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
Department of Computer Science
Weizmann Institute of Science
Rehovot, ISRAEL.
lindell@wisdom.weizmann.ac.il

December 6, 2000

Abstract

We present session-key generation protocols in a model where the legitimate parties share *only* a human-memorizable password. 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.

Area: Cryptography.

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

*Supported by MINERVA Foundation, Germany.

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* 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 (cf. [18, 41]). (However, even in this case, one may want to establish "session keys" as discussed below.)

Shared (high-quality) secret keys: By *high-quality keys* we mean strings coming from distribution of high min-entropy (e.g., uniformly chosen 56-bit (or rather 192-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 (cf., [11, 45, 24]).

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 $|\mathcal{D}| = \text{poly}(n)$.

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 $m \leq |\mathcal{D}|$ instances of the mutual authentication protocol, guessing a different password in each of them, then with probability $m/|\mathcal{D}|$ 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 $O(m/|\mathcal{D}|)+\mu(n)$, where m 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 *session-key generation*.

Main Result (informally stated): *Assuming the existence of trapdoor one-way permutations, there exists a session-key generation protocol that satisfies the following properties in the password-only setting*:

- *Key-match*: For any (probabilistic polynomial-time) adversary controlling the channel, the probability that the parties output different session-keys without detecting this fact is bounded above by $O(1/|\mathcal{D}|)$.
- *Session-key and password secrecy*: For any (probabilistic polynomial-time) adversary that tries to distinguish the session-key output by each party from a uniformly distributed n-bit string, the distinguishing gap (i.e., the difference in the probability that the adversary outputs 1 in the two cases) is at most $O(1/|\mathcal{D}|) + \mu(n)$, where $\mu(n)$ is a negligible function in the security parameter n. Similarly, the distinguishing gap between the party's password and a uniformly distributed element of $\mathcal{D}$ is at most $O(1/|\mathcal{D}|) + \mu(n)$.

Similar claims hold when m sessions (referring to the same password) are conducted sequentially, with $O(1/|\mathcal{D}|)$ being replaced by $O(m/|\mathcal{D}|)$. This holds also when a polynomial number of other sessions w.r.t independently distributed passwords are conducted concurrently to the above m sessions. Additional desirable properties of session-key protocols also hold:

- *Intrusion detection*: if the adversary modifies any message sent in a session then with probability at least $1 - O(1/|\mathcal{D}|)$ this is detected.
- *Forward secrecy*: The session-key maintains its security even if the password is revealed *after* the session-key was established.

[1] We stress that many famous key-exchange protocols, such as the one of Diffie and Hellman [18], refer to a passive adversary. In contrast, this paper refers to active adversaries.

- *Loss of Session-Keys*: The *current* session-key maintains its security even if *prior* session-keys are revealed. Furthermore, the password maintains its security even if all session-keys are revealed. (This is also known as security against a known-key attack.)
- *Improved security in presence of a passive adversary*: If the adversary is passive (i.e., does not omit, modify or insert messages) then both the legitimate parties end-up with the same uniformly distributed session-key. From the adversary's point of view (which includes the messages exchanged in this session) the session-key is computationally indistinguishable from a uniformly distributed n-bit string, and the parties' joint password is computationally indistinguishable from a uniformly distributed element of $\mathcal{D}$.

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 section 1.4.

1.2 Comparison to prior 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 *access control* mechanisms based only on passwords is widely recognized as a central problem of computer practice and has such has received much attention.

The first protocol suggested for password-based session-key generation, was by Bellovin and Merritt [6]. This work was very influential and became the basis for much future work in this area [7, 43, 31, 34, 39, 44]. However, these protocols have not been proven and their security is based on heuristics. 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 [35, 32] (a very brief survey can be found in [30]).

A first rigorous treatment of the access control problem was provided by Halevi and Krawczyk [30]. 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 access control 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 even in the access control setting. Finally, we stress

[2] In particular, a specific focus has been on preventing *off-line dictionary attacks*. In such an off-line attack, the adversary records his 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.

[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. (In the Halevi–Krawczyk protocol, the public-password is the hash-value of the server's public-key, where hashing is via a (universal) collision-intractable function.)

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 both of 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 access control mechanism (and authenticated 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 [1]. Secure (password-based) access control schemes in the random oracle model were presented in [5, 13]. 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 [16], 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) and 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).

Necessary conditions for mutual authentication: Halevi and Krawczyk [30] proved that mutual-authentication in the shared-password model implies (unauthenticated) secret-key exchange. Boyarsky [12] further pointed out that in the shared-password model, mutual-authentication implies Oblivious Transfer.[5]

1.3 Techniques

One central idea underlying our protocol is due to Naor and Pinkas [38]. They suggested the following protocol for the case of *passive* adversaries, based on a secure polynomial evaluation.[6] In order to generate a session-key, party A first chooses a linear polynomial $Q(\cdot)$, uniformly distributed over a large field. Next, A and B execute a polynomial evaluation in which B obtains $Q(w)$, where w is their joint password. The session-key is then set to equal $Q(w)$.

In [12] 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 abovementioned protocol of [38] by several additional mechanisms. Indeed, we use non-malleable commitments [19], but in addition we also use a *specific* zero-knowledge proof [40], ordinary commitment schemes [8], a *specific* pseudorandom generator (of [11, 45, 10]), and message authentication schemes (MACs). The analysis of the resulting protocol is very complicated, *even when the adversary initiates a single session.* As explained below, we believe that these

[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.

[5]Oblivious Transfer is known to imply (unauthenticated) secret-key exchange [33], but the other direction is not known to hold.

[6]In the polynomial evaluation functionality, parties A and B have a polynomial $Q(\cdot)$ and an element x for their respective inputs. The evaluation is such that A learns nothing, and B learns $Q(x)$ (i.e., the functionality is defined by $(Q, x) \mapsto (\lambda, Q(x))$).

complications are unavoidable given the current state-of-art regarding the *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).[7] 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 (say 6), 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 (since neither such adequate theorems nor such adequate protocols are known). 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. The adversary is then given these secrets enabling him to effectively emulate the second execution internally. Thus, only the first execution remains and the property can be directly proven. 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 protocols.

1.4 Discussion

The thrust of this work is in demonstrating the *feasibility* of performing *session-key generation based only on* (low-quality) *passwords*. Doing so, this work is merely the first step in a research project directed towards providing a good solution to this practical problem. We discuss three aspects of this project that require further study.

Concurrent executions: 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 password is never used in concurrent executions. A simple mechanism enforcing the above is to disallow a party to enter an execution with a particular password if less than Δ units of time have passed since last entering an execution with the same password. 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.

The definition of security: Our notion of session-key generation is stated in terms of a number of properties that capture the intuitive security goals. Thus, we provide protocols satisfying (at the

[7]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, and when these two executions refer to the same joint Alice–Bob password.

very least) a reasonable notion of security. Still, it is desirable to use "simulation-based definitions"; that is, definitions that require secure protocols to emulate functionalities defined in an appropriate ideal model (cf. [9, 36, 14]). We note that finding the "right" definitions for the goals of session-key generation is far beyond the scope of this work, and is an ongoing research project even in the simpler setting in which parties may share high-quality keys (and not merely low-quality passwords). The interested reader is referred to [2, 3, 4, 42].

Efficiency: It is indeed desirable to have more efficient protocols than the one suggested by us.

1.5 Organization

In Sections 2 and 3 we present the formal setting and our protocol for password-based session-key generation. Then, in Section 4 we present proof sketches of the main properties of our protocol. Following these sketches, we present the full proofs in Sections 5 to 10. The proof sketches are rather detailed and demonstrate our main techniques. Thus, we believe that a reading of the paper until the end of Section 4 is enough to obtain a good understanding of the results presented.

2 Formal Setting

In this section we present notation and definitions specific to our setting as well as a definition for Authenticated Session-Key Generation. Given these, we state our main result. A security parameter n is often implicit in our notations and discussions. We first present the following notations:

- Let C be the *channel* (probabilistic polynomial time adversary) through which parties A and B communicate. We adopt the notation of Bellare and Rogaway [2] and model the communication by giving C oracle access to A and B. We denote by $C^{A(x),B(y)}$, the output of C when he communicates with A and B, holding respective inputs x and y. We denote C's view in this execution by view $\left(C^{A(x),B(y)}\right)$.
- The password dictionary is denoted by $\mathcal{D} \subseteq \{0,1\}^n$. We denote $\epsilon = \frac{1}{|\mathcal{D}|}$.
- We denote by U_n the uniform distribution over 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.
- We denote an unspecified negligible function 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, we denote $f \approx g$ if $|f(n) - g(n)| < \mu(n)$.
- We denote computational indistinguishability by $\stackrel{c}{\equiv}$.

Some the definitions in Appendix A are presented in the non-uniform model of computation. A number of our proofs also seem to be in the non-uniform model, but can actually be carried out in the uniform model. Thus a naive reading of our proofs makes our main result hold assuming the existence of trapdoor permutations that cannot be inverted by polynomial size circuits. However, the same result can be achieved under the analogous uniform assumption.

2.1 $(1-\epsilon)$-indistinguishability and pseudorandomness

Extending the definition of computational indistinguishability, we define the concept of $(1-\epsilon)$-indistinguishability. Two ensembles are $(1-\epsilon)$-indistinguishable if for every ppt machine, the probability of distinguishing between them is at most negligibly greater than ϵ. Thus, computational indistinguishability coincides with 1-indistinguishability. The formal definition is as follows.

Definition 2.1 ($(1-\epsilon)$-indistinguishability): *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 *if for every probabilistic polynomial time distinguisher D, for every polynomial $p(\cdot)$, all sufficiently large n's and all auxiliary information $w\in\{0,1\}^{poly(n)}$*

$$|Pr[D(X_n,1^n,w)=1]-Pr[D(Y_n,1^n,w)=1]|<\epsilon+\frac{1}{p(n)}$$

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, 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 string 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, every polynomial $p(\cdot)$ and all sufficiently large n's*

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

2.2 Authenticated Session-Key Generation

We now define the requirements from an authenticated session-key generation protocol. Let C be an arbitrary ppt channel.

Definition 2.4 (Authenticated Session-Key Generation): *P is a secure protocol for authenticated session-key generation based on passwords if it fulfills the following requirements:*

- **Input:** *A and B share a secret $w\in_R\mathcal{D}$, where $\epsilon=1/|\mathcal{D}|$.*
- **Output:** *Each party outputs an n-bit string (called the* key*), denoted k_A and k_B respectively, and an* accept/reject *bit. The* accept/reject *bit is* public *(known to the adversary) and the keys are* private.
- **Requirements:**
  1. Viability: *If C is passive*[8]*, then $k_A=k_B$. Furthermore, with respect to C's view, k_A is pseudorandom and w is computationally indistinguishable from a random $\tilde{w}\in_R\mathcal{D}$.*

[8]That is, he does not modify, omit or insert any messages sent between A or B. However, he may attempt to learn secret information from the messages sent. Passive adversaries are also referred to as *semi-honest* in the literature.

