q(w))$ and the session-key $k_2(Q(w))$. Therefore, even if the session-key challenge equals $k_2(Q(w))$, this cannot help C generate a correct MAC. Given that this is the case, the rest of the proof also follows for the augmented definition. ■

We now use Claim 4.12 and Theorem 4.5 (the key-match property) to show that the probability that B accepts in executions for which C is not reliable is at most $O(\epsilon)$. (Recall that C is reliable in a *particular execution* if it acts like a passive (eavesdropping) adversary in that execution.)

Proposition 4.13 *For every ppt channel C,*

$$\Pr[\mathrm{dec}_B = \mathsf{acc}\ \&\ \mathrm{reliable}_C = \mathsf{false}] < 5\epsilon + \mu(n)$$

Proof Sketch: We show this proposition by combining the following facts:

- Theorem 4.5 states that the probability that B accepts and $\pi_A \neq \pi_B$ is at most negligibly greater than 3ϵ.
- Let t_A and t_B be the (A, C) and (C, B) message-transcripts, respectively. Then, Claim 4.12 states that if $t_A \neq t_B$, then $MAC_{k_1(\pi_A)}(t_B)$ is $(1 - 2\epsilon)$-pseudorandom with respect to C's view.
- B only accepts if he receives $MAC_{k_1(\pi_B)}(t_B)$ (i.e., a MAC value keyed by $k_1(\pi_B)$) in the last step of the protocol.

Now, consider the case that C is not reliable and thus by definition $t_A \neq t_B$. Then, if $\pi_A = \pi_B$ we have that, by the security of the MAC, party B accepts with probability at most 2ϵ. On the other hand, if $\pi_A \neq \pi_B$, then by the key-match property, party B accepts with probability at most 3ϵ (irrespective of the MAC). Therefore, the probability that B accepts and $t_A \neq t_B$ is at most negligibly greater than 5ϵ. ■

Given Proposition 4.13, we can complete the proof of Lemma 4.11. First, we describe the adversary $\tilde{C}$ (who interacts with A and $B_{\not{d}ec}$). Channel $\tilde{C}$ emulates an execution of $C^{A(Q,w),B(w)}$, while interacting with A and $B_{\not{d}ec}$. This emulation is "easy" for $\tilde{C}$, except for the accept/reject decision bit of B (since this is the only difference between its execution with A and $B_{\not{d}ec}$, and an execution with A and B). Therefore, at the conclusion of the $(C, B_{\not{d}ec})$ execution, $\tilde{C}$ attempts to guess B's accept/reject decision-bit (which is not given to $\tilde{C}$ but which C does expect to see) and outputs whatever C does. Channel $\tilde{C}$'s guess for B's decision-bit is according to the natural rule (suggested by the above discussion): B accepts if and only if C was reliable. We stress that $\tilde{C}$ can easily determine whether or not C was reliable (in the current execution). To establish the approximate-correctness of the above rule, observe that, on one hand, if C is reliable then B always accepts (and so, in this case, $\tilde{C}$'s guess is always correct). On the other hand, if C was not reliable, then B accepts with probability at most $5\epsilon + \mu(n)$. Therefore, $\tilde{C}$ is wrong in its guess with probability at most $5\epsilon + \mu(n)$, and the difference in C's view in the case that it really receives B's output bit and the case it receives $\tilde{C}$'s guess, is at most negligibly greater than 5ϵ. ■

4.7 The Security of Protocol 3.2 for Arbitrary Adversaries

The fact that Protocol 3.2 satisfies Definition 2.4 (i.e., Theorem 3.3) follows by combining the passive adversary case (i.e., Theorem 4.1) and the active adversary case (i.e., Theorem 4.14, below).

Theorem 4.14 (active executions): *Protocol 3.2 satisfies Condition 2 in Definition 2.4. That is, for every ppt real-model channel C, there exists a ppt ideal-model channel $\hat{C}$, such that*

$$\left\{\mathrm{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)\right\}_{n,\mathcal{D},\sigma} \stackrel{12\epsilon}{\equiv} \left\{\mathrm{REAL}_{C}(\mathcal{D},\sigma)\right\}_{n,\mathcal{D},\sigma}$$

where $\mathcal{D} \subseteq \{0,1\}^n$ is any ppt samplable dictionary, $\sigma \in \{0,1\}^{\mathrm{poly}(n)}$ is the auxiliary input for the adversary, and $\epsilon = \frac{1}{|\mathcal{D}|}$.

In this section, we present a full proof of Theorem 4.14. Our main tools are the simulations provided by Theorems 4.8 and 4.9 (presented in Sections 4.5 and 4.6, respectively). In addition, we make an essential use of Proposition 4.13 (of Section 4.6), and a marginal use (i.e., in order to save an $O(\epsilon)$ term) of Proposition 4.3 (of Section 4.3).

Proof: We begin by describing the ideal-model channel $\hat{C}$. Adversary $\hat{C}$ is derived from the transformations of Theorems 4.8 and 4.9. That is, combining these theorems together, we have that for every ppt real-model channel, there exists a *non-interactive* machine C'' such that

$$\left\{w, U_n, \mathrm{output}(C''(\sigma))\right\} \stackrel{7\epsilon}{\equiv} \left\{w, k_2(Q(w)), \mathrm{output}(C^{A(Q,w),B(w)}(\sigma))\right\} \tag{16}$$

Next, we define the ideal-model channel $\hat{C}$ as follows: $\hat{C}$ first invokes the non-interactive machine C'' guaranteed by Eq. (16). When $\hat{C}$ receives the output of C'' (which contains C's view and in particular B's accept/reject bit), $\hat{C}$ sets the value of b (the bit sent by it to the trusted party) as follows:

- If B accepted in the view output by C'', then $\hat{C}$ sends $b = 1$ to the trusted party.
- If B rejected in this view, then $\hat{C}$ sends $b = 0$ to the trusted party.

(Recall that upon receiving $b = 1$, the trusted party hands the same uniformly distributed key to A and B. On the other hand, upon receiving $b = 0$, the trusted party hands a uniformly distributed key to A, and B receives $\perp$.) Finally, $\hat{C}$ halts and outputs the output of C''.

Before proceeding, we remark on an important property regarding the simulated view of C that is output by C''. In this view, B accepts if and only if C was reliable. This is because this is the way that B's accept/reject bit is simulated by $\tilde{C}$, as described at the end of the proof of Lemma 4.11. We will use this property below.

We now show that the combined input/output distributions in the real and ideal models are at most negligibly greater than 12ϵ apart. By Eq. (16) and the definition of $\hat{C}$, we have that

$$\left\{w, U_n, \text{output}(\hat{C})\right\} \stackrel{7\epsilon}{\equiv} \left\{w, k_2(Q(w)), \text{output}(C^{A(Q,w),B(w)})\right\} \tag{17}$$

This seems very close to proving the theorem (the first distribution is "almost" the real-model distribution and the second distribution is "almost" the ideal-model distribution), where in both cases the only thing missing is B's local output (which may or may not equal A's local output). included). It remains to show that the distributions are still $(1-O(\epsilon))$-indistinguishable even when B's output is included. Loosely speaking, this is shown by separately considering the cases that C acts reliably and unreliably. When C is reliable, then the IDEAL and REAL distributions are computationally indistinguishable (by Proposition 4.3). On the other hand, when C is not reliable, then B rejects with probability at least $1 - 5\epsilon$, in which case B's output is defined as $\perp$.

Formally, let D be any ppt distinguisher who attempts to distinguish between the IDEAL and REAL distributions. We separately analyze the distance between the distributions when B accepts and when B rejects. When referring to B's decision (i.e., dec_B), within the context of $\text{IDEAL}_{\hat{C}}$, we mean B's decision as included in the emulated view of C (which is part of the output of $\hat{C}$). (Note that by the construction of $\hat{C}$, it holds that B's decision in the emulated view matches the output of B in the ideal-model; i.e., $\text{dec}_B = \mathsf{rej}$ iff the output of B in the ideal-model is $\perp$.) We begin with the case that B rejects:

$$\begin{aligned}
&\left|\Pr[D(\text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)) = 1 \ \&\ \text{dec}_B = \mathsf{rej}] - \Pr[D(\text{REAL}_C(\mathcal{D},\sigma)) = 1 \ \&\ \text{dec}_B = \mathsf{rej}]\right| \\
&= \Big|\Pr_w\left[D\left(w, U_n, \perp, \text{output}(\hat{C})\right) = 1 \ \&\ \text{dec}_B = \mathsf{rej}\right] \\
&\quad - \Pr_{Q,w}\left[D\left(w, k_2(Q(w)), \perp, \text{output}(C^{A(Q,w),B(w)})\right) = 1 \ \&\ \text{dec}_B = \mathsf{rej}\right]\Big|
\end{aligned}$$

The above follows from the protocol definition that states that when B rejects it outputs $\perp$, and from the construction of the ideal-model adversary $\hat{C}$ who sends $b = 0$ to the trusted party (causing B's output to be $\perp$) in the case that B rejects in the view output by C''. Noting that C's view includes B's accept/reject decision bit (and thus implicitly B's output of $\perp$ in the case that B rejects), by Eq. (17) we have that

$$\begin{aligned}
&\left|\Pr[D(\text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)) = 1 \ \&\ \text{dec}_B = \mathsf{rej}]\right. \\
&\quad \left. -\Pr[D(\text{REAL}_C(\mathcal{D},\sigma)) = 1 \ \&\ \text{dec}_B = \mathsf{rej}]\right| < 7\epsilon + \mu(n)
\end{aligned} \tag{18}$$

We now analyze the case that B accepts. Here, we further break down the events and separately consider the case that C is reliable and C is not reliable. (Recall that in the ideal distribution, the event of C being reliable or not refers to its behavior as implicit in the view output by C'' for $\hat{C}$.) Starting with the subcase in which C is reliable, we have:

$$\begin{aligned}
&\big|\Pr[D(\text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)) = 1 \ \&\ \text{dec}_B = \mathsf{acc} \ \&\ \text{reliable}_C = \mathsf{true}] \\
&\quad - \Pr[D(\text{REAL}_C(\mathcal{D},\sigma)) = 1 \ \&\ \text{dec}_B = \mathsf{acc} \ \&\ \text{reliable}_C = \mathsf{true}]\big| \\
= \ &\Big|\Pr_w\Big[D\left(w, U_n, U_n, \text{output}(\hat{C})\right) = 1 \ \&\ \text{dec}_B = \mathsf{acc} \ \&\ \text{reliable}_C = \mathsf{true}\Big] \\
&\quad - \Pr_{Q,w}\Big[D\left(w, k_2(Q(w)), k_2(Q(w)), \text{output}(C^{A(Q,w),B(w)})\right) = 1 \ \&\ \text{dec}_B = \mathsf{acc} \ \&\ \text{reliable}_C = \mathsf{true}\Big]\Big|
\end{aligned}$$

Noting that when C is reliable, B always accepts (and so its real-model and ideal-model outputs are always $k_2(Q(w))$ and U_n respectively), we have that the above difference equals

$$\big|\Pr[D(\text{REAL}_C(\mathcal{D},\sigma)) = 1 \ \&\ \text{reliable}_C = \mathsf{true}] - \Pr[D(\text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)) = 1 \ \&\ \text{reliable}_C = \mathsf{true}]\big|$$

By Proposition 4.3 this difference is at most negligible. We now consider the case in which B accepts and C is not reliable.

$$\begin{aligned}
&\big|\Pr[D(\text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)) = 1 \ \&\ \text{dec}_B = \mathsf{acc} \ \&\ \text{reliable}_C = \mathsf{false}] \\
&\quad - \Pr[D(\text{REAL}_C(\mathcal{D},\sigma)) = 1 \ \&\ \text{dec}_B = \mathsf{acc} \ \&\ \text{reliable}_C = \mathsf{false}]\big|
\end{aligned}$$

By Proposition 4.13 we have that in a real execution, $\Pr[\text{dec}_B = \mathsf{acc} \ \&\ \text{reliable}_C = \mathsf{false}] < 5\epsilon + \mu(n)$. In contrast, in an ideal execution, $\Pr[\text{dec}_B = \mathsf{acc} \ \&\ \text{reliable}_C = \mathsf{false}] = 0$. This is due to the fact that in the simulated transcript output by C'', party B accepts if and only if C was reliable (as mentioned above). Therefore, we have that the above difference is at most negligibly greater than 5ϵ. Putting these together, we have that

$$\begin{aligned}
&\big|\Pr[D(\text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)) = 1 \ \&\ \text{dec}_B = \mathsf{acc}] \\
&\quad - \Pr[D(\text{REAL}_C(\mathcal{D},\sigma)) = 1 \ \&\ \text{dec}_B = \mathsf{acc}]\big| < 5\epsilon + \mu(n)
\end{aligned} \tag{19}$$

Combining Equations (18) and (19) we conclude that,

$$\big|\Pr[D(\text{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)) = 1] - \Pr[D(\text{REAL}_C(\mathcal{D},\sigma)) = 1]\big| < 12\epsilon + \mu(n)$$

and the theorem follows. ■

5 Full Proof of Security for Passive Adversaries

In this section, we present the proof of Lemma 4.2, used for proving the "passive adversaries" requirement of Definition 2.4. Recall that this lemma relates to a *passive* channel C who can only eavesdrop on protocol executions between *honest* parties A and B. This means that C receives the transcript of messages sent by A and B and tries to "learn something" based on this transcript alone.

Lemma 5.1 (Lemma 4.2 – restated): *For every* passive *ppt channel C,*

$$\left\{w, k_2(Q(w)), \text{output}(C^{A(Q,w),B(w)})\right\} \stackrel{\text{c}}{\equiv} \left\{w, U_n, \text{output}(C^{A(Q,\tilde{w}),B(\tilde{w})})\right\}$$

where Q is a random non-constant linear polynomial, and w and $\tilde{w}$ are independently and uniformly distributed in $\mathcal{D}$.

Proof: As we have mentioned, since C is passive, it merely receives a message transcript of a two-party protocol. We stress that there are no concurrent adversarial executions in this case, but rather merely a transcript of a standard stand-alone protocol execution between two honest parties. The issue is merely what can a third party (i.e., C) learn from such a transcript. We answer this question by relying on the (stand-alone) security of the different modules in our protocol. We start by presenting notation for transcripts of executions of our protocol.

The message-transcript of an execution of our protocol is a function of the inputs Q and w, and the respective random coins of A and B, denoted r_A and r_B. We denote the message transcript of the *first two stages* of the protocol by $t_2(Q, w, r_A, r_B)$. Furthermore, we denote by $T_2(Q, w)$ a random variable that assumes values of $t_2(Q, w, r_A, r_B)$ where r_A and r_B are randomly chosen. (Note that the security parameter n, and thus the lengths of Q,w,r_A and r_B are implicit in all these notations.)

We begin by showing that the probability ensemble $\{T_2(Q_1, w_1)\}_{Q_1,Q_2,w_1,w_2}$ is computationally indistinguishable from $\{T_2(Q_2, w_2)\}_{Q_1,Q_2,w_1,w_2}$.[34] This is proved in the following claim, which is then used to establish the lemma (which refers to the *entire* protocol execution, rather than just to the first two stages as shown in the claim).

Claim 5.2 *The probability ensemble $\{T_2(Q_1, w_1)\}_{Q_1,Q_2,w_1,w_2}$ is computationally indistinguishable from $\{T_2(Q_2, w_2)\}_{Q_1,Q_2,w_1,w_2}$. That is, for every ppt distinguisher D, every polynomial $p(\cdot)$ and all sufficiently large pairs (Q_1, w_1) and (Q_2, w_2),*

$$|\Pr[D(Q_1, Q_2, w_1, w_2, t_2(Q_1, w_1, r_A, r_B)) = 1] - \Pr[D(Q_1, Q_2, w_1, w_2, t_2(Q_2, w_2, r_A, r_B)) = 1]| < \frac{1}{p(n)}$$

where r_A and r_B are uniformly chosen strings.

Proof: The proof is based on the security of the different modules in the protocol.

The Commitments: Due to the hiding property of string commitments, a non-malleable commitment to (Q_1, w_1) is indistinguishable from one to (Q_2, w_2), and likewise an ordinary commitment to Q_1 is indistinguishable from one to Q_2.

The Polynomial Evaluation: The inputs to the polynomial evaluation are Q, w and Commit(Q). Denote by $T_P(Q, w)$, a random variable assuming transcripts for this evaluation (when the parties A and B use uniformly distributed random tapes). We claim that for all sufficiently large values of Q_1, Q_2, w_1, w_2, we have that $\{T_P(Q_1, w_1)\}$ and $\{T_P(Q_2, w_2)\}$ are indistinguishable. This can be derived from the following two facts (and is based on the security of the polynomial evaluation that implies that A learns nothing and that B learns only $Q(w)$):

1. For all sufficiently large non-constant, linear polynomials Q, passwords $w \in \mathcal{D}$ and strings $x \in \{0,1\}^n$, we have that

$$\{T_P(Q, w)\}_{Q,w,x} \stackrel{c}{\equiv} \{T_P(Q, x)\}_{Q,w,x} \tag{20}$$

   This is based directly on the fact that A learns nothing of B's input (which is either w or x) from the evaluation. Therefore, given her message transcript, A must not be able to distinguish the case that B used w from the case that B used x. Eq. (20) follows.

[34] Notice that it is not true that the ensembles $\{T(Q_1, w_1)\}_{Q_1,Q_2,w_1,w_2}$ and $\{T(Q_2, w_2)\}_{Q_1,Q_2,w_1,w_2}$ are indistinguishable, where $T(Q, w)$ is a random variable assuming message transcripts for the *entire* protocol (including the validation stage). This is because the string $y = f^{2n}(Q(w))$ is sent during the validation stage. Thus, given (Q_1, w_1), a distinguisher may compare $f^{2n}(Q_1(w_1))$ to the y-value of the transcript, and determine whether or not the transcript is based on (Q_1, w_1).

2. For all sufficiently large non-constant, linear polynomials Q_1, Q_2 and strings $x \in \{0,1\}^n$ such that $Q_1(x) = Q_2(x)$, it holds that

$$\{T_P(Q_1, x)\}_{Q_1,Q_2,x} \stackrel{c}{\equiv} \{T_P(Q_2, x)\}_{Q_1,Q_2,x} \tag{21}$$

This is because B obtains only $Q(x)$ from the evaluation, where A inputs $Q \in \{Q_1, Q_2\}$. Since $Q_1(x) = Q_2(x)$, party B cannot distinguish the case that A inputs Q_1 or Q_2 into the evaluation (otherwise he learns more than just $Q(x)$). Eq. (21) follows.

Now, for every two non-constant polynomials Q_1 and Q_2, there exists a third polynomial Q_3 and values x_1 and x_2 such that $Q_1(x_1) = Q_3(x_1)$ and $Q_2(x_2) = Q_3(x_2)$. Therefore, we have that for all sufficiently large non-constant linear polynomials Q_1, Q_2 and passwords $w_1, w_2 \in \mathcal{D}$

$$\begin{aligned}
\{T_P(Q_1, w_1)\} &\stackrel{c}{\equiv} \{T_P(Q_1, x_1)\} \\
&\stackrel{c}{\equiv} \{T_P(Q_3, x_1)\} \\
&\stackrel{c}{\equiv} \{T_P(Q_3, x_2)\} \\
&\stackrel{c}{\equiv} \{T_P(Q_2, x_2)\} \\
&\stackrel{c}{\equiv} \{T_P(Q_2, w_2)\}
\end{aligned}$$

where Q_3, x_1, x_2 are as described above (the ensembles in the above equations are indexed by $(Q_1, Q_2, Q_3, w_1, w_2, x_1, x_2)$). The first, third and fifth "$\stackrel{c}{\equiv}$" are due to Eq. (20) and the second and fourth is from Eq. (21). We therefore have that $\{T_P(Q_1, w_1)\} \stackrel{c}{\equiv} \{T_P(Q_2, w_2)\}$. Combining this with what we have shown regarding the commitments, the claim follows. ■

Loosely speaking, the above claim shows that the transcript of the first two stages of the protocol reveals nothing significant about the polynomial or password used in the execution. Recalling that no messages are sent in the last (fourth) stage, it remains to analyze the additional messages sent in the third stage of the protocol. Recall that the third stage (validation) consists of A sending $y = f^{2n}(Q(w))$, a zero-knowledge proof, and a MAC of the session-transcript keyed by $k_1(Q(w))$. To simplify the exposition, we will assume that A sends the MAC-key itself, rather than the MAC-value (which can be computed by C from the MAC-key and the visible session-transcript). Intuitively, the zero-knowledge proof reveals nothing, and the session-key $k_2(Q(w))$ remains pseudorandom even given $f^{2n}(Q(w))$ and $k_1(Q(w))$ because $G(Q(w)) \stackrel{\text{def}}{=} (f^{2n}(Q(w)), k_1(Q(w)), k_2(Q(w)))$ constitutes a pseudorandom generator. Furthermore, the password w is "masked" by Q, and therefore remains secret, even given $Q(w)$ itself. Details follow.

By the definition of zero-knowledge, there exists a simulator that generates proof-transcripts indistinguishable from real proofs. Thus, we may ignore this part of the validation stage for the rest of the proof (because, using the simulator, C may generate this part by itself). Thus, we may assume that the entire session-transcript consists of $T_2(Q, w)$ along with the pair $\left(f^{2n}(Q(w)), k_1(Q(w))\right)$. From here on, we denote by $\{T_2(Q, w), Q, w\}_{n,\mathcal{D}}$ (and variations of this), a probability ensemble where Q and w are chosen randomly and then the transcript T_2 is generated based on this Q and w. (Previously, the ensembles considered here were indexed by Q and w; from here on, the ensembles are indexed by n and $\mathcal{D}$.) Thus, in order to complete the proof of the lemma, it remains to show that for randomly chosen values Q_1, Q_2, w_1, w_2,

$$\begin{aligned}
&\{T_2(Q_1, w_1), f^{2n}(Q_1(w_1)), k_1(Q_1(w_1)), k_2(Q_1(w_1)), w_1\}_{n,\mathcal{D}} \\
&\quad \stackrel{c}{\equiv} \{T_2(Q_2, w_2), f^{2n}(Q_2(w_2)), k_1(Q_2(w_2)), U_n, w_1\}_{n,\mathcal{D}}
\end{aligned} \tag{22}$$

Now, using Claim 5.2 and the fact that for a random Q_1, the value $Q_1(w_1)$ is uniformly distributed in $\{0,1\}^n$ (for *every* w_1), we have

$$\{T_2(Q_1, w_1), Q_1(w_1), w_1\}_{n,\mathcal{D}} \stackrel{\rm c}{\equiv} \{T_2(Q_2, w_2), Q_1(w_1), w_1\}_{n,\mathcal{D}} \stackrel{\rm c}{\equiv} \{T_2(Q_2, w_2), U_n, w_1\}_{n,\mathcal{D}} \tag{23}$$

(Notice that Q_1 is independent of $T_2(Q_2, w_2)$.) This then implies that

$$\begin{aligned}
&\{T_2(Q_1, w_1), f^{2n}(Q_1(w_1)), k_1(Q_1(w_1)), k_2(Q_1(w_1)), w_1\}_{n,\mathcal{D}} \\
&\stackrel{\rm c}{\equiv} \{T_2(Q_2, w_2), f^{2n}(U_n), k_1(U_n), k_2(U_n), w_1\}_{n,\mathcal{D}} \\
&\stackrel{\rm c}{\equiv} \{T_2(Q_2, w_2), f^{2n}(U_n^{(1)}), k_1(U_n^{(1)}), U_n^{(2)}, w_1\}_{n,\mathcal{D}}
\end{aligned} \tag{24}$$

where the last "$\stackrel{\rm c}{\equiv}$" is by pseudorandomness of the generator $G(s) = \big(f^{2n}(s), k_1(s), k_2(s)\big)$, and $U_n^{(1)}$ and $U_n^{(2)}$ denote independent uniform distributions over n-bit strings. Using Eq. (23), we have

$$\{T_2(Q_2, w_2), U_n\}_{n,\mathcal{D}} \stackrel{\rm c}{\equiv} \{T_2(Q_1, w_1), Q_1(w_1)\}_{n,\mathcal{D}} \equiv \{T_2(Q_2, w_2), Q_2(w_2)\}_{n,\mathcal{D}}$$

and since $w_1 \in_R \mathcal{D}$ independently of (Q_2, w_2), it holds that

$$\{T_2(Q_2, w_2), U_n^{(1)}, w_1\}_{n,\mathcal{D}} \stackrel{\rm c}{\equiv} \{T_2(Q_2, w_2), Q_2(w_2), w_1\}_{n,\mathcal{D}}$$

This, in turn, implies that

$$\{T_2(Q_2, w_2), f^{2n}(U_n^{(1)}), k_1(U_n^{(1)}), U_n^{(2)}, w_1\}_{n,\mathcal{D}} \stackrel{\rm c}{\equiv} \{T_2(Q_2, w_2), f^{2n}(Q_2(w_2)), k_1(Q_2(w_2)), U_n^{(2)}, w_1\}_{n,\mathcal{D}} \tag{25}$$

Combining Equations (24) and (25), we obtain Eq. (22) completing the proof of the lemma (by replacing w_1 with w and w_2 with $\tilde{w}$ in Eq. (22), this is the same as the lemma statement). ■

6 Full Proof of the Key-Match Property

The key-match property captured in Theorem 4.5 states that the probability that A and B both accept, yet have different pre-keys (i.e., $\pi_A \neq \pi_B$) is at most $O(\epsilon)$. Recall that $\pi_A \stackrel{\rm def}{=} Q(w)$ and that π_B is B's output from the polynomial evaluation. We prove this theorem by considering two complementary schedulings of the concurrent executions. We show that for each scheduling, the probability that B accepts and $\pi_A \neq \pi_B$ is at most $O(\epsilon)$. (In fact, in the first scheduling, B accepts with probability at most $O(\epsilon)$, irrespective of whether or not $\pi_A = \pi_B$.)

6.1 Proof of Lemma 4.6 (The Unsynchronized Case)

The proof of Lemma 4.6 will involve considering a variety of different settings. Specifically, we will consider the probability that B accepts when interacting with C, which in turn interacts with a pair of machines that are not necessarily A and B. For sake of clarity, we introduce the notation $\mathrm{dec}(C^{A',B'})$ that means the decision of B' (which is public and known to C) when interacting with C that interacts concurrently also with A'.

Lemma 6.1 (Lemma 4.6 – restated; Case 1 – unsynchronized): *Let C be a ppt channel and define* Case 1 *to be a scheduling of the protocol execution by which C completes the polynomial evaluation with A* before *concluding the non-malleable commitment with B. Then, for every polynomial $p(\cdot)$ and all sufficiently large n's*

$$\Pr[\mathrm{dec}(C^{A,B}) = \mathrm{acc} \ \&\ \text{Case 1}] < 2\epsilon + \frac{1}{p(n)}$$

Proof: The proof of this lemma is the most complex proof in this paper. It proceeds by reducing the concurrent setting to a two-party stand-alone setting. However, before performing this reduction, we "remove" the zero-knowledge proofs from the protocol. This is done in two steps: a *small step* in which the zero-knowledge proof in which B plays the verifier is removed (from the (C, B) interaction), and a *big step* in which the zero-knowledge proof in which A plays the prover is removed (from the (A, C) interaction).

We start with the small step. We consider a modified party, denoted B', that accepts or rejects based solely on the y-value received in the validation stage. That is, B' does not play the verifier in the zero-knowledge proof given by C, and also ignores the MAC sent by C. Since we only omitted checks that may make B reject, we have that

$$\Pr[\mathrm{dec}(C^{A,B}) = \mathrm{acc} \ \&\ \text{Case 1}] \leq \Pr[\mathrm{dec}(C^{A,B'}) = \mathrm{acc} \ \&\ \text{Case 1}] \tag{26}$$

The proof of the lemma proceeds by showing that the r.h.s is upper-bounded by $2\epsilon + \mu(n)$. We stress that (by considering B' rather than B) we have removed the zero-knowledge proof given by C to B, but the zero-knowledge proof given by A to C still remains. The next subsection is devoted to getting rid of the latter proof, which is the big step (mentioned above). Once this is achieved, we turn (in Subsection 6.1.2) to analyzing the residual protocol, by reducing the analysis of its execution in the concurrent three-part setting to an analysis of an auxiliary two-party protocol in the standard stand-alone setting.

6.1.1 Simulating A's zero-knowledge proof

We begin by showing that when C interacts with A and B', the zero-knowledge proof given by A to C can be simulated. Since the proof (given by A to C) is zero-knowledge, it seems that the channel C (who plays the verifier in the proof) should be able to simulate it itself. This is true (by definition) if the zero-knowledge proof is executed as stand-alone. However, the definitions of zero-knowledge guarantee nothing in our setting, where the proof is run concurrently with other related protocols (belonging to the (C, B')-execution). Technically speaking, the zero-knowledge simulation of A typically requires rewinding C. However, messages belonging to the (C, B')-execution may be interleaved with the proof. For example, C's queries to A in the proof may depend on messages received from B'. Rewinding C would thus also require rewinding B'. However, since B' is an external party, he *cannot* be rewound.

We remark that concurrent zero-knowledge does not solve this problem either, since it relates to concurrent executions of a (zero-knowledge) protocol with itself, and not concurrently with arbitrary protocols. Still, we use the ideas underlying the concurrent zero-knowledge proof system of Richardson and Kilian [47] in order to address the problem that arises in our application.

We refer the reader to Appendix A.4 for a description of the Richardson and Kilian (RK) proof system. Recall that we set the parameter m (the number of iterations in the first part of the RK proof) to equal $r + t(n)$, where r is the total number of rounds in the first two stages of our protocol, and $t(n)$ is any non-constant function of the security parameter n (e.g., $t(n) = \log \log n$).

We now motivate how the proof simulation is done in our scenario, where C interacts with A and B'. In such a case (when B' rather than B is involved), the total number of rounds in the (C, B') execution equals $r = m - t$ (since B' does not participate in the zero-knowledge proof given by C in the validation stage). On the other hand, the number of iterations in the first part of the RK-proof given by A to C equals m. Therefore there are t complete iterations in the first part of this proof in which C receives no messages from B'. In these iterations it is possible to rewind C without rewinding B'. This is enough to establish zero-knowledge, since the Richardson-Kilian construction is such that as soon as rewinding is possible in one iteration, the entire proof may be simulated. The crucial point is that we rewind C at a place that does not require the rewinding of B' (which is not possible, since B' is an outside party). With this motivation in mind, we move to our actual proofs.

The modification of A into $A_{\not{z}k}$. In our above description, when we say that A's proof can be simulated by C itself, this means that A can be modified to a party $A_{\not{z}k}$, whose protocol definition does not include providing a zero-knowledge proof in the validation stage. Before continuing, we formally define what we mean by this modification of A to $A_{\not{z}k}$. This needs to be done carefully because the transcript (and not just the result) of the zero-knowledge proof affects other parts of our protocol. Specifically, in the validation stage, A sends a MAC of her entire message-transcript to C. This message-transcript includes also the messages of the zero-knowledge proof. Therefore, the protocol of $A_{\not{z}k}$ must be appropriately redefined to take this issue into account.

In the zero-knowledge proof with C, party A plays the prover. The essence of the modification of A to $A_{\not{z}k}$ is in replacing A's actions as prover in the (A, C)-proof by C simulating the resulting messages by itself. This modification works only if C's view in the protocol execution with $A_{\not{z}k}$ is indistinguishable from its view in an execution with A. As mentioned, the MAC sent by A in the validation stage refers to the entire message transcript, including messages from the zero-knowledge proof. Therefore, the MAC value sent by $A_{\not{z}k}$ must also include messages from the simulated proof. However, $A_{\not{z}k}$ does not see these messages as the simulation is internal in C; therefore the message transcript of the proof must be explicitly given to her.

In light of this discussion, we define the modified $A_{\not{z}k}$ to be exactly the same as A, except that she does not provide a zero-knowledge proof (in her validation stage). Instead, at the point in which A's zero-knowledge proof takes place, she receives a string s that she appends to her message transcript. This means that the only difference between A and $A_{\not{z}k}$'s message transcripts is that A's transcript includes messages from a zero-knowledge proof and $A_{\not{z}k}$'s transcript includes s instead. Intuitively, if s is the transcript of the simulated proof, then A and $A_{\not{z}k}$'s message transcripts are indistinguishable. This ensures that the MACs sent by A and $A_{\not{z}k}$, respectively, are indistinguishable.

The simulation. We now show that for every channel C interacting with A and B', there exists a channel C' interacting with $A_{\not{z}k}$ and B' such that the channels' views in the two cases are indistinguishable. Since B's accept/reject bit is part of C's view (which is included in C's output), it follows that the probability that B' accepts (in an execution with C' and $A_{\not{z}k}$) is negligibly close to the probability that B' accepts (in an execution with C and A). This enables us to continue proving Lemma 6.1 by considering the setting where C interacts with $A_{\not{z}k}$ and B' (rather than with A and B').

Lemma 6.2 *Let $A_{\not{z}k}$ and B' be as above. Then, for every ppt channel C there exists a ppt channel C' such that,*

$$\left\{\mathrm{output}(C'^{A_{\not{z}k}(Q,w),B'(w)})\right\} \stackrel{c}{\equiv} \left\{\mathrm{output}(C^{A(Q,w),B'(w)})\right\}$$

Proof: Our proof is based on the intuitive simulation strategy described above. In order to formally see why this works, we first recall the RK-proof system (or actually a simplification of it which suffices for our purposes). This proof system (for NP-statements) consists of two parts. The first part consists of m iterations, where in iteration i the verifier (who is played by C in our case) sends the prover a commitment to a random string, denoted v_i. The prover then sends a commitment to a random string, denoted p_i, and the verifier decommits. The commitments used are perfectly binding, and so given the commitment we can refer to the unique value committed to by it. (Indeed, we shall make extensive use of this fact.) In the second part of the proof, the prover proves (using a witness-indistinguishable proof [22]) that either there exists an i such that $p_i = v_i$ or that the original NP-statement (i.e., the one on which the proof system is invoked) is correct. In a real proof, the prover will not be able to set $p_i = v_i$, except with negligible probability, which implies that the proof system is sound (i.e., false statements can be proved only with negligible probability). On the other hand, if there is one iteration of the first part in which the simulator can rewind the verifier, then it can set $p_i = v_i$ (because it rewinds after obtaining the decommitment value v_i and can thus set its commitment p_i to equal v_i). In this case, it can successfully execute the witness-indistinguishable proof (by using this $p_i = v_i$ and without knowing a proof of the original statement).

Now, in our case there are t iterations in which no messages are sent to B'. In these iterations it is possible to rewind C. The only problem remaining is that C may refuse to decommit (or decommit improperly, which is effectively the same). If during the execution of a real proof, C refuses to decommit, then the prover halts. During the simulation, however, we must ensure that the probability that we halt due to C's refusal to decommit is negligibly close to this probability in a real execution. This prevents us from simply halting if, after a rewind, C refuses to decommit (since this may skew the probability).

Before we continue, we define the concepts of *promising* and *successful* iterations, which are used in describing our simulation strategy. Loosely speaking, a promising iteration is one that enables the simulator to rewind C (i.e., C properly decommits before sending any message to B'), with the *hope* of obtaining a successful simulation. (Recall that once C has been rewound, the simulator can send a commitment to the value p_i satisfying $p_i = v_i$.) However, even if C can be rewound at some point, a successful simulation is not necessarily obtained. This is because after rewinding, it is possible that C refuses to decommit (or sends a message to B'). Thus, a successful iteration is one in which, *after* C receives a commitment to p_i such that $p_i = v_i$, it (i.e., C) properly decommits (before sending any messages to B'). That is:

- An iteration i is called promising if when C receives a commitment generated according to the protocol instructions (i.e., a commitment to a random p_i), the iteration is such that no messages are sent to B' and C decommits properly. (This refers to the situation before any rewinding of iteration i.)
- An iteration i is called successful if when C receives a commitment to p_i such that $p_i = v_i$, the iteration is such that no messages are sent to B' and C decommits properly. (This typically occurs after rewinding when p_i can be set to v_i.)

Restating the above motivating discussion using this terminology, we have that when any iteration is successful, we can complete a full simulation of the proof. This is because the first part of the proof is such that there exists an i for which $p_i = v_i$. Therefore the simulator (having an adequate NP-witness) can execute the necessary witness-indistinguishable proof. Another important point is that the probability that an iteration is successful is very close to the probability that it is promising (by the hiding property of the commitment used on p_i). Finally, we note that unless there exists an iteration in which C refuses to decommit when it receives a commitment to a random p_i, there must be at least t promising iterations. Retrying to rewind each promising iteration polynomially many times yields that with overwhelming probability, at least one of these rewinding tries is successful, allowing us to complete the simulation.

The Actual Simulator: We now show how C' runs the simulation for C. The channel C' plays the prover to C; in each iteration i it receives a commitment to v_i from C and replies with a commitment to a random string p_i. If an iteration is not promising, then there are two possible reasons why: (1) C refused to decommit – in this case C' halts the simulation (successfully); (2) C sent a message to B' during the iteration – in this case C' simply continues to the next iteration. We call this (first) execution of the ith iteration the *initial execution*, and call the subsequent executions of the ith iteration *rewinding attempts*. Note that rewinding attempts for iteration i take place only if the initial execution of iteration i is promising.

If iteration i is promising, then C' obtains the decommitted value v_i, rewinds C and commits to $p_i = v_i$. That is, C' attempts to obtain a successful iteration. If the rewound iteration is successful, then (as we have argued) C' can complete the entire simulation successfully. However, the iteration may not be successful after the rewinding. That is, C may refuse to decommit or may send messages to B'. As long as the rewound iteration is not successful, C' continues to rewind up to N times (where $N = O(n^2)$). If none of the rewinds were successful then C' resends its original commitment to a random p_i (i.e., the very same commitment sent in the initial execution), and continues to the next iteration. We stress that each rewinding attempt is independent of the others in the sense that C' sends an independent random commitment to $p_i = v_i$ each time.

It is crucial that during a rewinding attempt C' blocks any message sent by C to B'. This is because C cannot be rewound beyond a point in which it sent a message to B' (because B' is an outside party and its message receipt event cannot be rewound). Furthermore, since C may refuse to decommit, further rewindings (or a replay of the initial execution) may be necessary. Thus, in case that C sends a message to B' during a rewinding attempt, C' halts the attempt (without forwarding the message), and rewinds again (up to N times).

The Output of the Simulator: We will show below that, with overwhelmingly high probability, either the initial executions of all iterations are non-promising or one of the rewinding attempts succeeds. In both cases C' completes the simulation, and outputs a transcript of a (simulated) proof. We claim that this transcript is indistinguishable from transcripts of real executions of the RK-proof.

Consider first the simulation of the first part of the RK-proof. For each iteration, consider the initial execution of this iteration by the simulator, and note that this execution is distributed identically to the real execution. In case the initial execution is non-promising the simulator just appends it to the simulation transcript (and truncates the simulation if the verifier has decommitted improperly). Thus, this case is identical to the real execution. If, on the other hand, the initial execution is promising then the simulator tries to rewind it. If none of the rewinding attempts succeeds then the simulator appends the initial execution to the simulation transcript, which again means that the appended part is distributed identically to the real execution. On the other hand, if one of the rewinds is successful then the simulator appends its (i.e., the rewinding's) transcript to the simulation transcript. By the hiding property of the commitment scheme, the appended part is computationally indistinguishable from the corresponding part in the real execution (although these distributions are statistically far apart). We conclude that the simulation of the first part of the RK-proof is computationally indistinguishable from the first part of a real RK-proof.