The indistinguishability of w from $\tilde{w}$ is formally stated as follows. If C is passive, then the ensembles $\left\{\text{view}\left(C^{A(W),B(W)}\right), W\right\}$ *and* $\left\{\text{view}\left(C^{A(W),B(W)}\right), \widetilde{W}\right\}$ *are computationally indistinguishable, where $W, \widetilde{W}$ are independent and uniformly distributed in $\mathcal{D}$.*[9]

2. Key-Match: *The probability that both A and B output* accept *and yet $k_A \neq k_B$ is at most $O(\epsilon)$.*
3. Secrecy: *There are two secrecy conditions:*
   - Session-Key Secrecy: *At the conclusion of the protocol, for every ppt channel C, k_A and k_B are $(1 - O(\epsilon))$-pseudorandom with respect to C's view.*
   - Password Secrecy: *At the conclusion of the protocol, for every ppt channel C, w is $(1 - O(\epsilon))$-indistinguishable from $\tilde{w} \in_R \mathcal{D}$ with respect to C's view.*

The motivation behind our definition is as follows. The viability requirement refers to the case of a passive adversary and in this case we demand that nothing (significant) be learned of the password or the session-key. However, in the case of an *active* adversary such a level of security is impossible because the adversary can always guess the password correctly with probability ϵ (and can verify its guess by seeing if the parties accept). Therefore the session-key generated can only be $(1-\epsilon)$-pseudorandom, and no undesired events can be prevented with probability greater than $1-\epsilon$. Aside from this inherent limitation, a successful session-key generation execution must conclude with both parties having the same session-key (key-match). Furthermore, we wish the session-key generated to be $(1 - O(\epsilon))$-pseudorandom and that the protocol reveal no more than necessary about the password w (secrecy). We stress that the secrecy requirements hold even though the adversary is given the accept/reject bit. This formal requirement is necessary, since in practice this information can be implicitly understood from whether or not the parties continue communication after the session-key generation protocol has terminated.

We note that the above definition also enables mutual-authentication. This is because an adversary cannot cause a party to accept with a key that is not $(1 - O(\epsilon))$-pseudorandom to the adversary (session-key secrecy). As this key is secret, it can be used for explicit authentication via a (mutual) challenge/response protocol.[10] By augmenting the session-key protocol in such a way, we obtain explicit mutual-authentication.

Non-Uniform Distributions over $\mathcal{D}$: For simplicity, we assume that the parties share a uniformly chosen $w \in_R \mathcal{D}$. However, our proofs hold for any distribution over any dictionary $\mathcal{D}'$ so that no element occurs with probability greater than ϵ.

2.3 Our Main Result

Given Definition 2.4, we can now formally state our main result.

Theorem 2.5 *Assuming the existence of trapdoor permutations, there exist secure protocols for authenticated session-key generation based on passwords.*

[9] The pseudorandomness of k_A, as well as the secrecy requirements below, are formally defined in an analogous manner.

[10] 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 simple challenge/response protocol.

2.4 Multi-Session Security

The definition above relates to two parties executing a *single* session-key generation protocol. Clearly, we are interested in the more general case where many different parties run the protocol any number of times. In fact, it is enough to prove that a protocol is secure for a single invocation between two parties, by our definition, in order to show that it is secure in the multi-party and sequential multi-session case. In this section we briefly discuss this issue.

2.4.1 Many Invocations by Two Parties

Let A and B be parties who invoke m sequential executions of a session-key generation protocol. Given that we wish that an adversary gains no more than $O(1)$ password guesses upon each invocation, the security upon the m'th invocation should be $O(m\epsilon)$ (e.g., the session-key is $(1 - O(m\epsilon))$-pseudorandom and analogously for all other requirements). In Section 11 we prove that any secure session-key generation protocol maintains $O(m\epsilon)$ security after m invocations. Intuitively, this is due to the password-secrecy requirement which states that after a single invocation, the password is $(1 - O(\epsilon))$-indistinguishable from a random password. Then, any "success" greater than $O(m\epsilon)$ after m invocations can be reduced to learning more than $O(\epsilon)$ about the password in a single invocation.

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). We stress that a scenario whereby the adversary invokes B twice or more (sequentially) during a single execution with A is not allowed. 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). Note that parties do not usually need to initiate session-key generation protocols immediately one after the other. Therefore, this delay mechanism need only be employed when an attempted session-key generation execution fails. This means that parties not "under attack" by an adversary are not inconvenienced in any way.

We note that this limitation 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. This key may then be used as a shared, high-quality key for (concurrently) generating any polynomial number of $(1 - O(\epsilon))$-pseudorandom session-keys; one for each communication line (simple and efficient protocols exist for this task).

2.4.2 Many Parties

We now briefly discuss a generalization to the case where many parties execute the session-key protocol simultaneously. This includes the case that the adversarial channel controls any number of the legitimate parties.[11] Specifically, we claim that for m invocations of the protocol (which must be sequential for the same pair of parties and may be concurrent otherwise), the security is $O(m\epsilon)$.

We show this in the case of m different pairs, each pair executing a single invocation (the general case is similar). Consider m pairs of parties $(A_1, B_1), \ldots, (A_m, B_m)$ such that each pair shares a

[11] The importance of this extension was pointed out by Boyarsky [12].

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.) We first focus on the security of a pair of parties (A_i, B_i) when i is *fixed*. It is clear that the $O(\epsilon)$ security holds because C can internally simulate all other executions by choosing $w_j \in_R \mathcal{D}$ for every $j \neq i$, and we obtain a reduction to the single-pair case. The same argument holds regarding the security of a random pair (A_i, B_i), where $i \in_R \{1, \ldots, m\}$ is chosen randomly before the execution begins.

In the general case, we wish to analyze the security where i is not fixed or chosen at random ahead of time. Now, assume that there exists an adversary C and an index $j \in \{1, \ldots, m\}$, such that C succeeds with respect to (A_j, B_j) with probability greater than $O(m\epsilon)$. Then, C can be used to contradict the $O(\epsilon)$ security in the case that i is randomly chosen. This is because with probability $1/m$ we have that $i = j$ and therefore C succeeds with probability greater than $O(\frac{1}{m} \cdot m\epsilon) = O(\epsilon)$. We conclude that for every j, the security with respect to the (A_j, B_j) execution is $O(m\epsilon)$.

3 Our Session-Key Generation Protocol

All arithmetic below is over the finite field $\mathrm{GF}(2^n)$ 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*. This protocol involves two parties A and B; A has a non-constant, linear polynomial $Q(\cdot) \in \{0,1\}^{2n}$ and B has a string $x \in \{0,1\}^n$. The functionality is $(Q, x) \mapsto (\lambda, Q(x))$; that is, A learns nothing and B learns the value $Q(x)$ (and nothing else). The fact 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 protocol by having A also input a commitment to the polynomial $c_A \in \mathrm{Commit}(Q)$ and its corresponding decommitment r (i.e., $c_A = C(Q, r)$). Furthermore, B also inputs a commitment value c_B. This augmentation is needed 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):

- **Input:** *A inputs a commitment c_A and its corresponding decommitment r, and a linear, non-constant polynomial Q. B inputs a commitment c_B and a value x.*
- **Output:**
  1. Correct Input Case: *If $c_A = c_B$ and $c_A = C(Q, r)$, then B receives $Q(x)$ (and A receives nothing).*
  2. Incorrect Input Case: *If $c_A \neq c_B$ or $c_A \neq C(Q, r)$, then B receives $\perp$ (and A receives nothing).*

We note that by [46, 27], this functionality can be securely computed (A provides the decommitment and so the input conditions can be checked in polynomial time).

3.1 The Protocol

Let f be a one-way permutation and b a hard-core of f.

Protocol 1 (Session-Key Generation Protocol)

- **Input:** A and B begin with a joint password w, which is supposed to be uniformly distributed in $\mathcal{D}$.
- **Output:** A and B output an accept/reject bit as well as session-keys k_A and k_B respectively (k_A "should" equal k_B).
- **The Protocol:**
  1. **Stage 1: Commit**
     (a) A chooses a random, linear, non-constant polynomial Q over $\mathrm{GF}(2^n)$.
     (b) A and B engage in a *non-malleable* 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 by r_B and denote B's view of the execution of the commitment protocol by $NMC(Q, w)$.[12]
     Following the commitment protocol, B sends his random coins r_B to A. This has no effect on the security since the commitment protocol has already terminated.
  2. **Stage 2: Pre-Key Exchange** - In this stage the parties "exchange" strings τ_A and τ_B, from which the output session-keys (as well as validation checks) are *derived*. Thus, τ_A and τ_B are called pre-keys.
     (a) A sends B a commitment $c = \mathrm{Commit}(Q) = C(Q, r)$ for a random r.
     (b) A and B engage in an augmented polynomial evaluation protocol. A inputs Q and (c, r); B inputs w and c.
     (c) We denote B's output by τ_B. (Note that τ_B "should" equal $Q(w)$.)
     (d) A internally computes $\tau_A = Q(w)$.
  3. **Stage 3: Validation**
     (a) A sends the string $y = f^{2n}(\tau_A)$ to B.
     (b) A proves to B in zero-knowledge that she input the same polynomial in both the non-malleable and ordinary commitments, and that the value y is "consistent" with the non-malleable commitment. Formally, A proves the following statement:

     There exists a string $(X_1, x_2) \in \{0,1\}^{3n}$ and random coins $r_{A,1}, r_{A,2}$ (where $r_{A,1}$ and $r_{A,2}$ are A's random coins in the non-malleable and ordinary commitments respectively) such that
       i. B's view of the non-malleable commitment, $NMC(Q, w)$, 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 is the string of B's random coins in the non-malleable commitment.)[13]
       ii. $c = C(X_1, r_{A,2})$, and
       iii. $y = f^{2n}(X_1(x_2))$.

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

[13] The view of a protocol execution is a function of the parties' respective inputs and random strings. Therefore, (X_1, x_2), $r_{A,1}$ and r_B define a single possible view. Furthermore, 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 $r_{A,1}$, enabling him to efficiently prove the statement.

The specific zero-knowledge proof of Richardson and Kilian [40] is used here, with a specific setting of parameters; see Appendix A.4.[14]

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

4. **Decision Stage**

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

(b) B accepts if and only if all the following conditions are fulfilled:

- $y = f^{2n}(\tau_B)$ where y is the string sent by A to B in step 3(a) above and τ_B is B's output from the polynomial evaluation.
  (Note that if $\tau_B = \perp$ then no string fulfills this equality and B always rejects.)
- he accepts the zero-knowledge proof in step 3(b) above, and
- $\text{Verify}_{k_1(\tau_B)}(t_B, m) = 1$, where t_B is the session-transcript as seen by B, the string m is the alleged-MAC value that B receives, and $k_1(\tau_B) = b(\tau_B)\cdots b(f^{n-1}(\tau_B))$ is the MAC key used in verification.

If B accepts, then he locally outputs $k_2(\tau_B) = b(f^n(\tau_B))\cdots b(f^{2n-1}(\tau_B))$, otherwise he outputs a random string. (Recall that the accept/reject decision bit is considered a public output.)

It is imperative 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 related only to parties A and B. That is, we have ignored the existence of the channel C. Therefore, when A sends a string y to B, we "pretend" that B actually received y and not something else. In a real execution, this may not be the case at all. In the future we will therefore subscript every value by its owner, as we have denoted τ_A and τ_B in the protocol. For example, A sends a string y_A and we denote the string received by B by y_B.

3.2 Motivation for the Protocol

We now briefly motivate the design principle and structure of our protocol. The core of the session-key generation is the polynomial evaluation. In the case of a passive channel, executing a secure polynomial evaluation with a random, linear polynomial is a satisfactory protocol. That is, A chooses a random Q and inputs it to a polynomial evaluation with B who inputs w and receives $Q(w)$. Party A then internally computes $Q(w)$ 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, the protocol reveals nothing of w and $Q(w)$ to a passive C.

Attacks by an Adversarial Channel: In our case the channel can alter messages, causing B to receive $Q'(w) \neq Q(w)$. This clearly contradicts the *key-match* requirement (see Definition 2.4), but there are also strategies for which C can contradict the *session-key secrecy* requirement as well. For example, C can execute the polynomial evaluation with B, inputting a polynomial Q'

[14] The setting of parameters referred to relates to the number of iterations m in the first part of the Richardson-Kilian proof. We set m to equal the number of rounds in all other parts of our protocol plus $t(n)$, where $t(\cdot)$ is any non-constant function.

that he chooses (independent of the execution with A). In this case, B's output key is clearly not $(1-O(\epsilon))$-pseudorandom (unless $\mathcal{D}$ is super-polynomial in size). C may distinguish $Q'(w)$ from a uniformly distributed string by scanning the entire dictionary $\mathcal{D}$ and comparing $Q'(w')$ for all $w' \in \mathcal{D}$ to the challenge. Note that this is feasible here since $\mathcal{D}$ may be polynomial in size and C knows Q'.

"Controlling" the Channel: The commit and validation stages ensure that C cannot "modify" Q to $Q' \neq Q$ and have B accept. In the commit stage, B is supposed to receive a commitment to (Q, w). Loosely speaking, the zero-knowledge proof and B's check that $y_B = f^{2n}(\tau_B)$, ensure that if the commitment is to (Q', w') where $w' \neq w$, then B will reject. However, the secrecy of an *ordinary* commitment scheme does *not* prevent C from generating a commitment to (Q', w) based on A's commitment to (Q, w) (even though w is secret and unknown to C). It is for this reason that we use non-malleable commitments (see Section A.3). Thus, if C modifies the commitment sent by A, the probability that it is of the form (Q', w) (for $Q' \neq Q$) is at most negligibly greater than ϵ.

On the other hand, if C does not modify the commitment sent by A, then the validation stage ensures that B will reject unless $\tau_B = Q(w) = \tau_A$ (as desired).

Achieving Password-Secrecy: The MAC sent by A in the validation stage is needed in order to ensure secrecy of the password after the protocol. Otherwise, C can learn information about w based on whether B accepts or rejects. In fact, if we modify our protocol so that no MAC is sent, then we have an attack on the resulting protocol that enables C to learn one bit of w in every invocation (and therefore possibly learn w after only $\log|\mathcal{D}|$ invocations). See Appendix B for a description of the attack. By including the MAC, the channel C can itself predict whether B accepts or rejects based on whether or not C modified any messages sent between A and B. This means that no additional information is revealed by B's accept bit.

It is interesting to note that due to the necessity of the MAC, we do not know how to solve the seemingly simpler problem of mutual authentication without first generating a session-key.

We now explain the motivation behind some specific choices we made in designing the protocol.

Using the 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 we discuss in Appendix A.5, this is a seed-committing pseudorandom generator (i.e., $f^{2n}(s)$ uniquely determines s). The string τ_B received by B from the polynomial evaluation is used both for validation and for deriving the session-key. As part of the validation stage, some function F of τ_B is sent by A to B. The properties required from F are that firstly it be 1–1 (in order that it be effective for validation). Secondly, we require that pseudorandom keys for the MAC and output session-key may still be obtained, even though the adversary is given $F(\tau_B)$. Viewed in this light, using a seed-committed pseudorandom generator and taking $F(\cdot)$ as $f^{2n}(\cdot)$ 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. This is 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 τ uniquely determine w such that $Q(w) = \tau$.[15]

[15] 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, $\tau_B = Q'(w)$ is a fixed value and is thus KNOWN to C. Furthermore, C can execute the zero-knowledge proof in the validation stage correctly because $y = f^{2n}(Q'(w))$ is consistent with $NMC(Q', w')$ for *every* w'. We conclude that B accepts with a session-key known to C, in contradiction to the session-key secrecy requirement.

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

For technical reasons, B outputs a random string in the case that he rejects. This guarantees that his output string is always $(1 - O(\epsilon))$-pseudorandom.

3.3 Properties of Protocol 1

The main properties of Protocol 1 are captured by the following theorem.

Theorem 3.2 *Protocol 1 constitutes a secure protocol for authenticated session-key generation based on passwords (as defined in Definition 2.4).*

We further prove that Protocol 1 has a number of additional properties desirable for session-key generation. Specifically, we show that the protocol enables intrusion detection and that it maintains both "forward secrecy" and "security if prior session-keys are revealed" (see Section 4.7).

Protocol 1 as a feasibility result: All the cryptographic tools used in Protocol 1 can be securely implemented assuming the existence of trapdoor permutations. Thus, at the very least, Theorem 3.2 implies the feasibility result captured by Theorem 2.5.

Protocol 1 as a basis for efficient solutions: We now briefly discuss the efficiency of our protocol. The three main modules of the protocol are a non-malleable commitment, a secure polynomial evaluation and a zero-knowledge proof. The number of rounds of communication required for the zero-knowledge proof is m, where m equals the number of rounds in all other parts of the protocol *plus* some non-constant function in the security parameter (say $\log\log n$). In fact, as discussed in Section 7, this can be reduced to a single additional round assuming that expected polynomial-time simulation is sufficient. We thus conclude that the main bottleneck with respect to the number of rounds of communication is due to the non-malleable commitment and secure polynomial evaluation modules.

Current implementations for non-malleable commitment [19] and two-party computation [46, 27] require n rounds of communication.[16] (It is however remarked in [19] that the non-malleable commitment protocol can be improved to only $\log n$ rounds.) Therefore, any improvements in the efficiency of these modules would result in greater efficiency for our protocol. The same is true regarding the bandwidth, which is also large for currently known implementations of these modules. We note that if indeed efficient constructions are found for these modules, then our protocol may be used as an efficient solution for password-based session-key generation.

We comment that under a stronger set-up assumption that includes a common random string accessible by all parties (including the adversary), our protocol can be implemented more efficiently.[17] Specifically, the zero-knowledge proof could be replaced by a non-interactive proof improving the round complexity. Furthermore, it is conceivable that in the future, efficient protocols for non-malleable commitment in this model may exist.[18]

[16] Some researchers believe that Yao's protocol [46] can be implemented in a constant number of rounds.

[17] We note that it is not clear that the problem of password-based session-key generation is significantly easier in this common random string model.

[18] Efficient non-malleable commitment schemes have been shown in the common random string model [17, 21]. However, the definition of non-malleability in [17, 21] is weaker than that of [19]: non-malleability is guaranteed only if the adversary is also able to decommit (see [21]). In our protocol, the commitments are never opened and therefore the stronger definition of [19] is required.

4 Overview and Partial Proofs

Recall that C may omit, insert and modify any message sent between A and B. Thus, in actuality 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.[19] 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 our proofs. In fact, some of our proofs are clearly wasteful in this sense and the results are not tight. Our main objective is to make our (regrettably complex) proofs as modular and simple as possible.

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 visa 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 his current view and with some probability). This is in contrast to a passive adversary who by definition only eavesdrops on the communication.

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

We begin by proving the viability requirement (in Sections 4.2 and 5). Then, in Section 4.3 (and Section 6) we prove that $Q(w)$ is $(1 - O(\epsilon))$-pseudorandom at the conclusion of the Pre-Key Exchange stage between A and C. This is a central results towards proving that Protocol 1 is secure by Definition 2.4. Using this result, we prove the session-key secrecy requirement (Sections 4.4 and 8 with preliminaries from Section 7), the key-match requirement (Sections 4.5 and 9) and the password-secrecy requirement (Sections 4.6 and 10). Finally, in Section 4.7 we show additional properties that hold for our protocol.

[19] We are aware of work in progress for concurrent, honest-majority computation [15]. However, this does not apply to the two-party case.

In Section 7 we show how (and under what circumstances) A's zero-knowledge proof can be simulated in our concurrent setting, proving that C learns nothing from this proof in the protocol. We do not know how to show that C learns nothing when using regular zero-knowledge proofs, rather than the *specific* proof of Richardson and Kilian. Furthermore, the "zero-knowledge property" of the proof in our setting is *not* derived merely from the fact that the Richardson and Kilian proof provides concurrent zero-knowledge. Concurrent zero-knowledge only refers to a setting where many instances of the *same proof* are run concurrently, but says nothing when the proof is run concurrently with other protocols (as occurs in our case).

4.1 Formalizing the Setting

In this subsection, we present formal notations for the setting in which A and B interact via the channel C. In order to measure what C has learned from a protocol execution, we consider the following mental experiment that works in two stages. First, C invokes protocol executions with A and B. Then, following these executions, C receives some "challenge" string. This string may, for example, be either the session-key output by A and B or a random string. Then, C's inability to distinguish these cases points to the pseudorandomness of the output session-key.

For this experiment, we separate C into two parts, C_1 and C_2. The channel C_1 interacts concurrently with A and B for the entire protocol (or parts of it). At the conclusion of this interaction, C_1 outputs a string s representing its state information. Then, C_2 (who plays the role of the distinguisher and outputs a single bit) is given s and the challenge z. For example, in order to analyze C's capability of distinguishing the session-key from a random string, we consider the difference in the probability that C_2 outputs 1 when z equals the session-key and when z is a randomly chosen string.