Assuming that the simulator has succeeded in generating a successful rewinding, it has obtained an NP-witness to the claim that $p_i = v_i$. Playing the role of the prover while using this witness, allows the simulator to produce a transcript of the second part of the RK-proof. By the witness-indistinguishability of the proof system used in the second part, it follows that the simulated transcript is computationally indistinguishable from the real one. (Actually, we rely on the fact that the latter proof system has a strong witness-indistinguishability property; that is, if two claims are computationally indistinguishable then so are the real proof transcripts regardless of which witness is used by the prover [26, Sec. 4.6].) Thus, it remains to show that the probability that the simulator fails to generate a successful rewind (in case some iteration is promising) is negligible.

Analysis of the simulator's failure probability: Recall that the simulator fails only if it has completed a non-truncated simulation of the first part of the RK-proof without generating any successful rewinding. Note that for this to happen, each of the simulated iterations must include a proper decommitment, or else the simulation terminates successfully while outputting a truncated transcript (as the prover would do in a real proof). Since there at least t iterations for which C does not send any messages to B' (recall that there are m iterations and only $m-t$ messages are sent from C to B'), it follows that a non-truncated transcript must contain at least t promising iterations. The simulation fails only if all N rewinding attempts for these promising iterations are not successful; we show that for an adequate choice of N (the number of rewindings of a promising iteration), this occurs with at most negligible probability.

The above statement is easy to establish in case the identities of the promising iterations are fixed. If iteration i is always promising then a corresponding rewinding attempt must be successful with overwhelming probability (or else a contradiction to the hiding property of the commitment is reached). What makes the analysis more complicated is that the identities of the promising iterations may be random variables (which may even depend on the transcript of previous iterations).

Our aim is to show that the simulation fails with negligible probability. That is, for every positive polynomial p, we show that (for all but finitely many n's) the simulation fails with probability smaller than $1/p(n)$. In the rest of the analysis we assume that $m < \sqrt{n}$ (this is easy to enforce, possibly by artificially increasing the original security parameter n to a polynomial in n). We use the following notation:

- Let $X_1, \ldots, X_m$ be random variables such that $X_i = 1$ if and only if C sends no messages to B' during the initial execution of iteration i (i.e., when a random commitment to a random p_i is sent, before any rewinding of iteration i).

- Let $Y_1, \ldots, Y_m$ be random variables such that $Y_i = 1$ if and only if C correctly decommits during the initial execution of iteration i.

Thus, an iteration i is *promising* if and only if $X_i = Y_i = 1$.

We now introduce similar notations for rewinding attempts of iterations.

- Let $X'_1, \ldots, X'_m$ be random variables such that $X'_i = 1$ if and only if C sends no messages to B' during a single rewinding attempt for iteration i, when a random commitment to $p_i = v_i$ is sent.

- Let $Y'_1, \ldots, Y'_m$ be random variables such that $Y'_i = 1$ if and only if C correctly decommits during a single rewinding attempt for iteration i, when a random commitment to $p_i = v_i$ is sent.

Thus, a given rewinding attempt for iteration i is *successful* if and only if $X'_i = Y'_i = 1$.

We note that some of the above random variables may be undefined, in which case we just define them arbitrarily. Specifically, the random variables of iteration i are not defined (above) if the simulation halted in some iteration $j < i$ (which happens if and only if $Y_j = 0$).

We start by showing that the success event $X'_i = Y'_i = 1$ occurs essentially as often as the promising event $X_i = Y_i = 1$. We wish to establish this not only for the a-priori probabilities but also when conditioned on any past event that occurs with noticeable probability. Specifically, we prove the following.

Claim 6.3 *For every polynomial q, every $i \leq m$, and every $\alpha \in \{0,1\}^{i-1}$ either*

$$\Pr[Y_1 \cdots Y_{i-1} = 1^{i-1} \,\&\, X_1 \cdots X_{i-1} = \alpha] < \frac{1}{q(n)} \tag{27}$$

or

$$\begin{array}{ll} \text{if} & \Pr[X_i = Y_i = 1 \,|\, Y_1 \cdots Y_{i-1} = 1^{i-1} \,\&\, X_1 \cdots X_{i-1} = \alpha] \geq \frac{1}{n} \\ \text{then} & \Pr[X'_i = Y'_i = 1 \,|\, Y_1 \cdots Y_{i-1} = 1^{i-1} \,\&\, X_1 \cdots X_{i-1} = \alpha] > \frac{1}{2n} \end{array} \tag{28}$$

Proof: The claim follows by the hiding property of the commitment scheme. Specifically, an algorithm violating the hiding property is derived by emulating the first $i-1$ iterations (of the real execution) with the hope that $Y_1 \cdots Y_{i-1} = 1^{i-1} \,\&\, X_1 \cdots X_{i-1} = \alpha$ holds, which indeed occurs with noticeable probability. Given that this event occurs, the algorithm can distinguish a commitment to a random value from a commitment to a given v_i. More precisely, contradiction to the hiding property is derived by presenting two algorithms. The first algorithm emulates the real interaction for $i-1$ iterations and checks if the event $Y_1 \cdots Y_{i-1} = 1^{i-1} \,\&\, X_1 \cdots X_{i-1} = \alpha$ occurred. If yes, then it emulates the i^{th} iteration. If it holds that $X_i = Y_i = 1$, then the algorithm obtains v_i from the verifier decommitment. This string v_i is then output by the first algorithm as the challenge string for the commitments. Then, the second algorithm is given the view of the first algorithm along with a challenge commitment which is either a commitment to v_i or to a random p_i. This algorithm then re-emulates the i^{th} iteration using the challenge commitment, and outputs 1 if and only if C correctly decommits and sends no messages to B'. The point is that if the algorithm received a commitment to a random p_i, then it outputs 1 with probability exactly $\Pr[X_i = Y_i = 1 \,|\, Y_1 \cdots Y_{i-1} = 1^{i-1} \,\&\, X_1 \cdots X_{i-1} = \alpha]$. In contrast, if it received a commitment to v_i, then it outputs 1 with probability exactly $\Pr[X'_i = Y'_i = 1 \,|\, Y_1 \cdots Y_{i-1} = 1^{i-1} \,\&\, X_1 \cdots X_{i-1} = \alpha]$. Thus, if the claim does not hold, then the algorithm distinguishes commitments with non-negligible probability. ■

Using Claim 6.3, we show that the simulation fails with at most negligible probability. That is,

Claim 6.4 *Let* `fail` *denote the event in which the simulation fails. Then, for every polynomial $p(\cdot)$ and all sufficiently large n's*

$$\Pr[\texttt{fail}] < \frac{1}{p(n)}$$

Proof: Our aim is to upper bound the probability that the simulation fails, by considering all possible values that $X = X_1 \cdots X_m$ can obtain in such a case. We have:

$$\begin{aligned} \Pr[\texttt{fail}] &= \sum_{\beta \in \{0,1\}^m} \Pr[\texttt{fail} \,\&\, X = \beta] \\ &= \sum_{\alpha \in S} \Pr[\texttt{fail} \,\&\, X_1 \cdots X_{|\alpha|} = \alpha] \end{aligned} \tag{29}$$

where S is any *maximal prefix-free* subset of $U \stackrel{\text{def}}{=} \cup_{i=1}^{m} \{0,1\}^i$, and Eq. (29) is justified below. Recall that a set S is *prefix-free* if for every $\alpha, \beta \in S$ it holds that α is not a prefix of β. By *maximality* we mean that adding any string in U to S violates the prefix-free condition. It follows that every $\alpha \in \{0,1\}^m$ has a (unique) prefix in S. To justify Eq. (29) observe that the strings in $\{0,1\}^m$ can be partitioned into subsets such that all the strings in each subset have a unique prefix in the set S, and so we can consider events corresponding to these prefixes rather than events that correspond to all possible m-bit long strings.

For a constant $k < t$ to be determined later, we define H_k to be the set of all strings having length at most $m-1$ and hamming weight exactly k (note that k will eventually be set to $1 + 2\lim_{n\to\infty}\log_n p(n)$). Let $S_1 \stackrel{\rm def}{=} \{\alpha'1 : \alpha' \in H_k\}$ (i.e., strings of length at most m and hamming weight $k+1$ that have no strict prefix satisfying this condition), and S_2 be the set of all m-bit long strings having hamming weight at most k. Observe that $S_1 \cup S_2$ is a maximal prefix-free subset of $\{0,1\}^m$. (Prefix-freeness holds because all strings in S_1 have hamming weight $k+1$ and so cannot be prefixes of string in S_2, nor can any m-bit string be a prefix of another m-bit string.) Applying Eq. (29) we have:

$$\begin{aligned}
\Pr[\texttt{fail}] &= \sum_{\alpha\in S_1\cup S_2} \Pr[\texttt{fail}\,\&\,X_1\cdots X_{|\alpha|}=\alpha] \\
&= \sum_{\alpha'\in H_k} \Pr[\texttt{fail}\,\&\,X_1\cdots X_{|\alpha'|+1}=\alpha'1]
\end{aligned}$$

where the last equality follows because $S_1 = \{\alpha'1 : \alpha' \in H_k\}$ and $\Pr[\texttt{fail}\,\&\,X \in S_2] = 0$, where the latter fact is justified as follows. Recall that the simulator may fail only if C properly decommits in all the first $m-1$ iterations, which implies that all X_i's are properly defined (i.e., reflect what actually happens the these iterations, rather than when fictitiously defined in an arbitrary manner). This implies that there must be at least $t \geq k+1$ iterations/indices i such that $X_i = 1$ holds (i.e., no message was sent to B'), and so $X \notin S_2$. Now, using $|H_k| < m^{k+1}$, we have

$$\begin{aligned}
\Pr[\texttt{fail}] &< m^{k+1}\cdot \max_{\alpha'\in H_k}\{\Pr[\texttt{fail}\,\&\,X_1\cdots X_{|\alpha'|+1}=\alpha'1]\} \\
&\leq m^{k+1}\cdot \max_{\alpha'\in H_k}\{\Pr[\texttt{fail}\,\&\,X_1\cdots X_{|\alpha'|}=\alpha']\}
\end{aligned}$$

We will show that, for every $\alpha' \in H_k$, it holds that

$$\Pr[\texttt{fail}\,\&\,X_1\cdots X_{|\alpha'|}=\alpha'] < \frac{1}{m^{k+1}\cdot p(n)} \tag{30}$$

which establishes our claim that the simulation fails with probability smaller than $1/p(n)$. In order to establish Eq. (30), we fix an arbitrary $\alpha' \in H_k$, let $i = |\alpha'|+1$, and we consider two cases:

Case 1: $\Pr[Y_1\cdots Y_{i-1} = 1^{i-1}\,\&\,X_1\cdots X_{i-1}=\alpha'] < \frac{1}{m^{k+1}\cdot p(n)}$. In this case, using the fact that the simulation never fails if any of the Y_j's equals 0 (i.e., the fail event implies that all the Y_j's equal 1), it follows that $\Pr[\texttt{fail}\,\&\,X_1\cdots X_{i-1}=\alpha'] < \frac{1}{m^{k+1}\cdot p(n)}$ as desired.

Case 2: $\Pr[Y_1\cdots Y_{i-1} = 1^{i-1}\,\&\,X_1\cdots X_{i-1}=\alpha'] \geq \frac{1}{m^{k+1}\cdot p(n)}$. In this case, setting $q(n) = m^{k+1}\cdot p(n)$, we conclude that Eq. (28) holds. Furthermore, for every $j \leq i$, it holds that $\Pr[Y_1\cdots Y_{j-1} = 1^{j-1}\,\&\,X_1\cdots X_{j-1}=\alpha''] \geq \frac{1}{m^{k+1}\cdot p(n)}$ holds, where α'' is the $(j-1)$-bit long prefix of α'. Thus, Eq. (28) holds for α'' too. We are particularly interested in prefixes α'' such that $\alpha''1$ is a prefix of α'. We know that there are k such prefixes $\alpha''1$ and we denote the set of their lengths by J (i.e., $j \in J$ if the j-bit long prefix of α' ends with a one). We consider two subcases:

1. If for some $j \in J$, it holds that $\Pr[X_j = Y_j = 1\,|\,Y_1\cdots Y_{j-1} = 1^{j-1}\,\&\,X_1\cdots X_{j-1}=\alpha''] \geq \frac{1}{n}$ then (by Eq. (28)) it holds that $\Pr[X'_j = Y'_j = 1\,|\,Y_1\cdots Y_{j-1} = 1^{j-1}\,\&\,X_1\cdots X_{j-1}=\alpha''] > \frac{1}{2n}$. This means that a rewinding attempt at iteration j succeeds with probability

greater than $1/2n$, and the probability that we fail in $O(n^2)$ attempts is exponentially vanishing. Thus, in this subcase $\Pr[\texttt{fail} \,\&\, X_1 \cdots X_{i-1} = \alpha'] < 2^{-n} < \frac{1}{m^{k+1} \cdot p(n)}$ as desired.

2. The other subcase is that for every $j \in J$, it holds that $\Pr[X_j = Y_j = 1 \,|\, Y_1 \cdots Y_{j-1} = 1^{j-1} \,\&\, X_1 \cdots X_{j-1} = \alpha''] < \frac{1}{n}$. Recalling that failure may occur only if all Y_j's equal one, and letting $\alpha' = \sigma_1 \cdots \sigma_{i-1}$, we get (using $\sigma_j = 1$ for $j \in J$)

$$\begin{aligned} &\Pr[\texttt{fail} \,\&\, X_1 \cdots X_{i-1} = \alpha'] \\ &\quad \leq \Pr[Y_1 \cdots Y_{i-1} = 1^{i-1} \,\&\, X_1 \cdots X_{i-1} = \alpha'] \\ &\quad = \prod_{j=1}^{i-1} \Pr[X_j = \sigma_j \,\&\, Y_j = 1 \,|\, Y_1 \cdots Y_{j-1} = 1^{j-1} \,\&\, X_1 \cdots X_{j-1} = \sigma_1 \cdots \sigma_{j-1}] \\ &\quad \leq \prod_{j \in J} \Pr[X_j = 1 \,\&\, Y_j = 1 \,|\, Y_1 \cdots Y_{j-1} = 1^{j-1} \,\&\, X_1 \cdots X_{j-1} = \sigma_1 \cdots \sigma_{j-1}] \\ &\quad < (1/n)^k \end{aligned}$$

By a suitable choice of k (e.g., $k = 1 + 2\lim_{n\to\infty} \log_n p(n)$) and recalling that $m < \sqrt{n}$, we have $\frac{1}{n^k} = \frac{1}{\sqrt{n}^{k+1}} \cdot \frac{1}{n^{\frac{k-1}{2}}} < \frac{1}{m^{k+1} \cdot p(n)}$ as desired.

Thus, we have established the desired bound of Eq. (30) in all possible cases. The claim follows. ■

Completing the (A, C) Simulation: So far we have focused on the simulation of the RK-proof (concurrently to interacting with B'), but actually our goal is to simulate (A, C) by $(A_{\not k}, C')$, while interacting concurrently with B'. Clearly, whatever happens in the (A, C) execution before the RK-proof is emulated trivially by the $(A_{\not k}, C')$ execution (which is identical at this stage). The issue is what happens *after* the (simulated) RK-proof. Recall that by the construction of $A_{\not k}$, party $A_{\not k}$ expects to receive a string s in place of the zero-knowledge proof. This string is then concatenated to $A_{\not k}$'s session-transcript before she (applies the MAC and) sends the MAC value. In order to ensure that C's view of the protocol in this simulation is indistinguishable from in a real execution (where A proves the zero-knowledge proof), the channel C' must ensure that C receives a MAC value that is indistinguishable from the MAC value that it would have received from A. Channel C' does this by defining s to be the transcript of the zero-knowledge simulation. This means that the resulting session-transcript of $A_{\not k}$ is identical to the transcript held by C. Furthermore, this transcript is indistinguishable from a transcript that C would hold after a real execution with A (rather than in this simulated interaction). This implies that the MAC value sent by $A_{\not k}$ is indistinguishable from one that A would have sent (because the transcripts are indistinguishable to C even given the MAC-key $k_1(Q(w))$). This completes the proof of Lemma 6.2. ■

Combining Eq. (26) and Lemma 6.2 (while noting that both the scheduling case and B's decision are visible by the channel), we get

Corollary 6.5 *For every ppt C there exists a ppt $\hat{C}$ such that*

$$\Pr[\mathrm{dec}(C^{A,B}) = \mathrm{acc} \;\&\; \mathrm{Case}\ 1] < \Pr[\mathrm{dec}(\hat{C}^{A_{\not k},B'}) = \mathrm{acc} \;\&\; \mathrm{Case}\ 1] + \mu(n)$$

A note on the number of rounds: Our simulator works given that the number of iterations in the first part of the RK-proof is greater than the total number of the rest of the rounds in the protocol by any non-constant function of the security parameter n (say $\log\log n$). We note that a single additional round actually suffices. This can be shown using the techniques of [28], but would further complicate the analysis.

6.1.2 Proof of a modified lemma 4.6 (when C interacts with $A_{\not k}$ and B')

In view of Corollary 6.5, we now proceed to show that when C interacts with $A_{\not k}$ and B', the probability that B' accepts in the synchronization of Case 1 is at most negligibly greater than 2ϵ. That is, we proved

Lemma 6.6 *For every ppt C*

$$\Pr[\text{dec}(C^{A_{\not k},B'}) = \text{acc} \ \& \ \text{Case 1}] < 2\epsilon + \mu(n)$$

Proof: Our first step is to reduce the concurrent setting to a two-party stand-alone setting. The key point in this reduction is in noticing that according to the scheduling of Case 1, the two polynomial evaluations are run *sequentially* without any overlap. Specifically, the $(A_{\not k}, C)$-evaluation terminates before the (C, B')-evaluation begins. As a warm-up, consider a simplified setting in which the entire $(A_{\not k}, C)$-protocol consists only of a single polynomial evaluation; likewise for the (C, B')-protocol. Then, when the scheduling is as mentioned, a party P, can execute two sequential polynomial evaluations with C; in the first P plays $A_{\not k}$'s role and in the second P plays B''s role. That is, when this scheduling occurs the above two-party setting perfectly simulates the concurrent setting.

The actual reduction is, however, more complex since the $(A_{\not k}, C)$ and (C, B') protocols involve other steps beyond the polynomial evaluation. The protocol that we define between P and C must correctly simulate these other steps as well. As we shall see, some of the additional steps can be internally simulated by C, and some are emulated by an interaction of C with P. Specifically, apart from playing in both polynomial evaluations, P plays $A_{\not k}$'s role in the $(A_{\not k}, C)$-commitment stage and B''s role in the (C, B')-validation stage. What remains is B''s role in the (C, B')-commitment stage and $A_{\not k}$'s role in the $(A_{\not k}, C)$-validation stage; these are internally simulated by C. Table 1 shows which party (P or C) simulates $A_{\not k}$ and B''s respective roles. Note that when we say that C plays a role, this means internal simulation of the corresponding stage by C (who plays both parties); whereas when we say that P plays a role this means that the corresponding stage is emulated by C interacting with P (who plays the other party). (For example, in the (C, B') commitment, C plays B''s role and this therefore means that the entire (C, B') commitment is internally run by C.) Table 1 also shows the information that is "needed" by the parties for the simulation of each stage.

	Roles	
Stage	$A_{\not k}$	B'
1. Commitment	run by P; needs (Q, w)	run by C; needs nothing
2. Pre-Key Exchange	run by P; needs Q	run by P; needs w
3. Validation	run by C; needs $Q(w)$	run by P; needs nothing

Table 1: The assignment of "simulation roles" to P and C.

In order to play the corresponding roles, both parties get suitable inputs. Specifically, Party P is given the input (Q, w), which enables it to play the roles of $A_{\not\,k}$ and B' (in any stage). Party C is given $Q(w)$ as an auxiliary input (which, as we show, enables it to internally simulate the remaining parts of the execution). Thus, we actually prove that Lemma 6.6 holds even when C gets $Q(w)$ as an auxiliary input.