Formally, denote by $C_1^{A(Q,w),B(w)}(1^n)$ the setting where the channel C instigates a protocol execution with parties A and B, who have respective inputs (Q, w) and w. Recall that A's input in the protocol is defined to be only w. However, modifying A so that she receives a random Q as additional input makes no difference to the outcome (recall that Q is chosen randomly by A in the first step of the protocol). This modification is made for the sake of the analysis and enables us to refer explicitly to Q when, for example, talking about the pseudorandomness of $Q(w)$.
Consider the following experiment for a fixed polynomial Q, password w and string z:

$$\begin{array}{l}\mathrm{Expt}_z^{A(Q,w),B(w)}(C): \\ \quad s \leftarrow C_1^{A(Q,w),B(w)}(1^n) \\ \quad \textsf{return } C_2(s,z)\end{array}$$

Then, for example, in order to analyze C's capability of distinguishing between the output session-key $k_2(Q(w))$ and a random string, we bound the following difference:

$$\left| Pr_{Q,w}[\mathrm{Expt}_{k_2(Q(w))}^{A(Q,w),B(w)}(C) = 1] - Pr_{Q,w}[\mathrm{Expt}_{U_n}^{A(Q,w),B(w)}(C) = 1] \right|$$

The above experiment template is used many times throughout the proof. In order to simplify notation, the experiment is sometimes written in a slightly modified form. For example, if Q plays no role in the experiment and C needs no explicit reference, then the experiment may be denoted only by $\mathrm{Expt}_z^{A(w),B(w)}$.

4.2 Viability

In this section, we state the viability requirement of the protocol. That is, if C is passive, then the result of a protocol execution is a joint session-key known to both parties, and this key is pseudorandom with respect to C's view. Furthermore, given C's view, the password w is indistinguishable from a random $\tilde{w} \in_R \mathcal{D}$. This means that when C is passive, effectively *nothing* is learned about the password or session-key from the protocol execution.

We note that the definitions of multi-party computation do not immediately imply that C cannot learn anything in our 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. The experiment referred to in the theorem is defined in Section 4.1.

Theorem 4.1 (Viability): *Let C be a passive channel. Then, both A and B accept and output the same session-key $k_2(Q(w))$. Furthermore, the session-key $k_2(Q(w))$ is pseudorandom and the password w is indistinguishable from $\tilde{w} \in_R \mathcal{D}$. That is, for every* passive *ppt channel C, for every polynomial $p(\cdot)$ and for all sufficiently large n's*

$$\left| Pr_{Q,w}[\mathrm{Expt}_{k_2(Q(w))}^{A(Q,w),B(w)}(C)=1] - Pr_{Q,w,U_n}[\mathrm{Expt}_{U_n}^{A(Q,w),B(w)}(C)=1] \right| < \frac{1}{p(n)}$$

and

$$\left| Pr_{Q,w}[\mathrm{Expt}_{w}^{A(Q,w),B(w)}(C)=1] - Pr_{Q,w,\tilde{w}}[\mathrm{Expt}_{\tilde{w}}^{A(Q,w),B(w)}(C)=1] \right| < \frac{1}{p(n)}$$

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

The proof can be found in Section 5. Since C is a passive adversary (in this case), the proof is rather straightforward.

4.3 Pseudorandomness of $Q(w)$

A central element of our proof is showing that the output keys are $(1-O(\epsilon))$-pseudorandom. We begin by proving that at the conclusion of *Stage (2)* of the protocol (pre-key exchange) between A and C, the string $Q(w)$ is $(1-2\epsilon)$-pseudorandom. From this, we derive both the security of the MAC-key $k_1(Q(w))$ and the output-key $k_2(Q(w))$, based on the properties of the seed-committed pseudorandom generator $G(s) \stackrel{\mathrm{def}}{=} k_1(s), k_2(s), f^{2n}(s)$.

We note that at the conclusion of the entire protocol, it is *not* true that the pre-key $Q(w)$ itself is $(1-O(\epsilon))$-pseudorandom. This is because in the Validation Stage, party A sends $y = f^{2n}(Q(w))$, which is seen by C. Then, let z be the challenge which is either $Q(w)$ or a random string. Since C is given z, he can easily distinguish the cases by comparing $f^{2n}(z)$ with y.

As in the mental experiment defined in Section 4.1, the channel C first runs a protocol execution with A and B. Following this, C is given a challenge z and should decide if $z = Q(w)$ or if z is a random string. However, the experiment must reflect the fact that we wish to analyze the pseudorandomness of $Q(w)$ at the conclusion of the (A, C) Pre-Key Exchange (rather than at the end of the entire protocol). We do this by *modifying* A to another party A_2 who only participates in the first two stages of Protocol 1. That is, A_2 halts immediately after the pre-key exchange stage. Then, we analyze the setting in which C interacts with A_2 and B (rather than A and B) and at the conclusion of *these* executions is given the challenge. Recall that we denote this setting by $C_1^{A_2(Q,w),B(w)}(1^n)$ and that the distinguisher C_2 receives C_1's state information and a challenge which is either $Q(w)$ or a random string.

Theorem 4.2 (Pseudorandomness of $Q(w)$): *Let C be an arbitrary ppt adversary interacting with A_2 and B. Then, for every polynomial $p(\cdot)$ and for all sufficiently large n's*

$$\left|Pr_{Q,w}[\mathrm{Expt}_{Q(w)}^{A_2(Q,w),B(w)}(C)=1] - Pr_{Q,w,U_n}[\mathrm{Expt}_{U_n}^{A_2(Q,w),B(w)}(C)=1]\right| < 2\epsilon + \frac{1}{p(n)}$$

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

Proof: The main body of the proof is found in Lemmas 4.3 and 4.4. The theorem states that C cannot distinguish between $Q(w)$ and U_n with any polynomial advantage over 2ϵ. In Lemma 4.3 we show that C cannot distinguish between U_n and $Q(\tilde{w})$ for a random $\tilde{w} \in_R \mathcal{D}$ with any polynomial advantage over ϵ. Then, Lemma 4.4 states that C also cannot distinguish between $Q(w)$ and $Q(\tilde{w})$ for an independent $\tilde{w} \in_R \mathcal{D}$ with any polynomial advantage over ϵ. Putting these two Lemmas together we have that C cannot distinguish between $Q(w)$ and U_n with any polynomial advantage over 2ϵ.

Lemma 4.3 *For every polynomial $p(\cdot)$ and for all sufficiently large n's*

$$\left|Pr_{Q,w,U_n}[\mathrm{Expt}_{U_n}^{A_2(Q,w),B(w)}(C)=1] - Pr_{Q,w,\tilde{w}}[\mathrm{Expt}_{Q(\tilde{w})}^{A_2(Q,w),B(w)}(C)=1]\right| < \epsilon + \frac{1}{p(n)}$$

Proof Sketch: The lemma holds *even* if C knows w itself. (Intuitively, C must distinguish between $Q(\tilde{w})$ and U_n, which is related to the secrecy of Q, not of w). Therefore, we can give C the password w, enabling him to internally emulate the entire session with B. (This emulation is perfect as knowledge of w is all that is needed to play B's role.) What remains is therefore a (non-concurrent) session with A_2 only, which can be analyzed in the stand-alone, two-party setting.

It is clear that the only place that C can learn about Q is from the polynomial evaluation itself (the commitments are indistinguishable and so reveal nothing). The security of the polynomial evaluation implies that the receiver can learn nothing beyond the value of $Q(\cdot)$ at a single point. Therefore, unless C inputs $\tilde{w}$ itself into the polynomial evaluation receiving $Q(\tilde{w})$, the values $Q(\tilde{w})$ and U_n are indistinguishable (recall that Q is a random, non-constant, linear polynomial and so we have "almost" pairwise independence). However, as $\tilde{w}$ is uniformly distributed in $\mathcal{D}$, the probability that C inputs $\tilde{w}$ is at most ϵ. This means that C can distinguish $Q(\tilde{w})$ from U_n with probability at most ϵ. $\Box$

(The full proof is presented in Section 6.1 and is derived in a straightforward manner from the sketch.)

Lemma 4.4 *For every polynomial $p(\cdot)$ and for all sufficiently large n's*

$$\left|Pr_{Q,w}[\mathrm{Expt}_{Q(w)}^{A_2(Q,w),B(w)}(C)=1] - Pr_{Q,w,\tilde{w}}[\mathrm{Expt}_{Q(\tilde{w})}^{A_2(Q,w),B(w)}(C)=1]\right| < \epsilon + \frac{1}{p(n)}$$

Proof Sketch: As in the previous lemma, we first remove the concurrency from the setting. The important points to notice here are as follows. Firstly, the lemma holds even if C knows the polynomial Q (and in such a case, distinguishing $Q(w)$ from $Q(\tilde{w})$ reduces to distinguishing w from $\tilde{w}$). Secondly, it is possible to emulate A_2 given *only* knowledge of Q (i.e., without knowing w). This is because the only input used by A_2 in the pre-key exchange is Q itself. It is true that A_2 uses w in the non-malleable commitment. However, due to the hiding property of the non-malleable commitment, this can also be simulated without knowing w. Therefore, C can simulate the entire session with A_2 internally. What remains is a single session between C and B which is in the standard two-party computation setting.

In the session between C and B, channel C's view consists of B's accept/reject bit only. Essentially, B accepts if during the validation stage he is convinced that C knows the value of w. This can be seen as follows. Let Q_C be the polynomial that C inputs into the polynomial evaluation. Then, B accepts only if he receives $y = f^{2n}(Q_C(w))$. However, since f^{2n} and Q_C are 1–1 functions, y defines a single possible value of w. This implies that B accepts only if C essentially guesses the value of w correctly. Since $w \in_R \mathcal{D}$ and C receives *nothing* from the interaction with B before sending y, the probability that B accepts is at most ϵ. We therefore have that with probability $1 - \epsilon$, B's output-bit can be correctly simulated (by simply guessing that it equals reject). This means that at most an "ϵ amount of information" can be learned from this bit (which is the only bit of information learned by C during the execution). $\square$

The full proof is presented in Section 6.2 and is not a direct implementation of the above ideas. The main difficulty involved is due to the fact that the use of the MAC in the validation stage of our protocol precludes the use of currently known composition theorems for secure computation (see Section 6.2 for details). We must therefore bypass this problem before analyzing C's probability of success.

The theorem follows immediately from Lemmas 4.3 and 4.4. $\blacksquare$

Intermediate Conclusion. At the end of Stage (2) of the protocol, A_2 has a uniformly distributed string $Q(w)$ that is $(1 - 2\epsilon)$-pseudorandom with respect to C. We stress that this says nothing of the string τ_B obtained by B from the polynomial evaluation. In fact, τ_B is not necessarily $(1 - 2\epsilon)$-pseudorandom at all. A strategy for C in which he explicitly chooses a polynomial Q' results in $\tau_B = Q'(w)$ where C knows Q'. Then, upon receiving the challenge z, the channel C can simply check for every $w \in \mathcal{D}$, if $Q'(w) = z$ (if $\mathcal{D}$ is not too large, then this is feasible) and C would distinguish $Q'(w)$ from U_n with overwhelming probability. However, as we will show, our protocol is such that the probability that B accepts and $\tau_B \neq Q(w)$ is small (see the key-match condition). Thus, we are assured that typically, either B rejects or τ_B is also a $(1 - 2\epsilon)$-pseudorandom string (which equals $Q(w)$).