The following protocol makes sense whenever the scheduling of Case 1 occurs (in the emulated execution of $C^{A_{\not\,k}(Q,w),B'(w)}$). We will show that for every channel C, the (two-session concurrent) execution of $C^{A_{\not\,k}(Q,w),B'(w)}$ in the scheduling of Case 1 is "simulated" (in some adequate sense) by an adversary C' to a single-session execution of the following (mental experiment) protocol (where C' may also internally emulate additional steps).

Protocol 6.7 (mental experiment protocol (P, C):)

Inputs:

- *P has (Q, w), where Q is a linear (non-constant) polynomial and $w \in \mathcal{D}$.*
- *C receives the string $Q(w)$, where (Q, w) is the input of P.*

Operation:

1. Emulation of Stage 1 of the $(A_{\not\,k}, C)$-execution (commitment stage):
   - *P sends C a non-malleable commitment to (Q, w).*
2. Emulation of Stage 2 of the $(A_{\not\,k}, C)$-execution (pre-key exchange):
   - *P sends C a commitment $c_1 = \mathrm{Commit}(Q) = C(Q, r_1)$ for a random r_1.*
   - *P and C invoke an augmented polynomial evaluation, where P inputs the polynomial Q and (c_1, r_1) and C inputs c_1 and some value w_C (of its choice). Party C then receives the output value $Q(w_C)$ (or $\perp$ in the case of incorrect inputs).*
3. Emulation of Stage 2 of the (C, B')-execution (pre-key exchange):
   - *C sends P a commitment $c_2 = C(Q_C, r_2)$, for some polynomial Q_C and r_2 (of its choice).*
   - *C and P invoke another augmented polynomial evaluation (in the other direction), where C inputs the polynomial Q_C and (c_2, r_2) and P inputs c_2 and w. Party P receives π, which equals either $Q_C(w)$ or $\perp$, from the evaluation.*
4. Emulation of Stage 3 of the (C, B')-execution (validation stage):
   - *C sends a string y to P, and P outputs* accept *if and only if $y = f^{2n}(\pi)$.*

We say that C succeeds if P outputs accept at the conclusion of the protocol execution. We now show that any C succeeding in having B' accept in the concurrent protocol with the scheduling of Case 1, can be used by a party C' to succeed with the same probability in the above protocol with P.

Claim 6.8 *Let C be a ppt channel interacting with $A_{\not\,k}$ and B'. Then there exists a ppt party C' interacting with P in Protocol-(P, C') such that*

$$\Pr_{Q,w}[P(Q, w) \text{ accepts when interacting with } C'(Q(w))] = \Pr[\mathrm{dec}(C^{A_{\not\,k},B'}) = \mathrm{acc} \ \& \ \text{Case } 1]$$

Recall that $C^{A_{\neq k},B'}$ is actually a shorthand for $C^{A_{\neq k}(Q,w),B'(w)}$, where (Q,w) are random as in the l.h.s above.

Proof: The party C' incorporates C internally and *perfectly* simulates the concurrent setting with $A_{\not k}$ and B' for C (i.e., $C^{A_{\not k}(Q,w),B'(w)}$). First notice that Step (4) of the (P,C') protocol constitutes the full validation stage of the (C,B')-protocol (recall that the validation stage for B' consists only of checking that $y = f^{2n}(\pi_B)$). This means that the (P,C') protocol contains all stages of the $(A_{\not k},C)$ and (C,B') protocols, *except* for the first stage of the (C,B')-protocol and the third stage of the $(A_{\not k},C)$-protocol. As mentioned above, these stages are internally simulated by C'.

The C' simulation: We now describe how C' runs the simulation. Party C' invokes C and emulates the $C^{A_{\not k}(Q,w),B'(w)}$ setting for it, while interacting with P. This involves separately simulating the $(A_{\not k},C)$ and (C,B') executions. We note that the values Q_C and w_C from the (P,C) protocol are only implicitly defined in the simulation by C'. This simulation is carried out as follows (recall that C fully controls the scheduling):

- *The $(A_{\not k},C)$ Execution:*
  1. *Stages 1 and 2:* All messages from these stages of the execution are passed between C and P (without any change). That is, C' forwards any messages sent from C (to $A_{\not k}$) to P and likewise, messages from P are forwarded to C.
  2. *Stage 3:* C' internally emulates $A_{\not k}$'s role here, and thus P is not involved at all. In this stage C expects to receive the string $y = f^{2n}(Q(w))$ and a MAC of the $(A_{\not k},C)$ session-transcript keyed by $k_1(Q(w))$. Party C' can determine and send these messages since it has $Q(w)$ as input, and can thus compute both the y-string and the MAC-key (and so the MAC value).
- *The (C,B') Execution:*
  1. *Stage 1:* C' internally emulates B''s role here, and thus P is not involved at all. Recall that B''s role in this stage is as the receiver of a non-malleable commitment; therefore no secret information is needed by C' to emulate this part (by using C).
  2. *Stages 2 and 3:* When C sends the first message belonging to Stage 2 of the (C,B')-execution, party C' acts as follows:
     - *Scheduling Violation Case:* If this first message was sent **before** the completion of Stage 2 of the $(A_{\not k},C)$ execution (i.e., the scheduling of Case 1 does not hold), then C' halts (the simulation fails).
     - *Scheduling Conforming Case:* If this first message was sent **after** the completion of Stage 2 of the $(A_{\not k},C)$ execution (i.e., the scheduling conforms with Case 1), then C' continues the simulation by forwarding this and all consequent messages belonging to these stages to P (and returning messages from P to C).

This completes the simulation. Note that, when the simulation succeeds, C's view is identical to a real execution with $A_{\not k}$ and B'. Recall that the (P,C)-protocol emulates Stages 1 and 2 of the $(A_{\not k},C)$ protocol *before* Stages 2 and 3 of the (C,B') protocol. Therefore, the simulation succeeds as long as C's scheduling is such that Stage 2 of the $(A_{\not k},C)$ execution is completed before Stage 2 of the (C,B') execution begins. However this is *exactly* the definition of the scheduling of Case 1. In other words, the simulation is successful if and only if the scheduling is according to Case 1. Now, if the simulation is successful, then P accepts with the same probability as B' would have. On the other hand, if the simulation is not successful (i.e., Case 1 did not occur), then P never accepts. We conclude that the probability that P accepts is exactly equal to the probability that the scheduling is according to Case 1 and B' accepts.

We note that since C' is given the value $Q(w)$, it can also simulate this scenario for C when the augmented definition of security is considered. The rest of the proof of this lemma therefore follows also for the augmented definition. ■

It remains to bound the probability that P accepts in Protocol 6.7.

Claim 6.9 *For every ppt party C' interacting with P in Protocol 6.7 it holds that*

$$\Pr_{Q,w}[P(Q,w) \text{ accepts when interacting with } C'(Q(w))] < 2\epsilon + \mu(n)$$

Proof: We analyze the probability that P accepts in the two-party protocol for P and C' defined above. This is an ordinary two-party setting, and as such it can be analyzed by directly considering the security of the different modules.

We first modify the protocol so that in Step 1, party P sends a random commitment, instead of a commitment to (Q,w). Due to the hiding property of the commitment, this can make at most a negligible difference. (We stress that this replacement has no impact because this commitment is not used anywhere in the rest of the protocol.[35]) Therefore, C' can internally emulate this commitment and this stage can be removed from the protocol. We thus remain with a protocol consisting of the following stages:

- (*Emulation of Stage 2 of* $(A_{\not k}, C)$): P sends C' a commitment to Q and then P and C' execute an augmented polynomial evaluation in which C' receives either $Q(w_C)$ for some w_C (chosen by C'), or $\perp$. By the security of the polynomial evaluation, C' receives either $Q(w_C)$ or $\perp$ and nothing else.
- (*Emulation of Stage 2 of* (C, B')): C' sends P a commitment to some polynomial Q_C and then C' and P execute an augmented polynomial evaluation in which P receives $Q_C(w)$ or $\perp$. By the security of the polynomial evaluation, C' receives *nothing* in this stage.
- (*Emulation of Stage 3 of* (C, B')): C' sends a string y to P and P accepts if $y = f^{2n}(Q_C(w))$.

The intuition behind showing that P accepts with probability at most negligibly greater than 2ϵ is as follows: C' must send the "correct" y based solely on the value $Q(w_C)$ that it (possibly) received from the first evaluation and its auxiliary input $Q(w)$. Now, if $w_C \neq w$, then the only thing that party C' learns about w (from $Q(w)$ and $Q(w_C)$) is that it does not equal w_C. This is due to the "pairwise independence" property of the random polynomial Q. Therefore, C must guess the correct value for y from $|\mathcal{D}| - 1$ possibilities (i.e., $f^{2n}(Q_C(w'))$ for every $w' \neq w_C$). On the other hand, the probability that $w_C = w$ is at most ϵ, because at the time that C' selects w_C it knows nothing about w (although it knows $Q(w)$ for a random Q). A detailed analysis follows.

The above argument is based on the security of the polynomial evaluations. We therefore proceed by analyzing the probability that P accepts in an ideal execution where the two polynomial evaluations are replaced by ideal evaluations. We denote the ideal-model parties by $\hat{P}$ and $\hat{C}'$. By the sequential composition theorem of multi-party computation [15], we have that the accepting probabilities of P (in a real execution) and $\hat{P}$ (in an ideal execution) are at most negligibly different.

We now upper bound the probability that $\hat{P}$ accepts in an ideal execution. Party $\hat{C}'$ is given $Q(w)$ for auxiliary input and in the first polynomial evaluation $\hat{C}'$ inputs a value w_C (of its choice). We differentiate between the case that $w_C = w$ and $w_C \neq w$, and separately upper bound the following probabilities:

[35] This is due to the fact that the zero-knowledge protocol of Stage 3 has already been removed.

1. $\Pr[\hat{P} = \text{acc} \ \& \ w_C = w]$
2. $\Pr[\hat{P} = \text{acc} \ \& \ w_C \neq w]$

Bounding the probability that $\hat{P} = \mathsf{acc}$ and $w_C = w$: We actually show that $\Pr[w_C = w] \leq \epsilon + \mu$ for some negligible function μ. The only message received by $\hat{C}'$ prior to its sending w_C is an (ordinary) commitment to the polynomial Q. That is, $\hat{C}'$'s entire view at this point consists of its auxiliary input $Q(w)$ and $\text{Commit}(Q)$. Due to the hiding property of the commitment, $\text{Commit}(Q)$ can be replaced by $\text{Commit}(0^{2n})$ and this makes at most a negligible difference. We therefore remove the commitment and bound the probability that $w_C = w$, where $\hat{C}'$ is only given $Q(w)$. Since Q is a random linear polynomial, we have that for every w, the string $Q(w)$ is uniformly distributed. That is, $Q(w)$ reveals no information about w. Therefore, we have that $\Pr[w_C = w] \leq \epsilon$ (with equality in case $w_C \in \mathcal{D}$). This implies that when $\hat{C}'$ is given a commitment to Q (rather than to 0^{2n}), we have that $\Pr[w_C = w] \leq \epsilon + \mu(n)$. Therefore,

$$\Pr[\hat{P} = \text{acc} \ \& \ w_C = w] \leq \Pr[w_C = w] \leq \epsilon + \mu(n) \tag{31}$$

Bounding the probability that $\hat{P} = \mathsf{acc}$ and $w_C \neq w$: We actually analyze the following conditional probability: $\Pr[\hat{P} = \text{acc} \mid w_C \neq w]$. Recall that $\hat{C}'$'s view (after the first polynomial evaluation) consists of its random tape, auxiliary input $Q(w)$ and the following messages:

1. A commitment to a polynomial Q sent by $\hat{P}$.

   As before, the commitment to Q can be replaced with a commitment to 0^{2n} with at most a negligible difference. We therefore ignore this part of $\hat{C}'$'s view from now on.

2. An input–output pair $(w_C, Q(w_C))$ (or $(w_C, \perp)$ in the case of incorrect inputs) from the first polynomial evaluation, where $w_C \neq w$.

   We are going to ignore the output case $(w_C, \perp)$, because C' knows a-priori which of the two input/output cases will occur, and we may give it $Q(w_C)$ for free in the incorrect inputs case.

The continuation of the protocol involves $\hat{C}'$ selecting and inputting a polynomial Q_C into the second polynomial evaluation and sending a string y, where $\hat{P}$ accepts if and only if $y = f^{2n}(Q_C(w))$. Restated, the probability that $\hat{P}$ accepts equals the probability that $\hat{C}'$, given its view $(Q(w), w_C, Q(w_C))$, generates a pair (Q_C, y) such that $y = f^{2n}(Q_C(w))$.

Now, the polynomial Q is random and linear, and we are considering the case that $w_C \neq w$. Therefore, by pairwise independence we have that $Q(w)$ is almost uniformly distributed, even given the value of Q at w_C. (Since Q cannot be a constant polynomial, $Q(w)$ is only statistically close to uniform; this is however enough.) This means that given $\hat{C}'$'s view, the password w is almost uniformly distributed in $\mathcal{D} - \{w_C\}$. Since both f^{2n} and Q_C are 1–1 functions, we have that the probability that $\hat{C}'$ generates a pair $(Q_C, f^{2n}(Q_C(w)))$ equals the probability that it guesses w, which equals $\frac{1}{|\mathcal{D}|-1} = \frac{\epsilon}{1-\epsilon}$. Replacing the commitment to 0^{2n} with a commitment to Q, we have that for some negligible function μ,

$$\Pr[\hat{P} = \text{acc} \mid w_C \neq w] \leq \frac{\epsilon}{1-\epsilon} + \mu(n) \tag{32}$$

Combining the bounds: Using Eq. (31) and Eq. (32), we conclude that in an ideal execution:

$$\begin{aligned}
\Pr[\hat{P} = \text{acc}] &= \Pr[\hat{P} = \text{acc} \mid w_C = w] \cdot \Pr[w_C = w] + \Pr[\hat{P} = \text{acc} \mid w_C \neq w] \cdot \Pr[w_C \neq w] \\
&\leq 1 \cdot \Pr[w_C = w] + \frac{\epsilon}{1-\epsilon} \cdot (1 - \Pr[w_C = w]) + \mu(n) \\
\Pr[w_C = w] &= \frac{\epsilon}{1-\epsilon} + \frac{1-2\epsilon}{1-\epsilon} \cdot \Pr[w_C = w] + \mu(n) \\
&\leq \Big(1 + (1-2\epsilon)\Big) \cdot \frac{\epsilon}{1-\epsilon} + \mu(n) \;=\; 2\epsilon + \mu(n)
\end{aligned}$$

where the last inequality is due to $\Pr[w_C = w] \leq \epsilon + \mu$. This implies that in a real execution, the probability that P accepts is at most negligibly greater than 2ϵ. The claim follows. ■

Lemma 6.6 follows by combining Claims 6.8 and 6.9. ■

Lemma 6.1 follows by combining Corollary 6.5 and Lemma 6.6. ■

6.2 Proof of Lemma 4.7 (The Synchronized Case)

Lemma 6.10 (Lemma 4.7 – restated; Case 2 - Synchronized): *Let C be a ppt channel and define* Case 2 *to be a scheduling of the protocol by which C completes the polynomial evaluation with A* after *completing the non-malleable commitment with B. Then*

$$\Pr[B = \text{acc} \;\&\; \pi_A \neq \pi_B \;\&\; \text{Case } 2] \;<\; \epsilon + \mu(n)$$

Proof: The proof of this lemma relies on the non-malleability of the commitment sent in the commitment stage of the protocol. As was explained in the proof sketch, in the case that $\pi_A \neq \pi_B$, the validation stage ensures that B only accepts if the non-malleable commitment he received was to (Q', w), where $Q' \neq Q$ and w is A and B's shared password. (Recall that in the case that $(Q', w') = (Q, w)$, party B rejects with overwhelming probability, unless $\pi_A = \pi_B$.)[36] Furthermore, the probability that C succeeds in generating such a commitment (in which $Q' \neq Q$ and yet w is the second element) is at most negligibly greater than ϵ. We now formally prove both these statements.

In order to use the non-malleability property (of the commitment sent in Stage 1), we define the following relation R. Recall that the non-malleable commitment value sent by A is (Q, w), and denote the value corresponding to the commitment received by B by (Q', w'). Define $R \subset \{0,1\}^{3n} \times \{0,1\}^{3n}$ such that

$$((Q, w), (Q', w')) \in R \text{ if and only if } (Q', w') \neq (Q, w) \text{ and } w' = w. \tag{33}$$

That is, C "succeeds" with respect to R (and thus B may accept) if C does not copy A's commitment (or rather does not commit to the same pair) and yet the second element of the committed pair is the correct password.

We consider the probability that B accepts in Case 2 and $\pi_A \neq \pi_B$ in two complementary subcases. In the first subcase, channel C succeeds with respect to the relation R and in the second subcase, C fails. We prove claims showing the following:

1. Success Case: $\Pr[B = \text{acc} \;\&\; \text{Case } 2 \;\&\; \pi_A \neq \pi_B \;\&\; ((Q, w), (Q', w')) \in R] < \epsilon + \mu(n)$

[36] This is because the validation stage essentially enforces that $\pi_B = Q'(w')$ (see Fact 6.14), and by the case hypothesis $Q'(w') = Q(w) = \pi_A$.

2. Fail Case: $\Pr[B = \text{acc} \ \& \ \text{Case 2} \ \& \ \pi_A \neq \pi_B \ \& \ ((Q,w),(Q',w')) \notin R] < \mu(n)$

The lemma follows by summing up B's accepting probability in the above two subcases. We begin by upper bounding the success case. Specifically, we show that the probability that C succeeds in generating a correct (related) commitment is at most negligibly greater than ϵ.

Claim 6.11 (Success w.r.t R): *Let C be a ppt channel and denote by (Q',w') the value committed to by C in the non-malleable commitment received by B (if the commitment is not valid, then (Q',w') is taken as some fixed value). Then*

$$\Pr[\text{Case 2} \ \& \ ((Q,w),(Q',w')) \in R] < \epsilon + \mu(n)$$

Proof: The definition of non-malleability states that a commitment is non-malleable when run concurrently with another commitment only. Therefore, in a simpler scenario in which the (A,C) and (C,B) non-malleable commitments are run in isolation, we can directly apply the non-malleability property to the relation R that we have defined above. However, in our scenario, other parts of the (A,C) protocol can also be run concurrently to the (C,B) non-malleable commitment. Specifically, by the scheduling of Case 2, (part of) the (A,C) pre-key exchange may run concurrently to the (B,C) commitment (but Stage 3 of the (A,C) execution starts only after the (B,C) commitment ends). The crux of the proof is in showing that the (A,C) pre-key exchange can be simulated. Given such a simulation, we have a scenario in which the (A,C) and (C,B) non-malleable commitments are run in isolation, and thus non-malleability holds.

Recall that A's input to the pre-key exchange stage depends only on the polynomial Q (and is independent of the password w). Therefore, if C has Q, then it can perfectly emulate this stage by itself (this is true irrespective of the security of the modules making up the pre-key exchange stage of the protocol). Fortunately, even if C is explicitly given Q, the probability that C can generate a commitment to (Q',w') for which $(Q',w') \neq (Q,w)$ and $w' = w$ is at most negligibly greater than ϵ (recall that C's sole aim here is to *generate* such a commitment). Thus, we prove that for every ppt channel C given auxiliary input Q, it holds that

$$\Pr[\text{Case 2} \ \& \ ((Q,w),(Q',w')) \in R] < \epsilon + \mu(n)$$

As we have described, C has Q and thus can perfectly emulate the (A,C) pre-key exchange. By the scheduling of Case 2, we have that the (C,B) commit stage concludes before the completion of the (A,C) pre-key exchange. Therefore, the probability that C succeeds with respect to R is the same as when the (A,C) and (C,B) non-malleable commitments are run in isolation.[37] We therefore proceed by upper-bounding the probability that a ppt adversary C (given a commitment to (Q,w) and auxiliary input Q) successfully generates a commitment to (Q',w') where $((Q,w),(Q',w')) \in R$.

Intuitively, A's commitment to (Q,w) does not help C in generating a related commitment. Therefore, the probability of generating a commitment to (Q',w) is the same as the probability of guessing w. Formally, by the definition of non-malleability, for every C there exists a simulator $\hat{C}$ who generates a commitment to $(\hat{Q}',\hat{w}')$ *without* seeing the commitment to (Q,w) such that

$$\left|\Pr[((Q,w),(Q',w')) \in R] - \Pr[((Q,w),(\hat{Q}',\hat{w}')) \in R]\right| < \mu(n)$$

[37] Formally, an adversary attacking a non-malleable commitment protocol (and given Q as auxiliary input) can use C in order to generate a related commitment with the same probability as C succeeds in our session-key protocol when the scheduling is according to Case 2.

Since w is uniformly distributed in $\mathcal{D}$ and $\hat{C}$ is given no information about w, the probability that $\hat{C}$ generates a commitment to $(\hat{Q}', w)$ is at most ϵ. Therefore, the probability that C generates a commitment to (Q', w) where $Q' \neq Q$, is less than $\epsilon + \mu(n)$ as required.

We note that the above holds also for the augmented definition of security. This is because in Case 2, channel C concludes its non-malleable commitment before A terminates. Therefore, it may receive a session-key challenge only after (Q', w') are determined. ■

We now show that when C fails with respect to R, then B accepts with at most negligible probability.

Claim 6.12 (Failure w.r.t R): *For every ppt channel C,*

$$\Pr[B = \text{acc} \ \& \ \pi_A \neq \pi_B \ \& \ ((Q, w), (Q', w')) \notin R] < \mu(n)$$

Proof: In proving this claim, we rely solely on the fact that C "fails" with respect to the relation R, in order to show that B rejects. As described in the proof sketch, intuitively B rejects in this case because the validation stage enforces consistency between the non-malleable commitment, the polynomial input by C into the polynomial evaluation and B's output from the polynomial evaluation. That is, with overwhelming probability, B rejects unless C inputs Q' into the polynomial evaluation *and* B's output from the evaluation equals $Q'(w')$. However, B's input into the polynomial evaluation is w, and thus (by the correctness condition of secure protocols) B's output must equal $Q'(w)$. Thus, with overwhelming probability B rejects unless $Q'(w') = Q'(w)$. As we will show, this implies that $\pi_A = \pi_B$, in contradiction to the claim hypothesis. In the following fact, we formally show that with overwhelming probability, when B accepts, its output from the polynomial evaluation equals $Q'(w)$ (recall that Q' is the polynomial committed to by C in the non-malleable commitment).

Fact 6.13 *For every ppt channel C,*

$$\Pr[B = \text{acc} \ \& \ \pi_B \neq Q'(w)] < \mu(n)$$

Proof: This fact is derived from the correctness condition of the secure polynomial evaluation and the soundness of the zero-knowledge proof. Loosely speaking, the *correctness* condition of a secure two-party protocol states that an adversary cannot cause the output of an honest party to significantly deviate from its output in an ideal execution (where the output is exactly according to the functionality definition). We stress that this has nothing to do with *privacy* and holds even if the adversary knows the honest party's input.

Now, let Q_C be the polynomial in the ordinary commitment sent by C to B before the polynomial evaluation. Then, by the definition of the augmented polynomial evaluation, B's output π_B is either $Q_C(w)$ (in the case of correct inputs) or $\perp$ (in the case of incorrect inputs). Therefore, in a stand-alone two-party setting, we have that with overwhelming probability $\pi_B \in \{Q_C(w), \perp\}$.

We now show that this also holds in our concurrent setting. As we have mentioned, the correctness requirement holds even if the adversary knows the honest party's input. That is, it holds even if C knows w (and Q), in which case C can perfectly emulate the entire (A, C) execution, and we remain with a non-concurrent execution with B. The correctness condition thus holds and we conclude that with overwhelming probability $\pi_B \in \{Q_C(w), \perp\}$. However, since B accepts only if $y = f^{2n}(\pi_B)$ and this never holds when $\pi_B = \perp$, we have $\pi_B = Q_C(w)$ (with overwhelming probability). Getting back to our original concurrent setting, we have:

$$\Pr[B = \text{acc} \ \& \ \pi_B \neq Q_C(w)] < \mu(n)$$

The proof is completed by noticing that the statement proved in the zero-knowledge proof implies (among other things) that $Q_C = Q'$. Thus, by the soundness of the zero-knowledge proof (which also holds in our setting), we conclude that

$$\Pr[B = \text{acc} \ \& \ \pi_B \neq Q'(w)] < \mu(n)$$

■

On the other hand, we now show that when B accepts, with overwhelming probability it holds that $\pi_B = Q'(w')$.

Fact 6.14 *For every ppt channel C,*

$$\Pr[B = \text{acc} \ \& \ \pi_B \neq Q'(w')] < \mu(n)$$

Proof: In the first step of the validation stage, B receives a string y. The statement proved by C (in zero-knowledge) includes the condition $y = f^{2n}(Q'(w'))$. Furthermore, by another check made by B, it rejects unless $y = f^{2n}(\pi_B)$. Since f^{2n} is a 1–1 function, we conclude that with overwhelming probability, B rejects unless $\pi_B = Q'(w')$. ■

We now use the above two facts to show that when $((Q,w),(Q',w')) \notin R$, party B rejects with overwhelming probability. There are two possible cases for which $((Q,w),(Q',w')) \notin R$: either $(Q',w') = (Q,w)$ or $w' \neq w$.

- *Case $(Q',w') = (Q,w)$:* By Fact 6.13 (or equivalently by Fact 6.14), we have that with overwhelming probability, B rejects unless $\pi_A = Q(w) = Q'(w') = \pi_B$, in contradiction to the hypothesis that $\pi_A \neq \pi_B$.
- *Case $w' \neq w$:* By Facts 6.13 and 6.14 that with overwhelming probability whenever B accepts it holds that $Q'(w') = \pi_B = Q'(w)$. However, Q' is a non-constant linear polynomial and is thus 1–1. This implies that $w' = w$, in contradiction to the case hypothesis.

This completes the proof of Claim 6.12. We note that the above proof also holds for the augmented definition of security. This can be seen by noticing that B rejects with overwhelming probability even if C knows Q and w. Therefore, C can generate the session-key challenge itself. ■

Lemma 6.10 is obtained by combining Claims 6.11 and 6.12. ■

7 Simulating the Stand-Alone (A, C) Execution

In this section we show that C's view of its execution with A can be simulated by a non-interactive machine C''. That is,

Theorem 7.1 (Theorem 4.8 restated): *For every ppt channel C' interacting with A only, there exists a* non-interactive *machine C'', such that*

$$\left\{w, k_2(Q(w)), \text{output}(C'^{A(Q,w)}(\sigma))\right\}_{n,\mathcal{D},\sigma} \stackrel{2\epsilon}{\equiv} \{w, U_n, \text{output}(C''(\sigma))\}_{n,\mathcal{D},\sigma}$$

where Q is a random non-constant linear polynomial, $\mathcal{D} \subseteq \{0,1\}^n$ is any ppt samplable dictionary, $w \in_R \mathcal{D}$, and $\epsilon = \frac{1}{|\mathcal{D}|}$.

Proof: As we described in the proof sketch, it is enough to prove that for every ppt channel C',

$$\left\{w, k_2(Q(w)), \text{output}(C'^{A(Q,w)}(\sigma))\right\} \stackrel{2\epsilon}{\equiv} \left\{w, U_n, \text{output}(C'^{A(Q,\tilde{w})}(\sigma))\right\}$$

where $w, \tilde{w} \in_R \mathcal{D}$ are independently chosen passwords from $\mathcal{D}$. This implies the theorem because C'' can simulate C''s view by choosing Q and $\tilde{w}$ and invoking an execution of $C'^{A(Q,\tilde{w})}(\sigma)$. See the proof sketch for details on how C'' works.

Notice that the distributions $\{w, U_n, \text{output}(C'^{A(Q,\tilde{w})})\}$ and $\{\tilde{w}, U_n, \text{output}(C'^{A(Q,w)})\}$ are *equivalent*. We therefore continue by showing that,

$$\left\{w, k_2(Q(w)), \text{output}(C'^{A(Q,w)})\right\} \stackrel{2\epsilon}{\equiv} \left\{\tilde{w}, U_n, \text{output}(C'^{A(Q,w)})\right\} \tag{34}$$

We begin by showing that the pair $(w, Q(w))$ is $(1-\epsilon)$-indistinguishable from $(\tilde{w}, U_n)$ at the end of the polynomial evaluation. For this aim, we consider a modified party A_2 who halts at the conclusion of Stage 2 of the protocol (i.e., after the polynomial evaluation). Then, we show that after an execution of C with A_2, the pair $(w, Q(w))$ is $(1-\epsilon)$-indistinguishable from the pair $(\tilde{w}, U_n)$ with respect to C's view. That is,

Lemma 7.2 *For every ppt channel C' interacting with a party A_2 who halts after the polynomial evaluation,*

$$\left\{w, Q(w), \text{output}(C'^{A_2(Q,w)})\right\} \stackrel{\epsilon}{\equiv} \left\{\tilde{w}, U_n, \text{output}(C'^{A_2(Q,w)})\right\}$$

where Q is a random non-constant linear polynomial, and $w, \tilde{w} \in_R \mathcal{D}$.

Proof: First note that the non-malleable commitment sent by A_2 in this setting plays no role in the continuation of the protocol (the commitment is referred to only in the validation stage, which A_2 does not reach). Due to the hiding property of the commitment, if A_2 commits to all-zeros instead of to the pair (Q, w), this makes at most a negligible difference to C''s view. This enables us to remove the non-malleable commitment entirely, because C' can internally simulate receiving such a commitment. From here on, we consider the modified party A_2 to be a party whose non-malleable commitment is to zeros and who halts after the polynomial evaluation.

What remains is thus the (A_2, C') pre-key exchange, consisting of A_2 sending Commit(Q) to C' followed by a single polynomial evaluation. Since the polynomial evaluation is secure, C can learn at most a single point of $Q(\cdot)$, but otherwise gains no other knowledge of the random polynomial Q (as with the non-malleable commitment, Commit(Q) reveals nothing of Q). As described in the proof sketch, this implies that C can distinguish $Q(w)$ from U_n with probability at most negligibly greater than ϵ (where the ϵ advantage comes from the case that w turns out to equal the input fed by C' into the polynomial evaluation). We now formally show how the limitation on C''s distinguishing capability is derived from the security of the polynomial evaluation.

The security of the polynomial evaluation states that C' can learn no more in a real execution than in an ideal scenario where the polynomial evaluation is replaced by an ideal module computed by a trusted third party. We stress that the other messages sent by the parties in the protocol, including the commitment to Q, remain unmodified in this (partially) ideal execution. Denote the ideal model parties by $\hat{A}_2$ and $\hat{C}'$ and an ideal execution by $\hat{C}'^{\hat{A}_2(Q,w)}$ (in this execution, $\hat{C}'$ is adversarial). By the definition of secure two-party computation, for every real adversary C' interacting with A_2, there exists an ideal adversary $\hat{C}'$ interacting with $\hat{A}_2$ such that the output distributions of C' and $\hat{C}'$ are computationally indistinguishable. However, by the definition of secure computation, the above output distributions should be indistinguishable also when given parties' respective inputs, and specifically A's input (Q, w). That is,

$$\left\{(Q, w), \text{output}(C'^{A_2(Q,w)})\right\} \stackrel{c}{\equiv} \left\{(Q, w), \text{output}(\hat{C}'^{\hat{A}_2(Q,w)})\right\}$$

In particular it follows that

$$\left\{w, Q(w), \text{output}(C'^{A_2(Q,w)})\right\} \stackrel{c}{\equiv} \left\{w, Q(w), \text{output}(\hat{C}'^{\hat{A}_2(Q,w)})\right\} \tag{35}$$

$$\left\{\tilde{w}, U_n, \text{output}(C'^{A_2(Q,w)})\right\} \stackrel{c}{\equiv} \left\{\tilde{w}, U_n, \text{output}(\hat{C}'^{\hat{A}_2(Q,w)})\right\} \tag{36}$$

Thus, it suffices to show that for every ppt party $\hat{C}'$ interacting with $\hat{A}_2$ in an *ideal* execution, it holds that

$$\left\{w, Q(w), \text{output}(\hat{C}'^{\hat{A}_2(Q,w)})\right\} \stackrel{\epsilon}{\equiv} \left\{\tilde{w}, U_n, \text{output}(\hat{C}'^{\hat{A}_2(Q,w)})\right\} \tag{37}$$

We therefore consider an ideal execution of the pre-key exchange consisting of $\hat{A}_2$ sending $\hat{C}'$ a commitment to Q followed by an ideal augmented polynomial evaluation. The view of $\hat{C}'$ in such an execution consists only of a commitment to Q and the result of the polynomial evaluation. (The exact definition of the augmented polynomial evaluation can be found in Section 3.)

Notice that the distributions $\{\tilde{w}, U_n, \text{output}(\hat{C}'^{\hat{A}_2(Q,w)})\}$ and $\{w, U_n, \text{output}(\hat{C}'^{\hat{A}_2(Q,w)})\}$ are equivalent. This is because $\hat{A}_2$ uses w nowhere in the execution (recall that the non-malleable commitment sent by $\hat{A}_2$ is to all-zeros). Therefore, we should actually show that,

$$\left\{w, Q(w), \text{output}(\hat{C}'^{\hat{A}_2(Q,w)})\right\} \stackrel{\epsilon}{\equiv} \left\{w, U_n, \text{output}(\hat{C}'^{\hat{A}_2(Q,w)})\right\} \tag{38}$$

Assume for now that the execution of the polynomial evaluation is such that $\hat{C}'$ always receives $Q(w_C)$ for some w_C input by it into the evaluation (and not $\perp$ as in the case of incorrect inputs). Then, $\hat{C}'$'s view is exactly $(r, \text{Commit}(Q), Q(w_C))$, where r is the string of its random coin tosses and w_C is determined by $\hat{C}'$ based on r and $\text{Commit}(Q)$. For the sake of clarity, we augment the view by w_C itself (i.e., we write $\hat{C}'$'s view as $(r, \text{Commit}(Q), w_C, Q(w_C))$). Assuming without loss of generality that $\hat{C}'$ always outputs its entire view, we conclude that our aim is to show that $\{w, Q(w), (r, \text{Commit}(Q), w_C, Q(w_C))\}$ is $(1-\epsilon)$-indistinguishable from $\{w, U_n, (r, \text{Commit}(Q), w_C, Q(w_C))\}$. We now show that if $w_C \neq w$, then the above two tuples are computationally indistinguishable. That is, we show that

$$\begin{aligned} &\{w, Q(w), (r, \text{Commit}(Q), w_C, Q(w_C)) \mid w_C \neq w\} \\ &\quad \stackrel{c}{\equiv} \{w, U_n, (r, \text{Commit}(Q), w_C, Q(w_C)) \mid w_C \neq w\} \end{aligned} \tag{39}$$

where Q is a random non-constant linear polynomial. First, by the hiding property of the commitment scheme, we can replace the commitment to Q in the above distributions with a commitment to 0^{2n}. (If this makes a non-negligible difference, then $\hat{C}'$ can be used to distinguish a commitment to Q from a commitment to 0^{2n}. We remark that this argument actually only holds if the probability that $w_C \neq w$ is non-negligible. However, this follows from the fact that $w \in_R \mathcal{D}$ appears nowhere in the $\hat{C}'^{A_2(Q,w)}$ execution (recall that the non-malleable commitment has been replaced with a commitment to zeros). Therefore, $w_C = w$ with probability at most ϵ.)

Next, having replace the commitment to Q by a commitment to 0^{2n}, notice that the following distributions are *statistically* close: $\{w, Q(w), (r, \text{Commit}(0^{2n}), w_C, Q(w_C)) \mid w_C \neq w\}$ and $\{w, U_n, (r, \text{Commit}(0^{2n}), w_C, Q(w_C)) \mid w_C \neq w\}$.[38] Then, by returning the commitment to Q in place of the commitment to 0^{2n}, we obtain Eq. (39).

As we have mentioned, the password $w \in_R \mathcal{D}$ does not appear anywhere in the $\hat{C'}^{A_2(Q,w)}$ execution. Therefore, $\Pr[w_C = w] \leq \epsilon$ (with equality when w_C is chosen from $\mathcal{D}$). Therefore, by separately considering the case that $w_C \neq w$ (where the distributions cannot be distinguished) and the case that $w_C = w$ (which occurs with probability at most ϵ), Eq. (38) follows.

This completes the analysis of the simplified case in which the output from the polynomial evaluation is always $Q(w_C)$ for some w_C (and never $\perp$). However, $\hat{C}'$ may cause the result of the evaluation to be $\perp$ and we must show that this cannot help it. Intuitively, if $\hat{C}'$ were to receive $\perp$ then it would learn *nothing* about Q and this would thus be a "bad" strategy. Nevertheless, it must be shown that $\hat{C}'$ cannot learn anything by the mere fact that it received $\perp$ and not $Q(w_C)$. The reason that $\hat{C}'$ indeed learns nothing from the latter event is because it can determine it a-priori (i.e., the output is $\perp$ if and only if $\hat{C}'$ does not supply the commitment explicitly sent to it in the previous step by the honest A (which will always input the corresponding decommit information)). This completes the proof of Lemma 7.2. ■

We now continue by showing that Lemma 7.2 implies Eq. (34) (and thus the current theorem); that is,

$$\left\{w, Q(w), \text{output}(C'^{A_2(Q,w)})\right\} \stackrel{\epsilon}{\equiv} \left\{\tilde{w}, U_n, \text{output}(C'^{A_2(Q,w)})\right\}$$

implies that

$$\left\{w, k_2(Q(w)), \text{output}(C'^{A(Q,w)})\right\} \stackrel{2\epsilon}{\equiv} \left\{\tilde{w}, U_n, \text{output}(C'^{A(Q,w)})\right\}$$

Notice that in the second equation, C' executes a complete execution of the protocol with A, rather than a truncated execution with A_2. Therefore, the additional messages sent by A in the validation stage must be taken into account. Recall that in the validation stage A sends the string $y = f^{2n}(Q(w))$, proves a statement in zero-knowledge and sends a MAC (keyed by $k_1(Q(w))$) of the entire message transcript. In order to simplify the proof, we assume that A sends the MAC-key $k_1(Q(w))$ itself during the validation stage. Given the MAC-key (i.e., $k_1(Q(w))$), the channel C' can always compute the MAC value by itself. Therefore, this only gives C' more information. We start by ignoring the zero-knowledge proof (which, as we show below, is easily justified in this context).

The proof is based on the fact that since $G(s) = (f^{2n}(s), k_1(s), k_2(s))$ is a pseudorandom generator, the output key $k_2(Q(w))$ is $(1 - O(\epsilon))$-pseudorandom, even given $f^{2n}(Q(w))$ and $k_1(Q(w))$. This must be justified, since in our case the generator is seeded by $Q(w)$ that is only $(1 - \epsilon)$-pseudorandom, whereas a generator is usually seeded by a truly random string. In the following claim we show that if, given some information, the string $Q(w)$ is $(1 - \epsilon)$-pseudorandom (as previously shown), then given the same information along with $f^{2n}(Q(w))$ and $k_1(Q(w))$, the string $k_2(Q(w))$ is $(1 - 2\epsilon)$-pseudorandom. (By "given" we mean that a ppt distinguishing machine is given these strings, along with the challenge string which is either $k_2(Q(w))$ or U_n.) Applied to the analysis of our protocol, this means that even after A sends the string $f^{2n}(Q(w))$ and the MAC in

[38] If Q was randomly chosen from all linear polynomials (rather than only from those that are non-constant), then due to pairwise independence the distributions would be identical. However, because Q cannot be constant, $w_C \neq w$ implies that $Q(w_C) \neq Q(w)$ always. On the other hand, $Q(w_C) = U_n$ with probability 2^{-n}. Therefore, with probability 2^{-n} the two distributions can be distinguished by seeing if the last two elements are equal or not. This is the only difference between the distributions and they are therefore statistically close.

the validation stage, the output session-key $k_2(Q(w))$ is $(1-2\epsilon)$-pseudorandom with respect to the view of C'.