4.4 Session-Key Secrecy

Session-Key Secrecy for A: The following theorem states that after the protocol execution, A's session-key is $(1 - O(\epsilon))$-pseudorandom with respect to C's view. Recall that $Q(w)$ is $(1 - O(\epsilon))$-pseudorandom after the first two stages of the protocol, and that the only messages sent by A in the third and final stage are $y = f^{2n}(Q(w))$, a zero-knowledge proof and a MAC of the message transcript. Intuitively, the zero-knowledge proof reveals nothing and since $Q(w)$ is $(1-O(\epsilon))$-pseudorandom after the Pre-Key Exchange, the session-key $k_2(Q(w))$ remains $(1-O(\epsilon))$-pseudorandom even *given* $y = f^{2n}(Q(w))$ and the MAC. This is due to the fact that y along with the MAC-key and the session-key constitutes a pseudorandom generator. Therefore, the proof of the theorem is derived from Theorem 4.2 and the properties of the seed-committed pseudorandom generator (see Appendix A.5).

The full proof of the fact that nothing is learned from A's zero-knowledge proof can be found in Section 7. We stress that this is not at all obvious and that we do not know whether it holds when using an ordinary zero-knowledge proof (rather than the Richardson-Kilian proof). In Section 8, we use Section 7 and the properties of the seed-committed generator to formally prove the following theorem for session-key secrecy. (The definition of the experiment in the theorem statement below can be found in Section 4.1.)

Theorem 4.5 (Session-Key Secrecy): *Let C be an arbitrary ppt channel. Then,*

$$\left|Pr_{Q,w}[\mathrm{Expt}_{k_2(Q(w))}^{A(Q,w),B(w)}=1]-Pr_{Q,w,U_n}[\mathrm{Expt}_{U_n}^{A(Q,w),B(w)}=1]\right|<4\epsilon+\frac{1}{poly(n)}$$

Session-Key Secrecy for B: B's session-key secrecy is shown by combining Theorem 4.5 (A's session-key secrecy) together with Theorem 4.6 proven below (the key-match requirement). Recall that B's session-key, denoted k_B, equals $k_2(\tau_B)$ in case B accepts and is a uniformly distributed string in case he rejects. Intuitively, k_B is $(1-O(\epsilon))$-pseudorandom because with probability $1-O(\epsilon)$, party B only accepts when $\tau_A=\tau_B$. However, in this case $k_B=k_2(\tau_A)$ and thus B's session-key secrecy is reduced to A's session-key secrecy.

Formally, in Theorem 4.6 we prove that the probability that B accepts and $\tau_A\neq\tau_B$ (or almost equivalently that $k_B\neq k_2(\tau_A)$) is at most negligibly more than 3ϵ. This implies that

$$\left|Pr_{Q,w}[\mathrm{Expt}_{k_B}^{A(Q,w),B(w)}=1\ \wedge\ B=\mathrm{acc}]-Pr_{Q,w,U_n}[\mathrm{Expt}_{k_2(\tau_A)}^{A(Q,w),B(w)}=1\ \wedge\ B=\mathrm{acc}]\right|<3\epsilon+\frac{1}{poly(n)}$$

(This can be seen by noticing that when $k_B=k_2(\tau_A)$, the experiments are identical. On the other hand, when $k_B\neq k_2(\tau_A)$, then both the probabilities in the difference are between 0 and $3\epsilon+\mu$. Thus the difference can be at most negligibly more than 3ϵ.) We therefore have that

$$\begin{aligned}&\left|Pr_{Q,w}[\mathrm{Expt}_{k_B}^{A(Q,w),B(w)}=1\ \wedge\ B=\mathrm{acc}]-Pr_{Q,w,U_n}[\mathrm{Expt}_{U_n}^{A(Q,w),B(w)}=1\ \wedge\ B=\mathrm{acc}]\right|\\ &<\ \left|Pr_{Q,w}[\mathrm{Expt}_{k_2(\tau_A)}^{A(Q,w),B(w)}=1\ \wedge\ B=\mathrm{acc}]-Pr_{Q,w,U_n}[\mathrm{Expt}_{U_n}^{A(Q,w),B(w)}=1\ \wedge\ B=\mathrm{acc}]\right| \quad (1)\\ &\quad+\ 3\epsilon+\frac{1}{poly(n)}\end{aligned}$$

Now, recall that B's accept bit is part of C's view. Therefore, any success in distinguishing $k_2(\tau_A)$ from a uniformly distributed string when B accepts, can be transformed into success in distinguishing $k_2(\tau_A)$ from U_n in the general case (i.e., where B may accept or reject). By the fact that $k_2(\tau_A)$ is $(1-4\epsilon)$-pseudorandom, we have that Equation (1) is upper bound $4\epsilon+\mu$. Thus when B accepts, his session-key k_B can be distinguished from a random string with probability at most $3\epsilon+4\epsilon+\mu$. On the other hand, in the case that B rejects, k_B is uniformly distributed and thus cannot be distinguished from a random string at all. Combining these facts, we have that

$$\left|Pr_{Q,w}[\mathrm{Expt}_{k_B}^{A(Q,w),B(w)}=1]-Pr_{Q,w,U_n}[\mathrm{Expt}_{U_n}^{A(Q,w),B(w)}=1]\right|<7\epsilon+\frac{1}{poly(n)}$$

4.5 The Key-Match Requirement

We now prove the *key-match* requirement which states that the probability that A and B both accept, yet have different keys is at most $O(\epsilon)$. Recall that $\tau_A \stackrel{\text{def}}{=} Q(w)$ and that τ_B is B's output from the polynomial evaluation.

Theorem 4.6 (Key-Match): *For every ppt adversarial channel C, every polynomial $p(\cdot)$ and all sufficiently large n's*

$$Pr[B = \text{acc} \ \wedge \ \tau_A \neq \tau_B] < 3\epsilon + \frac{1}{p(n)}$$

Proof Sketch: The proof is divided 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 states that a commitment is non-malleable when executed concurrently with another commitment.[20] 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 that the non-malleable property holds at all.

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, 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 *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.) In this case, when Q_C is independent of Q, the probability that the "y" value sent by C to B will match $f^{2n}(Q_C(w))$ is at most ϵ. This means that B will reject with probability $1 - \epsilon$. We call this case "unsynchronized".

In the other possible scheduling, C completes the polynomial evaluation with A **after** completing the non-malleable commitment with B (and so in this case the two executions are more synchronized). In this case we show how the (A, C) polynomial evaluation 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 4.6: What we prove is that according to each of the two scheduling cases, the probability that B accepts and there is a key *mismatch* is at most $O(\epsilon)$. Using the Union Bound, Theorem 4.6 follows.

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 $2\epsilon + \frac{1}{poly(n)}$ irrelevant of the values of τ_A and τ_B. This is enough because

$$Pr[B = \text{acc} \ \wedge \ \tau_A \neq \tau_B \ \wedge \ \text{Case 1}] \leq Pr[B = \text{acc} \ \wedge \ \text{Case 1}]$$

[20]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.

Lemma 4.7 (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*

$$Pr[B = \text{acc} \ \wedge \ \text{Case 1}] < 2\epsilon + \frac{1}{p(n)}$$

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, what we actually have is a two-party setting between C and P. As in previous proofs, we analyze what happens in this two-party setting and derive the result regarding our concurrent setting.

The actual reduction is more complex, as the (A, C) and (C, B) protocols involve other steps beyond the polynomial evaluation. Furthermore, some of these steps may be run concurrently (unlike the polynomial evaluations which are executed sequentially according to this scheduling). Therefore, the main difficulty in the proof is in defining the two-party protocol between C and P so that it correctly simulates the concurrent execution of our *entire* protocol. □

The full proof is presented in Section 9.1.

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** B accepts **and** $\tau_A \neq \tau_B$, is less than $\epsilon + \frac{1}{poly(n)}$.

Lemma 4.8 (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,*

$$Pr[B = \text{acc} \ \wedge \ \text{Case 2} \ \wedge \ \tau_A \neq \tau_B] < \epsilon + \frac{1}{p(n)}$$

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). Loosely speaking, this means that A's commitment does not help C in generating a commitment to a related pair. (This holds unless C simply copies A's commitment unmodified; however, then we can show that B rejects unless $\tau_A = \tau_B$, in which case key-match holds). Now, denote C's non-malleable commitment by (Q', w'). Then, we are interested in the probability that $Q' \neq Q$ and $w' = w$ (i.e., the second element in the pair is A and B's shared secret password).[21] Since A's commitment does not help

[21] As we have mentioned, in the case that C copies A's commitment unmodified and thus $(Q', w') = (Q, w)$, we can show that unless $\tau_A = \tau_B$, party B rejects with overwhelming probability. This is because the validation stage enforces that $\tau_B = Q'(w')$ and thus when $(Q', w') = (Q, w)$, we have that $\tau_B = Q'(w') = Q(w) = \tau_A$. In this sketch, we therefore only relate to the case that $(Q', w') \neq (Q, w)$.

C in generating this commitment and w is uniformly distributed in $\mathcal{D}$ with respect to C's view, the probability that C generates such a commitment is at most negligibly more than ϵ. If C indeed generates such a commitment, then he may cause B to accept, even when $\tau_A \neq \tau_B$. However, the probability that C succeeds in this is less than $\epsilon + \frac{1}{poly(n)}$.

On the other hand, if C fails to generate such a related commitment and B receives a non-malleable commitment to (Q', w') where $w' \neq w$, then the validation stage ensures that B will reject. This is because, essentially, the (C, B) validation stage enforces that B's output from the polynomial evaluation be consistent with the non-malleable commitment he received. That is, it ensures that B will reject unless he receives $\tau_B = Q'(w')$ from the polynomial evaluation. The validation stage also enforces that the polynomial input by C into the polynomial evaluation is Q'. That is, the respective inputs of C and B into the polynomial evaluation are Q' and w. By the correctness of the evaluation, it must hold that B receives $Q'(w)$. Therefore if B accepts it must be the case that $Q'(w') = Q'(w)$ which implies that $w' = w$. In other words, if C's commitment is such that $w' \neq w$, then B rejects with overwhelming probability. We conclude that B only accepts if C's (non-malleable) commitment was to (Q', w) and that this can occur with probability at most negligibly more than ϵ. $\square$

The full proof of Lemma 4.8 is presented in Section 9.2 and is significantly more involved than the above sketch.

Theorem 4.6 follows by combining Lemmas 4.7 and Lemma 4.8. ■

4.6 Password Secrecy

We now prove that at the conclusion of the protocol, the password w is $(1-O(\epsilon))$-indistinguishable from a random $\tilde{w} \in_R \mathcal{D}$.

Before beginning, we state below a corollary (proven at the end of Section 8), that relates to the security of the MAC. We note that, for simplicity, our proofs (throughout) refer to a MAC that is implemented by a pseudorandom function. However, all claims carry through for any implementation of a MAC. The corollary below states that it is "hard" for C to generate a correct MAC-value for any value other than that sent by A. Specifically, the function $MAC_{k_1(\tau_A)}(\cdot)$ is $(1-4\epsilon)$-pseudorandom. Therefore for any value t, unless t is the exact (A, C)-message-transcript (and thus A herself sends this value in the protocol), the probability that C generates a pair $(t, MAC_{k_1(\tau_A)}(t))$ is at most negligibly greater than 4ϵ.