Claim 7.3 *Let $I(\cdot)$ be a random process, and assume that $\{w, Q(w), I(Q,w)\}$ is $(1-\epsilon)$-indistinguishable from $\{\tilde{w}, U_n, I(Q,w)\}$. Then $\{w, k_2(Q(w)), I(Q,w), f^{2n}(Q(w)), k_1(Q(w))\}$ is $(1-2\epsilon)$-indistinguishable from $\{\tilde{w}, U_n, I(Q,w), f^{2n}(Q(w)), k_1(Q(w))\}$.*

Indeed, the claim will be applied with $I(Q,w) = \text{output}(C'^{A_2(Q,w)})$.

Proof: We prove the claim in three steps:

1. $\{w, I(Q,w), f^{2n}(Q(w)), k_1(Q(w)), k_2(Q(w))\} \stackrel{\epsilon}{\equiv} \{\tilde{w}, I(Q,w), f^{2n}(U_n), k_1(U_n), k_2(U_n)\}$.

   This is due to the hypothesis $\{w, I(Q,w), Q(w)\} \stackrel{\epsilon}{\equiv} \{\tilde{w}, I(Q,w), U_n\}$.

2. $\{\tilde{w}, I(Q,w), f^{2n}(U_n), k_1(U_n), k_2(U_n)\} \stackrel{c}{\equiv} \{\tilde{w}, I(Q,w), f^{2n}(U_n^{(1)}), k_1(U_n^{(1)}), U_n^{(2)}\}$, where $U_n^{(1)}$ and $U_n^{(2)}$ are two independent uniform distributions.

   This is derived directly from the fact that $(f^{2n}(U_n), k_1(U_n), k_2(U_n))$ is pseudorandom.

3. $\{\tilde{w}, I(Q,w), f^{2n}(U_n^{(1)}), k_1(U_n^{(1)}), U_n^{(2)}\} \stackrel{\epsilon}{\equiv} \{\tilde{w}, I(Q,w), f^{2n}(Q(w)), k_1(Q(w)), U_n^{(2)}\}$

   This is because the hypothesis implies that $\{I(Q,w), U_n\} \stackrel{\epsilon}{\equiv} \{I(Q,w), Q(w)\}$.

Combining the above, we have that

$$\{w, I(Q,w), f^{2n}(Q(w)), k_1(Q(w)), k_2(Q(w))\} \stackrel{2\epsilon}{\equiv} \{\tilde{w}, I(Q,w), f^{2n}(Q(w)), k_1(Q(w)), U_n\}$$

and this completes the proof of the claim. ■

Combining Lemma 7.2 and Claim 7.3, we now establish Eq. (34). We use the following facts:

1. In Stage 3, A sends the string $y = f^{2n}(Q(w))$, proves a statement in zero-knowledge and sends the MAC-key $k_1(Q(w))$ (recall that we assume that A sends the MAC-key instead of the MAC-value).

2. C' can simulate the zero-knowledge proof given by A in Stage 3, because here we are considering a stand-alone execution between A and C. Thus, the view of C' from the entire interaction with A can be generated out of its view of the first two stages (i.e., $\text{output}(C'^{A_2(Q,w)})$), the string $y = f^{2n}(Q(w))$ and the MAC-key $k_1(Q(w))$.

Using $I(Q,w) \stackrel{\text{def}}{=} \text{output}(C'^{A_2(Q,w)})$, Lemma 7.2 asserts that the corresponding hypothesis of Claim 7.3 holds. The corresponding conclusion (of Claim 7.3) implies that Eq. (34) holds (because $\text{output}(C'^{A(Q,w)})$ is easily computed from $\text{output}(C'^{A_2(Q,w)})$, $y = f^{2n}(Q(w))$ and $k_1(Q(w))$). The theorem follows. ■

As we have mentioned in the proof sketch, the above proof remains unchanged when proving the security of Protocol 3.2 with respect to the augmented definition of security (Def. 2.5). This is because in a stand-alone execution between A and C', the channel C' is given the session-key challenge only after the entire execution has been completed. Therefore, the session-key challenge can be generated from the input/output distribution as a post-processing step.

8 Simulating the (C, B) Execution

In this section show how the entire (C, B) execution can be simulated. The simulation is such that the joint distribution of C's view in the simulation along with the password and session key is at most $(1-5\epsilon)$-indistinguishable from the joint distribution of its view in a real execution along with the password and session-key.

Theorem 8.1 (Theorem 4.9 restated – simulating the (C, B) execution): *For every ppt channel C interacting with A and B, there exists a ppt channel C' interacting only with A, such that*

$$\left\{w, k_2(Q(w)), \mathrm{output}({C'}^{A(Q,w)}(\sigma))\right\}_{n,\mathcal{D},\sigma} \stackrel{5\epsilon}{\equiv} \left\{w, k_2(Q(w)), \mathrm{output}(C^{A(Q,w),B(w)}(\sigma))\right\}_{n,\mathcal{D},\sigma}$$

where Q is a random non-constant linear polynomial, $\mathcal{D} \subseteq \{0,1\}^n$ is any ppt samplable dictionary, $w \in_R \mathcal{D}$, and $\epsilon = \frac{1}{|\mathcal{D}|}$.

Proof: As we have described in the proof sketch, this lemma is proved in two stages. First, in Lemma 8.2, we show that when C interacts with A and a modified party $B_{\not{\mathsf{d}}\mathsf{ec}}$ who does not output any $\mathsf{accept/reject}$ bit, then the $(C, B_{\not{\mathsf{d}}\mathsf{ec}})$ execution can be simulated. Next, in Lemma 8.3, we show that B's $\mathsf{accept/reject}$ bit can also be simulated. Combining these two lemmas together, we have that the entire (C, B) execution can be simulated.

8.1 Simulating the $(C, B_{\not{\mathsf{d}}\mathsf{ec}})$ Execution

Lemma 8.2 (Lemma 4.10 restated): *Let $\tilde{C}$ be a ppt channel interacting with A and a modified party $B_{\not{\mathsf{d}}\mathsf{ec}}$ who does not output an* $\mathsf{accept/reject}$ *bit. Then, there exists a ppt channel C' interacting with A* only*, such that*

$$\left\{w, k_2(Q(w)), \mathrm{output}({C'}^{A(Q,w)})\right\} \stackrel{\rm c}{\equiv} \left\{w, k_2(Q(w)), \mathrm{output}(\tilde{C}^{A(Q,w),B_{\not{\mathsf{d}}\mathsf{ec}}(w)})\right\}$$

Proof: Intuitively, $B_{\not{\mathsf{d}}\mathsf{ec}}$'s role can be simulated without any knowledge of w. Loosely speaking, this is because $B_{\not{\mathsf{d}}\mathsf{ec}}$ only uses w in the $(\tilde{C}, B_{\not{\mathsf{d}}\mathsf{ec}})$ polynomial evaluation, and in this evaluation $\tilde{C}$ receives no output. Formally, this is shown by proving that if $B_{\not{\mathsf{d}}\mathsf{ec}}$ were to input an arbitrary, fixed $w' \in_R \mathcal{D}$ (into the polynomial evaluation), instead of w, then $\tilde{C}$ would not be able to tell the difference. That is, for every ppt channel $\tilde{C}$,

$$\left\{w, k_2(Q(w)), \mathrm{output}(\tilde{C}^{A(Q,w),B_{\not{\mathsf{d}}\mathsf{ec}}(w)})\right\} \stackrel{\rm c}{\equiv} \left\{w, k_2(Q(w)), \mathrm{output}(\tilde{C}^{A(Q,w),B_{\not{\mathsf{d}}\mathsf{ec}}(w')})\right\} \qquad (40)$$

(Observe that in the second distribution, $B_{\not{\mathsf{d}}\mathsf{ec}}$'s input is w'.) We prove Eq. (40) even when $\tilde{C}$ is given Q and w as auxiliary input. Now, since Q and w constitute all of A's input, the channel $\tilde{C}(Q, w)$ can perfectly simulate the entire (A, C) execution. That is, for every ppt channel $\tilde{C}$ there exists a ppt channel $\hat{C}$ such that the following two equations hold:

$$\begin{aligned} \{w, k_2(Q(w)), \mathrm{output}(\hat{C}^{B_{\not{\mathsf{d}}\mathsf{ec}}(w)}(Q,w))\} &\equiv \{w, k_2(Q(w)), \mathrm{output}(\tilde{C}^{A(Q,w),B_{\not{\mathsf{d}}\mathsf{ec}}(w)}(Q,w))\} \\ \{w, k_2(Q(w)), \mathrm{output}(\hat{C}^{B_{\not{\mathsf{d}}\mathsf{ec}}(w')}(Q,w))\} &\equiv \{w, k_2(Q(w)), \mathrm{output}(\tilde{C}^{A(Q,w),B_{\not{\mathsf{d}}\mathsf{ec}}(w')}(Q,w))\} \end{aligned}$$

It thus remains to show that

$$\left\{w, k_2(Q(w)), \mathrm{output}(\hat{C}^{B_{\not{\mathsf{d}}\mathsf{ec}}(w)}(Q,w))\right\} \stackrel{\rm c}{\equiv} \left\{w, k_2(Q(w)), \mathrm{output}(\hat{C}^{B_{\not{\mathsf{d}}\mathsf{ec}}(w')}(Q,w))\right\} \qquad (41)$$

The latter is derived from the security of the polynomial evaluation. Intuitively, $\hat{C}$ obtains no output from the polynomial evaluation, whereas the polynomial evaluation is the only part of the entire protocol in which $B_{\not{d}\mathsf{ec}}$ uses his input (of w or w'). Formally, we may consider a ppt $\hat{C}'$ that emulates the entire $(\hat{C}, B_{\not{d}\mathsf{ec}})$ execution except for the polynomial evaluation that it actually performs with $\hat{B}'$ that uses input w or w'. That is, $\hat{B}'$ is merely playing receiver in the polynomial evaluation protocol, whereas $\hat{C}'$ is some (fancy) adversary for that protocol. Still, the security of the latter protocol (as stand-alone) implies that $\hat{C}'$ cannot distinguish the case that $\hat{B}'$ has input w from the case that it has input w' (because the ideal-model adversary cannot do so). We conclude that $\hat{C}$ can distinguish the cases that $B_{\not{d}\mathsf{ec}}$ has input w or w' with at most negligible probability. Eq. (41) follows, and thus so does Eq. (40).

We are now ready to show how C' works (recall that C' interacts with A only and its aim is to emulate an execution of $\tilde{C}$ with A *and* $B_{\not{d}\mathsf{ec}}$). Machine C' begins by selecting an arbitrary $w' \in D$. Then, C' perfectly emulates an execution of $\tilde{C}^{A(Q,w),B_{\not{d}\mathsf{ec}}(w')}$ by playing $B_{\not{d}\mathsf{ec}}$'s role in the $(C, B_{\not{d}\mathsf{ec}})$ execution (C' can play $B_{\not{d}\mathsf{ec}}(w')$'s role because w' is known to it), and relaying all messages belonging to the (A, C) execution (i.e., passing each message sent by A to $\tilde{C}$, and each message sent by $\tilde{C}$ to A). Finally, C' outputs whatever $\tilde{C}$ does. By the definition of C', we have

$$\left\{w, k_2(Q(w)), \mathrm{output}(\tilde{C}^{A(Q,w),B_{\not{d}\mathsf{ec}}(w')})\right\} \equiv \left\{w, k_2(Q(w)), \mathrm{output}({C'}^{A(Q,w)})\right\} \tag{42}$$

The lemma follows from Equations (40) and (42). ■

8.2 Simulating B's Accept/Reject Decision Bit

Lemma 8.3 (Lemma 4.11 restated): *Let $B_{\not{d}\mathsf{ec}}$ be a modified party that does not output an* accept/reject *bit. Then, for every ppt channel C interacting with A and B, there exists a ppt channel $\tilde{C}$ interacting with A and $B_{\not{d}\mathsf{ec}}$, such that*

$$\left\{w, k_2(Q(w)), \mathrm{output}(\tilde{C}^{A(Q,w),B_{\not{d}\mathsf{ec}}(w)})\right\} \stackrel{5\epsilon}{\equiv} \left\{w, k_2(Q(w)), \mathrm{output}(C^{A(Q,w),B(w)})\right\}$$

Proof: We prove this lemma by showing how the accept/reject bit of B can be predicted by C. Specifically, we show that the MAC sent in the validation stage is such that if C was not reliable, then B rejects with probability $1-5\epsilon$. This enables $\tilde{C}$ to "predict" B's output-bit based on whether or not C was reliable. We thus start by proving the security of the MAC when keyed by $k_1(\pi_A)$ (recall that $\pi_A \stackrel{\mathrm{def}}{=} Q(w)$). As we have mentioned in the proof sketch, we need to show that the MAC is secure only *before* B outputs its accept/reject bit. Thus, we consider a scenario in which C interacts with A and the modified party $B_{\not{d}\mathsf{ec}}$. In the following claim, we formally state the security of the MAC. (Recall that for simplicity, we consider an implementation of a MAC by a pseudorandom function. However, our proof can be extended to any secure implementation of a MAC.)

Claim 8.4 (Claim 4.12 restated): *Let C be an arbitrary ppt channel interacting with A and a modified party $B_{\not{d}\mathsf{ec}}$ as in Lemma 8.3, and let t_A and t_B denote the transcripts from the (A, C) and $(C, B_{\not{d}\mathsf{ec}})$ executions, respectively. Then, if $t_A \neq t_B$, it holds that the value $MAC_{k_1(\pi_A)}(t_B)$ is $(1-2\epsilon)$-pseudorandom with respect to C's view. That is,*

$$\left\{MAC_{k_1(\pi_A)}(t_B), C^{A(Q,w),B_{\not{d}\mathsf{ec}}(w)}\right\} \stackrel{2\epsilon}{\equiv} \left\{U_n, C^{A(Q,w),B_{\not{d}\mathsf{ec}}(w)}\right\}$$

Proof: We prove this claim by first showing that the MAC-key $k_1(\pi_A)$ is $(1-2\epsilon)$-pseudorandom before A sends the MAC in the validation stage. Formally, consider a modified party $A_{\not{\mathsf{m}}\mathsf{ac}}$ who is exactly the same as A excepts that it does not send the MAC message. Then, we show that for every ppt C,

$$\left\{k_1(Q(w)), C^{A_{\not{\mathsf{m}}\mathsf{ac}}(Q,w), B_{\not{\mathsf{d}}\mathsf{ec}}(w)}\right\} \stackrel{2\epsilon}{\equiv} \left\{U_n, C^{A_{\not{\mathsf{m}}\mathsf{ac}}(Q,w), B_{\not{\mathsf{d}}\mathsf{ec}}(w)}\right\} \tag{43}$$

where Q is a random non-constant linear polynomial and $w \in_R \mathcal{D}$. First, we claim that for every ppt channel C interacting with $A_{\not{\mathsf{m}}\mathsf{ac}}$ and $B_{\not{\mathsf{d}}\mathsf{ec}}$, there exists a ppt channel C' interacting only with $A_{\not{\mathsf{m}}\mathsf{ac}}$, such that

$$\left\{k_1(Q(w)), C'^{A_{\not{\mathsf{m}}\mathsf{ac}}(Q,w)}\right\} \stackrel{\rm c}{\equiv} \left\{k_1(Q(w)), C^{A_{\not{\mathsf{m}}\mathsf{ac}}(Q,w), B_{\not{\mathsf{d}}\mathsf{ec}}(w)}\right\}$$

and

$$\left\{U_n, C'^{A_{\not{\mathsf{m}}\mathsf{ac}}(Q,w)}\right\} \stackrel{\rm c}{\equiv} \left\{U_n, C^{A_{\not{\mathsf{m}}\mathsf{ac}}(Q,w), B_{\not{\mathsf{d}}\mathsf{ec}}(w)}\right\}$$

The above two equations can be shown in the same way as Lemma 8.2 above. Therefore, in order to obtain Eq. (43), it is enough to show that for every ppt channel C' interacting only with $A_{\not{\mathsf{m}}\mathsf{ac}}(Q,w)$

$$\left\{k_1(Q(w)), C'^{A_{\not{\mathsf{m}}\mathsf{ac}}(Q,w)}\right\} \stackrel{2\epsilon}{\equiv} \left\{U_n, C'^{A_{\not{\mathsf{m}}\mathsf{ac}}(Q,w)}\right\}$$

Now, by Lemma 7.2, we have that after the conclusion of the polynomial evaluation between $A_{\not{\mathsf{m}}\mathsf{ac}}$ and C', it holds that $Q(w)$ is $(1-\epsilon)$-pseudorandom to C'. We claim that this implies that $k_1(Q(w))$ is $(1-2\epsilon)$-pseudorandom to C' at the conclusion of the complete protocol between $A_{\not{\mathsf{m}}\mathsf{ac}}$ and C. This can be shown using an almost identical proof as in Claim 7.3. (We note that here we are in a standard stand-alone setting, and so the zero-knowledge proof can be simulated, and reveals nothing about $k_1(Q(w))$.) This completes the proof of Eq. (43).

We have so far established that the MAC-key $k_1(Q(w))$ used by A is $(1-2\epsilon)$-pseudorandom with respect to C's view. It thus remains to show that using a $(1-2\epsilon)$-pseudorandom string as a key to a pseudorandom function (for the MAC) yields a $(1-2\epsilon)$-pseudorandom function. This then implies that when $t_A \neq t_B$, the value $MAC_{k_1(\pi_A)}(t_B)$ is $(1-2\epsilon)$-pseudorandom with respect to C's view. (Recall that A sends $MAC_{k_1(\pi_A)}(t_A)$ in the protocol execution and thus this value itself is not $(1-2\epsilon)$-pseudorandom. However, since the MAC used is a $(1-2\epsilon)$-pseudorandom function, the claim holds for any t that C produces, including t_B that is part of its view.) In the following claim, we prove that $MAC_{k_1(\pi_A)}(\cdot)$ is a $(1-2\epsilon)$-pseudorandom function. As in Claim 7.3, we model any information C may have learned about Q and w during the protocol with $A_{\not{\mathsf{m}}\mathsf{ac}}$ by a random process $I(\cdot)$ (i.e., $I(Q,w)$ denotes output$(C^{A_{\not{\mathsf{m}}\mathsf{ac}}(Q,w)})$).

Claim 8.5 *Assume that $\{k_1(Q(w)), I(Q,w)\}$ is $(1-2\epsilon)$-indistinguishable from $\{U_n, I(Q,w)\}$. Furthermore, let $f_r(\cdot)$ be a pseudorandom function when r is uniformly distributed. Then, given $I(Q,w)$, the function $f_{k_1(Q(w))}(\cdot)$ is $(1-2\epsilon)$-pseudorandom.*

Proof: The proof is based on the idea that a string distinguisher that needs to distinguish $k_1(Q(w))$ from U_n can simulate oracle queries to $f_{k_1(Q(w))}(\cdot)$ or $f_{U_n}(\cdot)$ depending on its input. Since we know that $f_{U_n}(\cdot)$ is indistinguishable from a random function (by the claim hypothesis), distinguishing $f_{k_1(Q(w))}(\cdot)$ from a random function essentially means distinguishing $k_1(Q(w))$ from U_n. Details follow.

Let D be a ppt oracle machine that receives the output of the random process $I(Q,w)$ as well as oracle access to either $f_{k_1(Q(w))}$ or a random function f. Then,

$$\left|\Pr[D^{f_{k_1(Q(w))}}(I(Q,w), 1^n) = 1] - \Pr[D^f(I(Q,w), 1^n) = 1]\right|$$

$$\leq \left|\Pr[D^{f_{k_1(Q(w))}}(I(Q,w),1^n)=1] - \Pr[D^{f_{U_n}}(I(Q,w),1^n)=1]\right| \quad (44)$$

$$+ \left|\Pr[D^{f_{U_n}}(I(Q,w),1^n)=1] - \Pr[D^{f}(I(Q,w),1^n)=1]\right| \quad (45)$$

Eq. (45) is negligible by the definition of a pseudorandom function. On the other hand, Eq. (44) is bounded above by $2\epsilon + \mu(n)$, because otherwise a ppt machine D' that, given $I(Q,w)$ and trying to distinguish $k_1(Q(w))$ from U_n, can invoke D on input $(I(Q,w),1^n)$ and answer all oracle queries by using f_x, where x denotes its input string (which is either $k_1(Q(w))$ or U_n). ■

This completes the proof of Claim 8.4. ■

We are now ready to show how $\tilde{C}$ works. Channel $\tilde{C}$ runs the protocol (with A and $B_{\not{d}\mathsf{ec}}$) by passing all messages, unmodified, via C. Furthermore, $\tilde{C}$ checks whether or not C was reliable during the execution. Recall that C is reliable if the (A,C) and (C,B) executions are run in a synchronized manner, and C does not modify any of the messages sent by A or B. This is a syntactic feature, which is easily checked by $\tilde{C}$ (since it views the entire interaction). If C was reliable then $\tilde{C}$ outputs accept for B, otherwise it outputs reject for B. This completes the simulation of C's interaction with A and B. Let $\chi_{\tilde{C}}$ denote the simulated $\mathsf{accept}/\mathsf{reject}$ bit output by $\tilde{C}$.

Now, when $\tilde{C}$ predicts B's output bit correctly, we have that C's view in this simulation is identical to a real execution with A and B. This means that the difference in the experiments referred to in the lemma's conclusion equals the probability that $\tilde{C}$'s prediction is wrong (i.e., the probability that $\mathrm{dec}_B = \mathsf{acc}$ and $\chi_{\tilde{C}} = \mathsf{rej}$ or vice versa). Noticing that $\chi_{\tilde{C}} = \mathsf{acc}$ if and only if C is reliable, we have that for every distinguisher D,

$$\left|\Pr_{Q,w}\left[D\Big(w, k_2(Q(w)), \tilde{C}^{A(Q,w),B_{\not{d}\mathsf{ec}}(w)}\Big)=1\right] - \Pr_{Q,w}\left[D\Big(w, k_2(Q(w)), C^{A(Q,w),B(w)}\Big)=1\right]\right|$$
$$\leq \Pr[\mathrm{dec}_B = \mathsf{acc} \;\&\; \mathrm{reliable}_C = \mathsf{false}] + \Pr[\mathrm{dec}_B = \mathsf{rej} \;\&\; \mathrm{reliable}_C = \mathsf{true}]$$

First, notice that when C is reliable, B *always* accepts. That is, $\Pr[\mathrm{dec}_B = \mathsf{rej} \;\&\; \mathrm{reliable}_C = \mathsf{true}]$ equals *zero*. We now show that $\Pr[\mathrm{dec}_B = \mathsf{acc} \;\&\; \mathrm{reliable}_C = \mathsf{false}]$ is at most negligibly more than 5ϵ, and this completes the proof of Lemma 8.3.

Proposition 8.6 (Proposition 4.13 – restated): *For every ppt channel C,*

$$\Pr[\mathrm{dec}_B = \mathsf{acc} \;\&\; \mathrm{reliable}_C = \mathsf{false}] < 5\epsilon + \mu(n)$$

Proof: The proof of this proposition is based on the security of the MAC sent in the validation stage. Intuitively, sending a MAC on the entire session transcript ensures that if any messages were modified (as in the case of an unreliable C), then this will be noticed by B. However, in our protocol, A and B may have *different* MAC-keys (in which case nothing can be said about detecting C's malicious behavior). Fortunately, the key-match property ensures that this happens (undetectably by B) with probability at most 3ϵ.

The security of the MAC, shown above in Claim 8.4 states the following. Let t_A and t_B be A and B's respective message transcripts. Then if $t_A \neq t_B$, the string $MAC_{k_1(\pi_A)}(t_B)$ is $(1-2\epsilon)$-pseudorandom with respect to C's view. By the definition of reliability, if C is not reliable then indeed $t_A \neq t_B$ and so $MAC_{k_1(\pi_A)}(t_B)$ is $(1-2\epsilon)$-pseudorandom with respect to C's view.

Now, party B's protocol definition is such that he rejects unless the last message he receives equals $MAC_{k_1(\pi_B)}(t_B)$, where $k_1(\pi_B)$ is the MAC-key used by B. We stress that the key used by B for the MAC is $k_1(\pi_B)$, whereas Claim 8.4 refers to a MAC keyed by $k_1(\pi_A)$. However, if $\pi_A = \pi_B$

then $k_1(\pi_A) = k_1(\pi_B)$. Therefore, if $\pi_A = \pi_B$ then the claim holds and the probability that C generates the correct MAC-value is at most negligibly greater than 2ϵ. That is,

$$\Pr[B = \text{acc} \ \& \ \text{reliable}_C = \mathsf{false} \ \& \ \pi_A = \pi_B] < 2\epsilon + \mu(n)$$

On the other hand, if $\pi_A \neq \pi_B$ then irrespective of the MAC, the probability that B accepts is at most negligibly more than 3ϵ. This is due to the key-match property proven in Theorem 4.5. We conclude that

$$\begin{aligned} &\Pr[\text{dec}_B = \mathsf{acc} \ \& \ \text{reliable}_C = \mathsf{false}] \\ &\quad = \quad \Pr[\text{dec}_B = \mathsf{acc} \ \& \ \text{reliable}_C = \mathsf{false} \ \& \ \pi_A \neq \pi_B] \\ &\qquad\quad + \Pr[\text{dec}_B = \mathsf{acc} \ \& \ \text{reliable}_C = \mathsf{false} \ \& \ \pi_A = \pi_B] \\ &\quad < \quad 3\epsilon + 2\epsilon + \mu(n) \end{aligned}$$

and the proposition follows. ■

As stated above, this completes the proof of Lemma 8.3. ■

8.3 Conclusion

Theorem 8.1 is obtained by combining Lemmas 8.2 and 8.3. ■

We note that when considering the augmented definition of security (Def. 2.5), the proof of Lemma 8.2 remains unchanged. This is because Eq. (40) (and thus the lemma) holds even in the case that $\tilde{C}$ is given Q and w and can generate the session-key challenge itself. On the other hand, there are some minor differences in the proof of Lemma 8.3. In particular, one must justify the correctness of Claim 8.4 even when C may also be given $k_2(Q(w))$. That is, we must show that the MAC-key $k_1(Q(w))$ remains $(1-2\epsilon)$-pseudorandom to C, even if it is given the session-key $k_2(Q(w))$ as well. However, this is derived from the property of the pseudorandom generator $G(s) = (k_1(s), k_2(s), f^{2n}(s))$. The actual proof of this is very similar to that of Claim 7.3. The rest of the proof remains almost the same.

Acknowledgements

We would like to thank Moni Naor for suggesting this problem to us and for his valuable input in the initial stages of our research. We are also grateful to Alon Rosen for much discussion and feedback throughout the development of this work. We also thank Jonathan Katz for helpful discussions, and Ran Canetti, Shai Halevi and Tal Rabin for discussion that led to a significant simplification of the protocol. Finally, we thank the anonymous referees for their many helpful comments.

References

[1] B. Barak. Constant-Round Coin-Tossing With a Man in the Middle or Realizing the Shared Random String Model. In *43rd IEEE Symposium on the Foundations of Computer Science*, 2002.

[2] D. Beaver. Secure Multi-party Protocols and Zero-Knowledge Proof Systems Tolerating a Fault Minority. *Journal of Cryptology*, 4(2):75–122, 1991.

[3] M. Bellare, R. Canetti and H. Krawczyk. Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols. In *30th ACM Symposium on Theory of Computing*, pages 419–428, 1998.

[4] M. Bellare, D. Pointcheval and P. Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. In *EUROCRYPT'00*, Springer-Verlag (LNCS 1807), pages 139–155, 2000.

[5] M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[6] M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *CRYPTO'93*, Springer-Verlag (LNCS 773), pages 232–249, 1994.

[7] M. Bellare and P. Rogaway. Provably Secure Session Key Distribution: The Three Party Case. In *27th ACM Symposium on Theory of Computing*, pages 57–66, 1995.

[8] S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *ACM/IEEE Symposium on Research in Security and Privacy*, pages 72–84, 1992.

[9] S.M. Bellovin and M. Merritt. Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In *1st ACM Conference on Computer and Communication Security*, pages 244–250, 1993.

[10] M. Blum. Coin Flipping by Phone. *IEEE Spring COMPCOM*, pages 133–137, February 1982.

[11] M. Blum and S. Goldwasser. An Efficient Probabilistic Public-Key Encryption Scheme which hides all partial information. In *CRYPTO'84*, Springer-Verlag (LNCS 196), pages 289–302.

[12] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM Journal of Computation*, 13(4):850–864, 1984.

[13] M. Boyarsky. Public-key Cryptography and Password Protocols: The Multi-User Case. In *6th ACM Conference on Computer and Communication Security*, 1999.

[14] V. Boyko, P. MacKenzie and S. Patel. Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman. In *EUROCRYPT'00*, Springer-Verlag (LNCS 1807), pages 156–171, 2000.

[15] R. Canetti. Security and Composition of Multi-party Cryptographic Protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[16] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *42nd IEEE Symposium on the Foundations of Computer Science*, pages 136–145, 2001.

[17] R. Canetti, O. Goldreich, and S. Halevi. The Random Oracle Methodology, Revisited. In *30th ACM Symposium on Theory of Computing*, pages 209–218, 1998.

[18] G. Di Crescenzo, Y. Ishai and R. Ostrovsky. Non-Interactive and Non-Malleable Commitment. In *30th ACM Symposium on Theory of Computing*, pages 141-150, 1998.

[19] W. Diffie, and M.E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22, pages 644–654, 1976

[20] W. Diffie, P.C. Van Oorschot and M.J. Wiener. Authentication and Authenticated Key Exchanges. *Designs Codes and Cryptography*, 2(2):107–125, 1992.

[21] D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. In *SIAM Journal on Computing*, 30(2):391–437, 2000.

[22] U. Feige and A. Shamir. Witness Indistinguishability and Witness Hiding Protocols. In *22nd ACM Symposium on Theory of Computing*, pages 416–426, 1990.

[23] M. Fischlin and R. Fischlin. Efficient Non-malleable Commitment Schemes. In *CRYPTO 2000*, Springer-Verlag (LNCS 1880), pages 413–431.

[24] Y. Gertner, S. Kannan, T. Malkin, O. Reingold and M. Viswanathan. The Relationship between Public-Key Encryption and Oblivious Transfer. In *41st IEEE Symposium on the Foundations of Computer Science*, 2000.

[25] O. Goldreich. *Secure Multi-Party Computation.* Manuscript. Preliminary version, 1998. Available from `http://www.wisdom.weizmann.ac.il/~oded/pp.html`.

[26] O. Goldreich. *Foundation of Cryptography: Volume 1 – Basic Tools.* Cambridge University Press, 2001.

[27] O. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions. *Journal of the ACM*, 33(4):792–807, 1986.

[28] O. Goldreich and A. Kahan. How To Construct Constant-Round Zero-Knowledge Proof Systems for NP. *Journal of Cryptology*, 9(3):167–190, 1996.

[29] O. Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing but their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *Journal of the ACM*, 38(1):691–729, 1991.

[30] O. Goldreich, S. Micali and A. Wigderson. How to Play any Mental Game – A Completeness Theorem for Protocols with Honest Majority. In *19th ACM Symposium on Theory of Computing,* pages 218–229, 1987. For details see [25].

[31] S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[32] S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal of Computation,* 18(1):186–208, 1989.

[33] S. Halevi and H. Krawczyk. Public-Key Cryptography and Password Protocols. In *5th ACM Conference on Computer and Communications Security*, pages 122–131, 1998.

[34] D.P. Jablon. Strong password-only authenticated key exchange. *SIGCOMM Computer Communications Review*, 26(5):5–26, 1996.

[35] J. Katz, R. Ostrovsky and M. Yung. Practical Password-Authenticated Key Exchange Provably Secure under Standard Assumptions. In *EUROCRYPT 2001*, Springer-Verlag (LNCS 2045), pages 475–494, 2001.

[36] C. Kaufman, R. Perlman and M. Speciner. *Network Security.* Prentice Hall, 1997.

[37] J. Kilian. Basing Cryptography on Oblivious Transfer. In *20th ACM Symposium on Theory of Computing*, pages 20–31, 1988.

[38] Y. Lindell. Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation. In *CRYPTO 2001*, Springer-Verlag (LNCS 2139), pages 171–189, 2001.

[39] S. Lucks. Open key exchange: How to defeat dictionary attacks without encrypting public keys. In *Workshop on Security Protocols*, Ecole Normale Superieure, 1997.

[40] A. Menezes, P. Van Oorschot and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.

[41] S. Micali and P. Rogaway. Secure Computation. Unpublished manuscript, 1992. Preliminary version in *Crypto'91*, Springer-Verlag (LNCS 576), 1991.

[42] M. Naor. Bit Commitment using Pseudorandom Generators. *Journal of Cryptology*, 4(2):151–158, 1991.

[43] M. Naor, R. Ostrovsky, R. Venkatesan and M. Yung. Zero-Knowledge Arguments for NP can be Based on General Assumptions. *Journal of Cryptology*, 11(2):87–108, 1998.

[44] M. Naor and B. Pinkas. Oblivious Transfer and Polynomial Evaluation. In *31st ACM Symposium on Theory of Computing*, pages 245-254, 1999.

[45] R.M. Needham and M.D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM*, 21(12):993–999, 1978.

[46] S. Patel. Number theoretic attacks on secure password schemes. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 236–247, 1997.

[47] R. Richardson and J. Kilian. On the Concurrent Composition of Zero-Knowledge Proofs. In *EUROCRYPT'99*, Springer-Verlag (LNCS 1592), pages 415–431.

[48] R. Rivest, A. Shamir and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[49] V. Shoup. On Formal Models for Secure Key Exchange. *Cryptology ePrint Archive*, Report 1999/012, `http://eprint.iacr.org/`.

[50] M. Steiner, G. Tsudik and M. Waidner. Refinement and extension of encrypted key exchange. *ACM SIGOPS Operating Systems Review*, 29(3):22–30, 1995.

[51] T. Wu. The secure remote password protocol. In *1998 Internet Society Symposium on Network and Distributed System Security*, pages 97–111, 1998.

[52] A.C. Yao. Theory and Application of Trapdoor Functions. In *23rd IEEE Symposium on the Foundations of Computer Science*, pages 80–91, 1982.

[53] A.C. Yao. How to Generate and Exchange Secrets. In *27th IEEE Symposium on the Foundations of Computer Science*, pages 162–167, 1986.

A Cryptographic Tools

In this section we briefly describe the tools used in our construction. That is, we describe secure two-party computation, string commitment and non-malleable string commitment, the Richardson-Kilian zero-knowledge proof system, seed-committed pseudorandom generators and message authentication codes. We present comprehensive and formal definitions for secure two-party computation as this forms the basis for the majority of our proofs.

A.1 Secure Two-Party Computation

In this section we present definitions for secure two-party computation. The following description and definition is taken from [25].

A two-party protocol problem is cast by specifying a random process which maps pairs of inputs (one input per each party) to pairs of outputs (one per each party). We refer to such a process as the desired functionality, denoted $f : \{0,1\}^* \times \{0,1\}^* \mapsto \{0,1\}^* \times \{0,1\}^*$. That is, for every pair of inputs (x, y), the desired output-pair is a random variable, $f(x, y)$, ranging over pairs of strings. The first party, holding input x, wishes to obtain the first element in $f(x, y)$; whereas the second party, holding input y, wishes to obtain the second element in $f(x, y)$.

Whenever we consider a protocol for securely computing f, it is implicitly assumed that the protocol is correct provided that both parties follow the prescribed program. That is, the joint output distribution of the protocol, played by honest parties, on input pair (x, y), equals the distribution of $f(x, y)$.

We consider arbitrary feasible deviation of parties from a specified two-party protocol. A few preliminary comments are in place. Firstly, there is no way to force parties to participate in the protocol. That is, possible malicious behavior may consist of not starting the execution at all, or, more generally, suspending (or aborting) the execution at any desired point in time. In particular, a party can abort at the first moment when it obtains the desired result of the computed functionality. We stress that our model of communication does not allow us to condition the receipt of a message by one party on the *concurrent* sending of a proper message by this party. Thus, no two-party protocol can prevent one of the parties from aborting when obtaining the desired result and before its counterpart also obtains the desired result. In other words, it can be shown that perfect fairness – in the sense of both parties obtaining the outcome of the computation concurrently – is not achievable in two-party computation. We thus give up on such fairness altogether.

Another point to notice is that there is no way to talk of the *correct input* to the protocol. That is, a party can always modify its local input, and there is no way for a protocol to prevent this.

To summarize, there are three things we cannot hope to avoid.

1. Parties refusing to participate in the protocol (when the protocol is first invoked).
2. Parties substituting their local input (and entering the protocol with an input other than the one provided to them).
3. Parties aborting the protocol prematurely (e.g., before sending their last message).

The ideal model. We now translate the above discussion into a definition of an ideal model. That is, we will allow in the ideal model whatever cannot be possibly prevented in any real execution. An alternative way of looking at things is that we assume that the two parties have at their disposal a trusted third party, but even such a party cannot prevent specific malicious behavior. Specifically, we allow a malicious party in the ideal model to refuse to participate in the protocol or to substitute its local input. (Clearly, neither can be prevent by a trusted third party.) In addition, we postulate that the *first* party has the option of "stopping" the trusted party just after obtaining its part of the output, and before the trusted party sends the other output-part to the second party. Such an option is not given to the second party.[39] Thus, an execution in the ideal model proceeds as follows (where all actions of the both honest and malicious party must be feasible to implement).

Inputs: Each party obtains an input, denoted z.

Send inputs to trusted party: An honest party always sends z to the trusted party. A malicious party may, depending on z, either abort or sends some $z' \in \{0,1\}^{|z|}$ to the trusted party.

Trusted party answers first party: In case it has obtained an input pair, (x,y), the trusted party (for computing f), first replies to the first party with $f_1(x,y)$. Otherwise (i.e., in case it receives only one input), the trusted party replies to both parties with a special symbol, $\perp$.

Trusted party answers second party: In case the first party is malicious it may, depending on its input and the trusted party answer, decide to *stop* the trusted party. In this case the trusted party sends $\perp$ to the second party. Otherwise (i.e., if not stopped), the trusted party sends $f_2(x,y)$ to the second party.

Outputs: An honest party always outputs the message it has obtained from the trusted party. A malicious party may output an arbitrary (polynomial-time computable) function of its initial input and the message it has obtained from the trusted party.