The pseudorandomness of the MAC function is based on the fact that at the conclusion of the (A, C) pre-key exchange stage τ_A is $(1-2\epsilon)$-pseudorandom (Theorem 4.2) and that $k_1(\tau_A)$ is $(1-4\epsilon)$-pseudorandom, even given $y = f^{2n}(\tau_A)$ (recall that A sends y during the protocol). Intuitively, this is because $f^{2n}(\tau_A) \cdot k_1(\tau_A)$ is a $(1-O(\epsilon))$-pseudorandom sequence, and thus $k_1(\tau_A)$ remains $(1-O(\epsilon))$-pseudorandom even given $f^{2n}(\tau_A)$.

Corollary 4.9 *Let C be an arbitrary ppt channel. Then, for every string t that differs from the (A, C)-message-transcript, the value $MAC_{k_1(\tau_A)}(t)$ is $(1-4\epsilon)$-pseudorandom with respect to C's view.*

We now prove that for every ppt channel C, at the conclusion of the protocol, with respect to C's view the actual password is $(1-O(\epsilon))$-indistinguishable from a uniformly chosen (new) password.

Theorem 4.10 (Password Secrecy): *Let C be a ppt channel interacting with A and B. Then, for* *every polynomial $p(\cdot)$ and for all sufficiently large n's*

$$\left| Pr_w[\mathrm{Expt}_w^{A(w),B(w)}(C) = 1] - Pr_{w,\tilde{w}}[\mathrm{Expt}_{\tilde{w}}^{A(w),B(w)}(C) = 1] \right| < 26\epsilon + \frac{1}{p(n)}$$

where w and $\tilde{w}$ are independently and uniformly distributed in $\mathcal{D}$.

Proof Sketch: We prove this theorem by first "removing" the concurrent (C, B) execution. This is done in a way that affects C's capability of distinguishing w from $\tilde{w}$) by at most $O(\epsilon)$.

The (C, B) validation is first removed by showing that C can actually predict B's accept/reject bit itself (this is enough as this bit is B's only output from this stage). This is shown by combining the following facts:

- Theorem 4.6 states that the probability that B accepts and $\tau_A \neq \tau_B$ is at most $O(\epsilon)$.
- Let t_A and t_B be the (A, C) and (C, B) message-transcripts respectively. Then, Corollary 4.9 states that if $t_A \neq t_B$, then $MAC_{k_1(\tau_A)}(t_B)$ is $(1 - O(\epsilon))$-pseudorandom with respect to C's view.
- B only accepts if he receives $MAC_{k_1(\tau_B)}(t_B)$ (keyed by τ_B) in the last step of the protocol. Note that if $\tau_A = \tau_B$, then Corollary 4.9 holds regarding this MAC.

Putting these together, we have that if $\tau_A = \tau_B$ and C is not reliable, then B rejects with probability $1 - O(\epsilon)$. On the other hand, if $\tau_A \neq \tau_B$, then B anyway rejects with probability $1 - O(\epsilon)$ (key-match). Therefore, the probability that B accepts and $t_A \neq t_B$ is at most $O(\epsilon)$. On the other hand, if $t_A = t_B$ (i.e., C was reliable), then B certainly accepts. By noticing that C always knows whether $t_A = t_B$ or $t_A \neq t_B$ holds, we have that C can predict B's accept/reject bit itself by "guessing" that B accepts if and only if $t_A = t_B$. Since, channel C is wrong in this guess with probability at most $O(\epsilon)$, the difference in C's view in the case that C really receives B's output bit or guesses it himself, is at most $O(\epsilon)$.

Next, the remainder of the (C, B) evaluation is removed. Intuitively this is possible because (in the remaining first two stages) the only place that B uses w is in the (C, B) polynomial evaluation. However, C receives no output from this evaluation and thus nothing is revealed about w. We conclude that B's role in the remaining execution can be simulated using any $w' \in \mathcal{D}$ and this simulation is indistinguishable from a real execution.

We now remain with a non-concurrent setting involving only A and C. In this setting, we show that C can distinguish w from $\tilde{w} \in_R \mathcal{D}$ with probability at most $O(\epsilon)$. As before, the (A, C) validation stage is first removed by simulating the zero-knowledge proof given by A and next by noticing that both y and the MAC sent by A are $(1 - O(\epsilon))$-pseudorandom to C. Then, if C receives random strings instead, this can make a difference of at most $O(\epsilon)$. Therefore, A's part in the validation stage can be replaced by the zero-knowledge simulator and random strings. (As explained in Section 4.4 this simulation of A's proof is not immediate, see Section 7.)

We now remain with a protocol between A, restricted to the first two stages, and C. Now, notice that in the first two stages of the protocol, A only uses w in her non-malleable commitment. Thus, due to the hiding property of the commitment, w remains computationally indistinguishable from $\tilde{w} \in_R \mathcal{D}$. This completes the proof sketch. $\square$

The full proof is presented in Section 10.

4.7 Additional Properties

In this section, we show additional desirable properties of our protocol. Namely, we show that our protocol satisfies Intrusion Detection, Forward Secrecy and security in the face of Session-Key Loss (also known as a "Known-Key Attack").

Intrusion Detection: *If the adversary modifies any message sent in a session, then with probability at least* $(1 - O(\epsilon))$ *this is detected.*

This property is immediately derived from Claim 10.4 (see Section 10), that states that the probability that B accepts and C is not reliable is at most $O(\epsilon)$. This prevents C from carrying out any undetected active attack (where an active attack is one in which C is not reliable). However, it does not mean that C cannot learn about the password and session-key by only eavesdropping. (We have already shown that C can learn at most $O(\epsilon)$ in Theorems 4.5 and 4.10, however this may be significant if it can go undetected.)

We now show that in actuality, C can only "significantly learn something" about w by being *unreliable*. Therefore, we are ensured that a channel C cannot learn anything noticeable about w, without us detecting adversarial behavior with probability at least $1 - O(\epsilon)$. Recall that a passive channel is always reliable but the reverse is not true. Furthermore, an active channel may dynamically decide to be reliable or not, possibly depending on what occurs during the protocol execution. Despite this, the following claim states that in a given execution for which the channel is reliable, he can learn no more than if he was passive.

Claim 4.11 *For every ppt* active *channel C there exists a* passive *channel C' such that for every randomized process $z = Z(Q, w)$*

$$Pr[C \text{ reliable } \wedge \text{ Expt}_z^{A(Q,w),B(w)}(C) = 1] = Pr[\text{Expt}_z^{A(Q,w),B(w)}(C') = 1]$$

From Theorem 4.1 (viability) we know that a passive channel learns nothing significant about the password w or the session-key. Therefore, this is also true of a reliable channel. This implies that in order for C to learn something, he must act unreliably. (Recall that even when C is unreliable, he can learn at most $O(\epsilon)$ about the password and session-key.)

The proof of the above claim is based on having C' emulate an execution for C. Recall that C' is passive and therefore 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 observing that C forwards all messages immediately and unchanged to their intended receiver. If at any point 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 for the entire execution, then C' outputs whatever C does from the experiment. The equality is obtained because when C is reliable, C''s emulation is perfect and when C is unreliable, C' never outputs 1. □

Before discussing the properties of Session-Key Loss and Forward Secrecy, we consider a version of Protocol 1 augmented to include mutual authentication (as discussed in Section 2.2). The augmentation is such that if B rejects, then with probability $1 - O(\epsilon)$, A also rejects. (Recall that when a party rejects, he outputs a uniformly distributed session-key, chosen independently of the protocol execution and password.) The following discussion relates to this augmented protocol.

Loss of Session-Keys: *The* current *session-key remains secure even if* prior *session keys are revealed. Furthermore, the password maintains its security even if all session-keys are revealed.*

First, recall that Claim 10.4 states that the probability that B accepts and C is not reliable is at most $O(\epsilon)$. Then, due to the augmentation, it also holds that the probability that A accepts and C is not reliable is at most $O(\epsilon)$. We continue by relating separately to session-keys generated from executions in which C was reliable and in which C was not reliable.

First, in executions for which C was *not* reliable, with probability $1 - O(\epsilon)$ per execution, we have that the output session-keys are uniformly distributed. Therefore, these session-keys reveal no information about the current session-key or password.

We now show that session-keys generated from executions in which C *was* reliable also reveal nothing significant about the current session-key or password. At the conclusion of the full proof of viability (Section 5), we show that the password w is indistinguishable from a random $\tilde{w} \in_R \mathcal{D}$, given an entire session-transcript *and* the resulting session-key. This refers to a passive channel; however, by Claim 4.11 we have that this also holds for sessions in which C is reliable. Although this is shown for a single session, the proof can be extended to any polynomial number of sessions. We therefore have that the password is secure even if all session-keys are revealed.

The fact that the current session-key also remains secure (when all prior session-keys are revealed), is derived directly from the password security just shown, and the fact that the polynomial Q is chosen randomly and independently in each session. □

Forward Secrecy: *The session-key remains secure even if the password is revealed at a later time.*

As in the case of Session-Key Loss, we need only relate to a session-key generated from an execution in which C is reliable. (Otherwise, with probability $1 - O(\epsilon)$ the session-key is independent of the password.) At the conclusion of the full proof of viability, we show this property for a passive channel C. As previously described, by applying Claim 4.11 we have that this also holds for sessions in which C is reliable. □

5 Full Proof of Viability

In this section, we present the proof of the viability requirement. The requirement for viability 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. We show that such a C will fail except with negligible probability; that is:

Theorem 5.1 (Theorem 4.1 – restated): *Let C be a ppt passive channel. Then, both A and B accept and output the same session-key $k_2(Q(w))$. Furthermore, for every polynomial $p(\cdot)$ and for all sufficiently large n's*

$$\left| Pr_{Q,w}[\mathrm{Expt}_{k_2(Q(w))}^{A(Q,w),B(w)}(C) = 1] - Pr_{Q,w,U_n}[\mathrm{Expt}_{U_n}^{A(Q,w),B(w)}(C) = 1] \right| < \frac{1}{p(n)} \qquad (2)$$

and

$$\left| Pr_{Q,w}[\mathrm{Expt}_{w}^{A(Q,w),B(w)}(C) = 1] - Pr_{Q,w,\tilde{w}}[\mathrm{Expt}_{\tilde{w}}^{A(Q,w),B(w)}(C) = 1] \right| < \frac{1}{p(n)} \qquad (3)$$

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

Proof: Clearly, if C is passive then both parties accept and output the same session-key, as required. Theorems 4.5 and 4.10 (session-key and password secrecy) immediately give us that $k_2(Q(w))$ is $(1-O(\epsilon))$-pseudorandom and that w is $(1-O(\epsilon))$-indistinguishable from a random $\tilde{w} \in_R \mathcal{D}$. However, we wish to prove something stronger: that the session-key is (fully) pseudorandom and that the password is (fully) indistinguishable from a random password.

As we have mentioned, since C is passive, C receives a message transcript and based on this transcript alone must distinguish the session-key from a random string (resp., the password w from $\tilde{w} \in_R \mathcal{D}$). We begin by showing that Q is pseudorandom and w is indistinguishable from $\tilde{w} \in_R \mathcal{D}$ given the transcript of the *first two stages* of the protocol.

Notation: The message-transcript of a protocol execution (by honest parties) 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 $T_2(Q, w) \stackrel{\rm def}{=} \{t_2(Q, w, r_A, r_B)\}_{r_A, r_B}$ the uniform distribution over all possible transcripts for a given Q and w. (Note that the security parameter n, and thus the lengths of Q,w,r_A and r_B are implicit in all these notations.)

The pseudorandomness of Q and indistinguishability of w mentioned above amounts to saying that the distribution ensembles induced by the probability distributions $\{Q_1, w_1, T_2(Q_1, w_1)\}_{Q_1, w_1}$ and $\{Q_2, w_2, T_2(Q_1, w_1)\}_{Q_1, Q_2, w_1, w_2}$ are computationally indistinguishable.[22] This is proved in the following claim. After establishing the claim we show that Equations (2) and (3) in the Theorem hold when C is given a transcript of the *entire* protocol execution (rather than just the first two stages as shown in the claim).

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

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

where Q_1 and Q_2 are random non-constant, linear polynomials over $GF(2^n)$, $w_1, w_2 \in_R \mathcal{D}$ and r_A and r_B are uniform random strings.

Proof: Equivalently, we establish that $\{\{Q_1, w_1, T_2(Q_1, w_1)\}_{Q_1, w_1}\}_{n \in \mathsf{N}}$ is computationally indistinguishable from $\{\{Q_1, w_1, T_2(Q_2, w_2)\}_{Q_1, Q_2, w_1, w_2}\}_{n \in \mathsf{N}}$. The proof is based on the security of the different modules in the protocol. We actual prove something stronger in that the distributions are indistinguishable for *every* pair of polynomials Q_1, Q_2 and passwords w_1, w_2 (i.e., not only when they are randomly chosen). We note that since C is passive, there is no concurrency in this setting. Therefore, we can directly analyze our protocol relying on the security of the different modules.

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.

[22] Notice that it is not true that the distributions $\{Q_1, w_1, T(Q_1, w_1)\}_{Q_1, w_1}$ and $\{Q_2, w_2, T(Q_1, w_1)\}_{Q_1, Q_2, w_1, w_2}$ are indistinguishable, where $T(Q, w)$ denotes the distribution of 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. Now, given (Q_i, w_i), a distinguisher need only compare $f^{2n}(Q_i(w_i))$ to the y-value of the transcript to know whether or not the transcript is based on (Q_i, w_i) or another pair (Q_{2-i}, w_{2-i}).

The Polynomial Evaluation: The inputs to the polynomial evaluation are Q, w and Commit(Q). Denote by $T_P(Q,w)$, the distribution of transcripts for this evaluation. We claim that for every Q_1, Q_2, w_1, w_2, we have that $\{Q_1, w_1, T_P(Q_1, w_1)\}$ and $\{Q_1, w_1, T_P(Q_2, w_2)\}$ are indistinguishable. This can be derived from the following two facts (and is based on the security of the polynomial evaluation that states that A learns nothing and that B learns $Q(w)$ only):

1. For every non-constant, linear polynomial Q, password $w \in \mathcal{D}$ and string $x \in \{0,1\}^n$, we have that
$$\{Q, w, x, T_P(Q,w)\} \stackrel{c}{\equiv} \{Q, w, x, T_P(Q,x)\} \tag{4}$$
where $\stackrel{c}{\equiv}$ denotes computational indistinguishability. 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, A must not be able to distinguish w from x given her message transcript. Equation (4) follows.

2. For every two non-constant, linear polynomials Q_1, Q_2 and string $x \in \{0,1\}^n$ such that $Q_1(x) = Q_2(x)$, it holds that
$$\{Q_1, Q_2, x, T_P(Q_1, x)\} \stackrel{c}{\equiv} \{Q_1, Q_2, x, T_P(Q_2, x)\} \tag{5}$$
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 $Q(x)$). Equation (5) follows.

Now, for every two non-constant polynomials Q_1 and Q_2, there exists a value x such that $Q_1(x) = Q_2(x)$. Therefore, we have that for *every* two non-constant linear polynomials Q_1, Q_2 and *every* two passwords $w_1, w_2 \in \mathcal{D}$

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

where x is such that $Q_1(x) = Q_2(x)$ and where the first and third "$\stackrel{c}{\equiv}$" are due to Equation (4) and the second is from Equation (5). We therefore have that $\{Q_1, w_1, T_P(Q_1, w_1)\} \stackrel{c}{\equiv} \{Q_1, w_1, T_P(Q_2, w_2)\}$ 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 both the polynomial Q and the password w. It remains now to analyze the additional messages from the third stage of the protocol. Recall that the third stage (validation) consists of A sending $y = f^{2n}(Q(w))$, a MAC of the session-transcript keyed by $k_1(Q(w))$ and a zero-knowledge proof. To simplify the exposition, we will assume that A sends the MAC-key itself. This makes no difference as C can compute the MAC value 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 transcripts indistinguishable from real proofs. This implies that this part of the validation stage reveals nothing of Q and w, and we therefore ignore it for the rest of the proof. That is, we may assume that the entire session-transcript consists of $T_2(Q,w)$ and the pair $\left(f^{2n}(Q(w)), k_1(Q(w))\right)$. We are now ready to show that Equations (2) and (3) hold.

Equation (2): Using Claim 5.2 and the fact that for a random Q_2, the value $Q_2(w_2)$ is uniformly distributed in $\{0,1\}^n$, we have

$$\{Q_1(w_1), T_2(Q_1,w_1)\} \stackrel{c}{\equiv} \{Q_2(w_2), T_2(Q_1,w_1)\} \stackrel{c}{\equiv} \{U_n, T_2(Q_1,w_1)\} \tag{6}$$

In particular,

$$\begin{aligned} \{T_2(Q,w), f^{2n}(Q(w)), k_1(Q(w)), k_2(Q(w))\} &\stackrel{c}{\equiv} \{T_2(Q,w), f^{2n}(U_n), k_1(U_n), k_2(U_n)\} \\ &\stackrel{c}{\equiv} \{T_2(Q,w), f^{2n}(U_n^{(1)}), k_1(U_n^{(1)}), U_n^{(2)}\} \end{aligned}$$

where the last "$\stackrel{c}{\equiv}$" is by pseudorandomness of the generator $G(s) = \left(f^{2n}(s), k_1(s), k_2(s)\right)$ and $U_n^{(1)}$ and $U_n^{(2)}$ denote independent uniform distributions over n-bit strings. Using Equation (6), we also have

$$\{T_2(Q,w), f^{2n}(U_n^{(1)}), k_1(U_n^{(1)}), U_n^{(2)}\} \stackrel{c}{\equiv} \{T_2(Q,w), f^{2n}(Q(w)), k_1(Q(w)), U_n\}$$

Combining these two corollaries we obtain that

$$\{T_2(Q,w), f^{2n}(Q(w)), k_1(Q(w)), k_2(Q(w))\} \stackrel{c}{\equiv} \{T_2(Q,w), f^{2n}(Q(w)), k_1(Q(w)), U_n\}$$

That is, the session-key $k_2(Q(w))$ is pseudorandom with respect to C's view, even given the entire protocol transcript.

Equation (3): Q is pseudorandom given $T_2(Q,w)$ and therefore $Q(w)$ completely hides w. (Here we use the fact that for every $x \in \{0,1\}^n$, the value of a random (non-constant) linear polynomial at x is uniformly distributed.) That is, $\{T_2(Q,w), Q(w), w\}_{Q,w}$ is indistinguishable from $\{T_2(Q,w), Q(w), \tilde{w}\}_{Q,w,\tilde{w}}$. This immediately implies that

$$\{T_2(Q,w), f^{2n}(Q(w)), k_1(Q(w)), w\} \stackrel{c}{\equiv} \{T_2(Q,w), f^{2n}(Q(w)), k_1(Q(w)), \tilde{w}\}$$

and we thus have that the password w is indistinguishable from $\tilde{w} \in_R \mathcal{D}$ with respect to C's view. ■

A Note Regarding Loss of Session Keys: Loosely speaking, the property of security in the face of session-key loss (described in Section 4.7) states that the password w remains secure even if the session-key is revealed. Our *proof* of Equation (3) is used to show this. That is, we have shown that $\{T_2(Q,w), Q(w), w\}_{Q,w}$ is indistinguishable from $\{T_2(Q,w), Q(w), \tilde{w}\}_{Q,w,\tilde{w}}$. This implies that

$$\{T_2(Q,w), f^{2n}(Q(w)), k_1(Q(w)), k_2(Q(w)), w\} \stackrel{c}{\equiv} \{T_2(Q,w), f^{2n}(Q(w)), k_1(Q(w)), k_2(Q(w)), \tilde{w}\}$$

That is, w is indistinguishable from $\tilde{w} \in_R \mathcal{D}$ even given the entire session-transcript *and* the resulting session-key $k_2(Q(w))$.

A Note Regarding Forward Secrecy: The property of forward secrecy (described in Section 4.7) states that the session-key remains secure even if the password is later revealed. As in the property of session-key loss, the proof of Equation (3) can be used to show that forward secrecy holds. That is,

$$\begin{aligned}
\{T_2(Q,w), f^{2n}(Q(w)), k_1(Q(w)), k_2(Q(w)), w\} &\overset{c}{\equiv} \{T_2(Q,w), f^{2n}(Q(w)), k_1(Q(w)), k_2(Q(w)), \tilde{w}\} \quad (7)\\
&\overset{c}{\equiv} \{T_2(Q,w), f^{2n}(Q(w)), k_1(Q(w)), U_n, \tilde{w}\} \quad (8)\\
&\overset{c}{\equiv} \{T_2(Q,w), f^{2n}(Q(w)), k_1(Q(w)), U_n, w\} \quad (9)
\end{aligned}$$

where Equation (7) is exactly as in the loss of session-keys, Equation (8) is due to the pseudorandomness of $k_2(Q(w))$ (Equation (2) in Theorem 4.1) and Equation (9) is due to the indistinguishability of w (Equation (3) in Theorem 4.1). This means that $k_2(Q(w))$ is pseudorandom with respect to the view of a passive channel, even when given the password w.

6 Full Proof of Pseudorandomness of $Q(w)$

Theorem 4.2 states that at the conclusion of Stage 2 of the (A,C)-execution, the value $Q(w)$ is $(1-2\epsilon)$-pseudorandom with respect to C's view. As described in Section 4.3, the theorem is obtained by combining Lemmas 4.3 and 4.4 which are proved in full here.