The ideal model computation is captured in the following definition.[40]

Definition A.1 (malicious adversaries, the ideal model): *Let* $f : \{0,1\}^* \times \{0,1\}^* \mapsto \{0,1\}^* \times \{0,1\}^*$ *be a functionality, where* $f_1(x,y)$ (resp., $f_2(x,y)$) *denotes the first* (resp., second) *element of* $f(x,y)$. *Let* $\overline{C} = (C_1, C_2)$ *be a pair of polynomial-size circuit families representing adversaries in the ideal model. Such a pair is* admissible (in the ideal malicious model) *if for at least one* $i \in \{1,2\}$ *we have* $C_i(I) = I$ *and* $C_i(I,O) = O$. *The* joint execution under $\overline{C}$ in the ideal model (on input pair (x,y)), *denoted* $\text{ideal}_{f,\overline{C}}(x,y)$, *is defined as follows*

- *In case* $C_2(I) = I$ *and* $C_2(I,O) = O$ (i.e., Party 2 is honest),

$$
\begin{array}{lll}
(C_1(x,\perp)\,,\,\perp) & \text{if } C_1(x) = \perp & (46)\\
(C_1(x, f_1(C_1(x),y), \perp)\,,\,\perp) & \text{if } C_1(x) \neq \perp \text{ and } C_1(x, f_1(C_1(x),y)) = \perp & (47)\\
(C_1(x, f_1(C_1(x),y))\,,\, f_2(C_1(x),y)) & \text{otherwise} & (48)
\end{array}
$$

- *In case* $C_1(I) = I$ *and* $C_1(I,O) = O$ (i.e., Party 1 is honest),

$$
\begin{array}{lll}
(\perp\,,\, C_2(y,\perp)) & \text{if } C_2(y) = \perp & (49)\\
(f_1(x,y)\,,\, C_2(y, f_2(x, C_2(y)))) & \text{otherwise} & (50)
\end{array}
$$

[39]This asymmetry is due to the non-concurrent nature of communication in the model. Since we postulate that the trusted party sends the answer first to the first party, the first party (but not the second) has the option to stop the third party after obtaining its part of the output. The second party, can only stop the third party before obtaining its output, but this is the same as refusing to participate.

[40]In the definition, the circuits C_1 and C_2 represent all possible actions in the model. In particular, $C_1(x) = \perp$ represents a decision of Party 1 not to enter the protocol at all. In this case $C_1(x,\perp)$ represents its local-output. The case $C_1(x) \neq \perp$, represents a decision to hand an input, denoted $C_1(x)$, to the trusted party. Likewise, $C_1(x,z)$ and $C_1(x,z,\perp)$, where z is the answer supplied by the trusted party, represents the actions taken by Party 1 after receiving the trusted party answer.

Equation (46) represents the case where Party 1 aborts before invoking the trusted party (and outputs a string which only depends on its input; i.e., x). Equation (47) represents the case where Party 1 invokes the trusted party with a possibly substituted input, denoted $C_1(x)$, and aborts while stopping the trusted party right after obtaining the output, $f_1(C_1(x), y)$. In this case the output of Party 1 depends on both its input and the output it has obtained from the trusted party. In both these cases, Party 2 obtains no output (from the trusted party). Equation (48) represents the case where Party 1 invokes the trusted party with a possibly substituted input, and allows the trusted party to answer to both parties (i.e., 1 and 2). In this case, the trusted party computes $f(C_1(x), y)$, and Party 1 outputs a string which depends on both x and $f_1(C(x), y)$. Likewise, Equation (49) and Equation (50) represent malicious behavior of Party 2; however, in accordance to the above discussion, the trusted party first supplies output to Party 1 and so Party 2 does not have an option analogous to Equation (47).

Execution in the real model. We next consider the real model in which a real (two-party) protocol is executed (and there exist no trusted third parties). In this case, a malicious party may follow an arbitrary feasible strategy; that is, any strategy implementable by polynomial-size circuits. In particular, the malicious party may abort the execution at any point in time, and when this happens prematurely, the other party is left with no output. In analogy to the ideal case, we use circuits to define strategies in a protocol.

Definition A.2 (malicious adversaries, the real model): *Let f be as in Definition A.1, and Π be a two-party protocol for computing f. Let $\overline{C} = (C_1, C_2)$ be a pair of polynomial-size circuit families representing adversaries in the real model. Such a pair is* admissible (w.r.t Π) (for the real malicious model) *if at least one C_i coincides with the strategy specified by Π. The* joint execution of Π under $\overline{C}$ in the real model (on input pair (x, y)), *denoted $\mathrm{real}_{\Pi,\overline{C}}(x, y)$, is defined as the output pair resulting of the interaction between $C_1(x)$ and $C_2(y)$.*

e assume that the circuit representing the real-model adversary (i.e., the C_i which does not follow Π) is deterministic. This is justified by standard techniques.

Security as emulation of real execution in the ideal model. Having defined the ideal and real models, we obtain the corresponding definition of security. Loosely speaking, the definition asserts that a secure two-party protocol (in the real model) emulates the ideal model (in which a trusted party exists). This is formulated by saying that admissible adversaries in the ideal-model are able to simulate (in the ideal-model) the execution of a secure real-model protocol (with admissible adversaries).

Definition A.3 (security in the malicious model): *Let f and Π be as in Definition A.2, Protocol Π is said to* securely compute f (in the malicious model) *if there exists a polynomial-time computable transformation of pairs of admissible polynomial-size circuit families $\overline{A} = (A_1, A_2)$ for the real model* (of Definition A.2) *into pairs of admissible polynomial-size circuit families $\overline{B} = (B_1, B_2)$ for the ideal model* (of Definition A.1) *so that*

$$\{\mathrm{ideal}_{f,\overline{B}}(x, y)\}_{x,y \text{ s.t. } |x|=|y|} \stackrel{\mathrm{c}}{\equiv} \{\mathrm{real}_{\Pi,\overline{A}}(x, y)\}_{x,y \text{ s.t. } |x|=|y|}$$

Implicit in Definition A.3 is a requirement that in a non-aborting (real) execution of a secure protocol, each party "knows" the value of the corresponding input on which the output is obtained. This is implied by the equivalence to the ideal model, in which the party explicitly hands the (possibly modified) input to the trusted party. For example, say Party 1 uses the malicious strategy A_1 and that $\mathrm{real}_{\Pi,\overline{A}}(x,y)$ is non-aborting. Then the output values correspond to the input pair $(B_1(x),y)$, where B_1 is the ideal-model adversary derived from the real-model adversarial strategy A_1.

Secrecy and correctness: By the above definition, the output of *both* parties together must be indistinguishable in the real and ideal models. The fact that the *adversarial* party's output is indistinguishable in both models formalizes the *secrecy* requirement of secure computation. That is, an adversary cannot learn more than what can be learned from his private input and output. On the other hand, the indistinguishability requirement on the *honest* party's output relates to the issue of *correctness*. Loosely speaking, the correctness requirement states that if a party is computing $f(x,y)$, then the adversary cannot cause him to receive $f'(x,y)$ for some $f' \neq f$. This is of course true in the ideal model as a trusted party computes f. Therefore the indistinguishability of the outputs means that it also holds in the real model (this is not to be confused with the adversary changing his own private input which is always possible). It is furthermore crucial that the secrecy and correctness requirements be intertwined, see [15, 25] for further discussion.

General plausibility results: Assuming the existence of trapdoor permutations, one may provide secure protocols for ANY two-party computation (allowing abort) [53], as well as for ANY multi-party computations with honest majority [30]. Thus, a host of cryptographic problems are solvable assuming the existence of trapdoor permutations. Specifically, any desired (input–output) functionality can be enforced, provided we are either willing to tolerate "early abort" (as defined above) or can rely on a majority of the parties to follow the protocol.

A.2 String Commitment

Commitment schemes are a basic ingredient in many cryptographic protocols. They are used to enable a party to commit itself to a value while keeping it secret. In a latter stage the commitment is "opened" and it is guaranteed that the "opening" can yield only a single value determined in the committing phase.

Loosely speaking, a commitment scheme is an efficient *two-phase* two-party protocol through which one party, called the *sender*, can commit itself to a *value* so that the following two conflicting requirements are satisfied.

1. *Secrecy* (or *hiding*): At the end of the first phase, the other party, called the *receiver*, does not gain any knowledge of the sender's value (this can be formalized analogously to the definition of indistinguishability of encryptions). This requirement has to be satisfied even if the receiver tries to cheat.

2. *Unambiguity* (or *binding*): Given the transcript of the interaction in the first phase, there exists at most one value that the receiver may later (i.e., in the second phase) accept as a legal "opening" of the commitment. This requirement has to be satisfied even if the sender tries to cheat.

The first phase is called the *commit phase*, and the second phase is called the *reveal phase*. Without loss of generality, the reveal phase may consist of merely letting the sender send, to the receiver, the original value and the sequence of random coin tosses that it has used during the commit phase.

The receiver will accept the value if and only if the supplied information matches its transcript of the interaction in the commit phase.

Our informal definition above describes a *perfectly binding* commitment scheme. That is, there exists only a single value that the receiver will accept as a decommitment. Therefore, even if the sender is computationally unlimited, he cannot cheat.

We now present a construction of a non-interactive, perfectly binding bit commitment using one-way permutations. Specifically, we use a one-way permutation, denoted f, and a hard-core predicate for it, denoted b. In fact, we may use any 1–1 one-way function.

1. *Commit Phase:* To commit to a bit $\tau \in \{0,1\}$, the sender uniformly selects $r \in \{0,1\}^n$ and sends the pair $(f(r), b(r) \oplus \tau)$.

2. *Reveal Phase:* The sender reveals the bit τ and the string r used in the commit phase. The receiver accepts τ if $f(r) = \alpha$ and $b(r) \oplus \tau = \beta$ where (α, β) is the receiver's view of the commit phase.

It is easy to see that this construction is a secure commitment scheme.

In order to commit to a *string* of n bits, $\tau = \tau_1 \cdots \tau_n$, the sender simply commits to each τ_i separately as above. We denote the commitment by $\mathrm{Commit}(\tau) = C(\tau, r)$ where the randomness used by the sender is $r = r_1, \ldots, r_n$ ($\forall i\ r_i \in_R \{0,1\}^n$).

A.3 Non-Malleable String Commitment

Loosely speaking, a non-malleable string commitment scheme is a commitment scheme with the additional requirement that given a commitment, it is infeasible to generate a commitment to a related value. We note that the commitment scheme presented in Section A.2 is easily malleable.[41] The concept of non-malleability was introduced by Dolev et al. in [21], where they also provide a perfectly binding, (interactive) non-malleable commitment scheme based on any one-way function.

We now present an informal definition of a non-malleable commitment scheme. Let $\mathcal{A}$ be an adversary who plays the receiver in a commitment protocol with a sender S. Furthermore, $\mathcal{A}$ *concurrently* plays the sender in a commitment protocol with a receiver T (one can look at S and T as executing a commitment protocol, with $\mathcal{A}$ playing a man-in-the-middle attack). To be more exact, consider the following experiment. Let $\mathcal{D}$ be some distribution of strings from which the values being committed to are chosen. In the experiment, the sender S chooses $\alpha \in_R D$ and commits to α in an execution of the commitment protocol with $\mathcal{A}$ as the receiver. Concurrently, the adversary $\mathcal{A}$ plays the sender in a commitment protocol with T as the receiver. We denote by β the value committed to by $\mathcal{A}$ in the execution between $\mathcal{A}$ and T. The adversary $\mathcal{A}$'s aim is to succeed in having its committed value β be *related* to α ($\mathcal{A}$ is not considered to have succeeded if $\beta = \alpha$; that is, copying is not ruled out). Thus, for a given polynomial-time computable relation R, we denote by $\Pi(\mathcal{A}, R)$, the probability that $\mathcal{A}$'s commitment is to a string β such that $(\alpha, \beta) \in R$. That is, $\Pi(\mathcal{A}, R)$ denotes the probability that $\mathcal{A}$ succeeds in generating a commitment that is related (by the relation R) to the commitment sent by S.

On the other hand, we consider another experiment involving an adversarial simulator $\mathcal{A}'$ who *does not* participate as the receiver in a commitment protocol with S. Rather, $\mathcal{A}'$ sends T a

[41] The malleability of the commitment scheme of Section A.2 can be seen as follows. Let (y, b) be a commitment to some bit τ (i.e., $y = f(r)$ for some string r, and $b(r) \oplus b = \tau$). Then, given this commitment, it is easy to generate a commitment to $\overline{\tau}$ by defining $C(\overline{\tau}) = (y, 1-b)$. We stress that this can be done without any knowledge whatsoever of the value of τ itself.

commitment to β and we denote by $\Pi'(\mathcal{A}', R)$ the probability that $(\alpha, \beta) \in R$ for $\alpha \in_R D$. That is, $\Pi(\mathcal{A}', R)$ denotes the a priori probability that a related commitment can be generated. We stress that $\mathcal{A}'$ must generate a "related" commitment without seeing any commitment to α.

We say that a string commitment scheme is non-malleable if for every ppt samplable distribution D, every polynomial-time relation R and every adversary $\mathcal{A}$, there exists an adversarial simulator $\mathcal{A}'$ such that $|\Pi(\mathcal{A}, R) - \Pi'(\mathcal{A}', R)|$ is negligible. Intuitively, this implies that the fact that $\mathcal{A}$ receives a commitment to α does not noticeably help it in generating a commitment to a related β. This formalization is conceptually similar to that of semantic security for encryptions (that states that the ciphertext itself does not help in learning any function of the plaintext).

A.4 The Zero-Knowledge Proof of Richardson and Kilian

We first review the notion of zero-knowledge. Loosely speaking, zero-knowledge proofs are proofs which yield nothing beyond the validity of the assertion. That is, a verifier obtaining such a proof only gains conviction in the validity of the assertion. Using the simulation paradigm this requirement is stated by postulating that anything that is feasibly computable from a zero-knowledge proof is also feasibly computable from the valid assertion alone.

The above informal paragraph refers to proofs as to interactive and randomized processes. That is, here a proof is a (multi-round) protocol for two parties, call verifier and prover, in which the prover wishes to convince the verifier of the validity of a given assertion. Such an *interactive proof* should allow the prover to convince the verifier of the validity of any true assertion, whereas NO prover strategy may fool the verifier to accept false assertions. Both the above *completeness* and *soundness* conditions should hold with high probability (i.e., a negligible error probability is allowed). The prescribed verifier strategy is required to be efficient. See [26] for a formal definition of interactive proofs. Zero-knowledge is a property of some prover strategies. More generally, we consider interactive machines which yield no knowledge while interacting with an arbitrary feasible (i.e., probabilistic polynomial-time) adversary on a common input taken from a predetermined set (in our case the set of valid assertions).

Definition A.4 (zero-knowledge [32]): *A strategy P is* zero-knowledge *on inputs from S if, for every feasible strategy V^*, there exists a feasible computation M^* so that the following two probability ensembles are computationally indistinguishable:*

1. $\{(P, V^*)(x)\}_{x \in S} \stackrel{\text{def}}{=}$ *the output of V^* when interacting with P on common input $x \in S$; and*
2. $\{M^*(x)\}_{x \in S} \stackrel{\text{def}}{=}$ *the output of M^* on input $x \in S$.*

A general plausibility result [29]: Assuming the existence of commitment schemes, there exist zero-knowledge proofs for membership in any $\mathcal{NP}$-language. Furthermore, the prescribed prover strategy is *efficient* provided it is given an $\mathcal{NP}$-witness to the assertion that is proven.

The protocol of Richardson and Kilian [47]

We actually simplify their presentation in a way that suffices for our own purposes. In essence, the protocol consists of two parts. In *the first part*, which is independent of the actual common input, m instances of *coin tossing into the well* [10] are sequentially executed where m is a parameter (to be discussed below). Specifically, the first part consists of m iterations, where the i^{th} iteration proceeds as follows: The verifier uniformly selects $v_i \in \{0,1\}^n$, and commits to it using a perfectly hiding commitment scheme. Next, the prover selects $p_i \in_R \{0,1\}^n$, and sends a perfectly binding commitment to it. Finally, the verifier decommits to v_i. (The result of the i^{th} coin-toss is defined as $v_i \oplus p_i$ and is known only to the prover.)

In *the second part*, the prover provides a witness indistinguishable (WI) proof [22] that either the common input is in the language or one of the outcomes of the m coin-tosses is the all-zero string (i.e., $v_i = p_i$ for some i). Intuitively, since the latter case is unlikely to happen in an actual execution of the protocol, the protocol constitutes a proof system for the language. However, the latter case is the key to the simulation of the protocol in the concurrent zero-knowledge model. We utilize this in our setting as well, when setting m to be equal to the total number of rounds in our own protocol (not including this subprotocol) *plus* any non-constant function of the security parameter n. The underlying idea is that whenever the simulator may cause $v_i = p_i$ to happen for some i, it can simulate the rest of the protocol (and specifically Part 2) by merely running the WI proof system with v_i (and the prover's coins) as a witness. (By the WI property, such an execution will be indistinguishable from an execution in which an NP-witness for the membership of the common input (in the language) is used.)

We remark that *any* perfectly hiding commitment scheme can be used in the first part of the proof. In particular, one can use an $O(n)$-round scheme based on any one-way permutation [43]. Alternatively, 2-round schemes exist under the (seemingly stronger) assumption of the existence of collections of claw-free functions with efficiently recognizable index sets [26].

A.5 Seed-Committed Pseudorandom Generators

A seed-committed pseudorandom generator is an efficiently computable deterministic function G mapping a seed to a (sequence,commitment) pair that fulfills the following conditions:

- The sequence is pseudorandom, even given the commitment.
- The partial mapping of the seed to the commitment is 1–1.

We use the following implementation ([12, 11]) of a seed-committed generator. Let f be a 1–1 one-way function and b a hard-core of f. Then define

$$G(s) = \langle b(s)b(f(s))\cdots b(f^{2n-1}(s)), f^{2n}(s)\rangle$$

This generator clearly fulfills the requirements: $b(s)\cdots b(f^{2n-1}(s))$ is the sequence and $f^{2n}(s)$ is the commitment.

We note that the following naive implementation does *not* work. Let G be any pseudorandom generator and consider the seed as a pair (s,r). Then define the mapping $(s,r) \mapsto (G(s), C(s,r))$ where $C(s,r)$ is a commitment to s using randomness r. It is true that the sequence is pseudorandom given the commitment. Furthermore, for every $s \neq s'$ and for every r, r' we have that $C(s,r) \neq C(s',r')$. However, there may be an s and $r \neq r'$ for which $C(s,r) = C(s,r')$ and therefore the mapping of the seed to the commitment is not necessarily 1–1.

A.6 Message Authentication Codes (MACs)

A Message Authentication Code, or MAC, enables parties A and B who share a joint secret key to achieve data integrity. That is, if B receives a message which is purportedly from A, then by verifying the MAC, B can be sure that A indeed sent the message and that it was not modified by any adversary on the way. A Message Authentication Scheme is comprised of the following algorithms:

1. A *Key Generation* algorithm that returns a secret key k.

2. A *Tagging* algorithm that given a key k and a message m, returns a tag $t = MAC_k(m)$.

3. A *Verification* algorithm that given a key k, a message m and a candidate tag t, returns a bit $b = \text{Verify}_k(m, t)$.

We now briefly, and informally, describe the security requirements of a MAC. Let $\mathcal{A}^{MAC_k(\cdot)}$ be a ppt adversary with oracle access to the tagging algorithm and let $m_1, \ldots, m_q$ be the list of $\mathcal{A}$'s oracle queries during her execution. Upon termination, $\mathcal{A}$ outputs a pair (m, t). We say that $\mathcal{A}$ *succeeds* if for every i, $m \neq m_i$ and furthermore $\text{Verify}_k(m, t) = 1$ (i.e., $\mathcal{A}$ generates a valid tag for a previously unseen message). Then, a MAC is secure if for every ppt machine $\mathcal{A}$, the probability that $\mathcal{A}$ succeeds is negligible.

This ensures integrity, because if an adversary modifies a message sent from A to B to one not previously seen, then B's verification will surely fail (there is an issue of replay attacks which we ignore here). The property that $\mathcal{A}$ cannot find an appropriate tag t for a "new" m, is called *unpredictability.*

It is easy to see that any pseudorandom function is a secure implementation of a MAC. This is because any random function is unpredictable and so any non-negligible success in generating t such that $f(m) = t$ (for an "unseen" m) distinguishes f from random.