6.1 Proof of Lemma 4.3

Lemma 6.1 (Lemma 4.3 – restated): *For every ppt channel C, every polynomial $p(\cdot)$ and for all sufficiently large n's*

$$\left|Pr_{Q,w,U_n}[\mathrm{Expt}_{U_n}^{A_2(Q,w),B(w)}(C)=1] - Pr_{Q,w,\tilde{w}}[\mathrm{Expt}_{Q(\tilde{w})}^{A_2(Q,w),B(w)}(C)=1]\right| < \epsilon + \frac{1}{p(n)}$$

Proof: The lemma holds even if C knows the password w, and so we prove the lemma for a channel C who is given w for auxiliary input. Since the password w is known to C, and this constitutes all of B's secret input, the (C,B)-execution can be perfectly emulated by C himself. This means that C (with auxiliary input w) can distinguish between $Q(\tilde{w})$ and U_n in a two-party setting (involving only A_2 and C) with the same probability as in our concurrent setting (where B is also involved). Formally, consider the following two-party experiment between A_2 and C (we stress that B does *not* participate in the execution):

$$\begin{aligned}
&\mathrm{Expt}_z^{A_2(Q,w)}(C(w)):\\
&\quad s \leftarrow C_1^{A_2(Q,w)}(w, 1^n)\\
&\quad \mathsf{return}\ C_2(s,z)
\end{aligned}$$

Formally, for every ppt channel C there exists a ppt machine C' such that for every randomized process $z = Z(Q,w)$

$$Pr_{Q,w}[\mathrm{Expt}_z^{A_2(Q,w)}(C'(w))=1] = Pr_{Q,w}[\mathrm{Expt}_z^{A_2(Q,w),B(w)}(C)=1] \quad (10)$$

C' works by emulating the $C_1^{A_2(Q,w),B(w)}$ setting for C. This is done by interacting with A_2 and playing the role of B (with input w) in the (C,B) execution. Since C' knows w, the emulation is perfect and thus the output from the experiment is identical. By applying Equation (10) to the probabilities in the lemma (once setting $z = U_n$ and once setting $z = Q(\tilde{w})$), we have that it is enough to prove that for every C' interacting only with A_2 (where there is no concurrent execution with B), the following holds

$$\left| Pr_{Q,w,U_n}[\text{Expt}_{U_n}^{A_2(Q,w)}(C'(w)) = 1] - Pr_{Q,w,\tilde{w}}[\text{Expt}_{Q(\tilde{w})}^{A_2(Q,w)}(C'(w)) = 1] \right| < \epsilon + \frac{1}{poly(n)} \tag{11}$$

We now prove Equation (11). First note that the non-malleable commitment sent by A_2 in this setting plays no role in the continuation of the protocol. Due to the hiding property of the commitment, if A_2 commits to random values instead of to (Q, w), this makes at most a negligible difference to C''s success. This enables us to remove the non-malleable commitment entirely because C' can internally simulate receiving a random commitment.

What remains is thus the (A_2, C) pre-key exchange, consisting of A_2 sending Commit(Q) to C followed by a single secure 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 Q. As described in the proof sketch, this implies that C' can distinguish $Q(\tilde{w})$ from U_n with probability at most negligibly greater than ϵ (where the ϵ advantage comes from the case that $\tilde{w}$ 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'_1 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. Denote the ideal model parties by $\hat{A}_2$ and $\hat{C}'_1$ ($\hat{C}'_1$ is adversarial). By the definition of secure two-party computation, for every real adversary C'_1 interacting with A_2, there exists an ideal adversary $\hat{C}'_1$ interacting with $\hat{A}_2$ such that the output distributions of C'_1 and $\hat{C}'_1$ are indistinguishable. Denote the outputs of C'_1 and $\hat{C}'_1$ by s and $\hat{s}$ respectively. It therefore holds that for every ppt distinguishing machine D, $Pr[D(s) = 1] \approx Pr[D(\hat{s}) = 1]$. However, by the definition of secure computation, the distinguishing machine D also receives the parties' respective inputs Q and w. Therefore, it likewise holds that for every randomized process $z = Z(Q, w)$, we have that $Pr[D(s, z) = 1] \approx Pr[D(\hat{s}, z) = 1]$. This is true for every D and in particular for C'_2 (who receives (s, z) by the experiment definition). That is, for every such $z = Z(Q, w)$,

$$\left| Pr[\text{Expt}_z^{A_2(Q,w)}(C'(w)) = 1] - Pr[\text{Expt}_z^{\hat{A}_2(Q,w)}(\hat{C}'(w)) = 1] \right| < \frac{1}{poly(n)}$$

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

$$\left| Pr_{Q,w,U_n}[\text{Expt}_{U_n}^{\hat{A}_2(Q,w)}(\hat{C}'(w)) = 1] - Pr_{Q,w,\tilde{w}}[\text{Expt}_{Q(\tilde{w})}^{\hat{A}_2(Q,w)}(\hat{C}'(w)) = 1] \right| < \epsilon + \frac{1}{poly(n)}$$

We thus 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 $\hat{C}'_1$ 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.)

Assume for now that the execution of the polynomial evaluation is such that $\hat{C}'_1$ 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, $\hat{C}'_1$'s view is exactly $(r, \text{Commit}(Q), Q(w_C))$, where r is the string of his random coin tosses and w_C is determined by $\hat{C}'_1$ based on r and Commit(Q). For sake of clarity, we augment the view by w_C itself (i.e, we write $\hat{C}'_1$'s view as $(r, \text{Commit}(Q), w_C, Q(w_C))$). Assuming without loss of generality that $\hat{C}'_1$ always outputs his entire view, we conclude that $\hat{C}'_2$ receives as input either

$(r, \text{Commit}(Q), w_C, Q(w_C), U_n)$ or $(r, \text{Commit}(Q), w_C, Q(w_C), Q(\tilde{w}))$, where $\tilde{w} \in_R \mathcal{D}$. We now show that if $w_C \neq \tilde{w}$, then the above two tuples are indistinguishable. That is,

$$\{r, \text{Commit}(Q), w_C, Q(w_C), U_n\}_{Q,U_n} \stackrel{c}{\equiv} \{r, \text{Commit}(Q), w_C, Q(w_C), Q(\tilde{w}) \mid w_C \neq \tilde{w}\}_{Q,\tilde{w}}$$

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}.) Next, notice that the distributions $\{r, \text{Commit}(0^{2n}), w_C, Q(w_C), Q(\tilde{w}) \mid w_C \neq \tilde{w}\}$ and $\{r, \text{Commit}(0^{2n}) w_C, Q(w_C), U_n\}$ are *statistically* close.[23] Then, by returning the commitment to Q in place of the commitment to 0^{2n}, we have that for every ppt $\hat{C}'_2$

$$\begin{aligned} &Pr_{Q,U_n}[\hat{C}'_2(r, \text{Commit}(Q), w_C, Q(w_C), U_n) = 1] \\ &\quad \approx Pr_{Q,\tilde{w}}[\hat{C}'_2(r, \text{Commit}(Q), w_C, Q(w_C), Q(\tilde{w})) = 1 \mid w_C \neq \tilde{w}] \end{aligned}$$

or equivalently

$$Pr_{Q,U_n}[\text{Expt}_{U_n}^{\hat{A}_2(Q,w)}(\hat{C}'(w)) = 1] \approx Pr_{Q,\tilde{w}}[\text{Expt}_{Q(\tilde{w})}^{\hat{A}_2(Q,w)}(\hat{C}'(w)) = 1 \mid w_C \neq \tilde{w}]$$

Since $\tilde{w} \in_R \mathcal{D}$ and it is chosen independently of the ${\hat{C}'}_1^{A_2(Q,w)}(w, 1^n)$ execution, we have that $Pr[w_C = \tilde{w}] \leq \epsilon$ (with equality when w_C is chosen from $\mathcal{D}$). Therefore

$$\left| Pr[\text{Expt}_{U_n}^{\hat{A}_2(Q,w)}(\hat{C}'(w)) = 1] - Pr[\text{Expt}_{Q(\tilde{w})}^{\hat{A}_2(Q,w)}(\hat{C}'(w)) = 1] \right| < \epsilon + \frac{1}{poly(n)} \qquad (12)$$

(The exact calculation is derived by breaking the probability into two conditional cases; the first where $w_C = \tilde{w}$ and the second where $w_C \neq \tilde{w}$.) This completes the analysis of the simplified case in which the polynomial evaluation always outputs $Q(w_C)$ for some w_C (and never outputs $\perp$). However, $\hat{C}'_1$ may cause the result of the evaluation to be $\perp$ and we must show that this cannot help him. Intuitively, if $\hat{C}'_1$ were to receive $\perp$ then he would learn *nothing* about Q and this would thus be a "bad" strategy. However, it must be shown that $\hat{C}'_1$ cannot learn anything by the mere fact that he received $\perp$ and not $Q(w_C)$.

This can be seen by noticing that the bit indicating whether $\hat{C}'_1$ receives $\perp$ or $Q(w_C)$, denoted χ_C, is almost independent of Q (by the hiding property of the commitment). Therefore, χ_C is also almost independent of the values $Q(\tilde{w})$ and U_n. Thus, augmenting the distinguisher's view by χ_C does not change the situation analyzed above and we have

$$\{r, \text{Commit}(Q), w_C, Q(w_C), U_n, \chi_C\}_{Q,U_n} \stackrel{c}{\equiv} \{r, \text{Commit}(Q), w_C, Q(w_C), Q(\tilde{w}), \chi_C \mid w_C \neq \tilde{w}\}_{Q,\tilde{w}}$$

Noting that the state output by $\hat{C}'_1$ is polynomial time computable from $(r, \text{Commit}(Q), w_C, Q(w_C), \chi_C)$, it follows that in also in the general case (where the polynomial evaluation outputs $\perp$), Equation (12) holds. ■

[23] 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, $w \neq \tilde{w}$ implies that $Q(w_C) \neq Q(\tilde{w})$ 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.

6.2 Proof of Lemma 4.4

Lemma 6.2 (Lemma 4.4 – restated): *For every ppt channel C, every polynomial $p(\cdot)$ and for all sufficiently large n's*

$$\left|Pr_{Q,w}[\text{Expt}_{Q(w)}^{A_2(Q,w),B(w)}(C)=1] - Pr_{Q,w,\tilde{w}}[\text{Expt}_{Q(\tilde{w})}^{A_2(Q,w),B(w)}(C)=1]\right| < \epsilon + \frac{1}{p(n)}$$

Proof: The outline of the proof is as follows. As in the previous lemma, the lemma holds even if C knows some secret information: in this case, the value of the polynomial Q. Given Q, we show how C can internally emulate the (A_2, C)-execution and we therefore remain only with the (C, B)-execution. Now we have a standard two-party setting and we wish to analyze the probability that, in this setting, C distinguishes $Q(w)$ from $Q(\tilde{w})$. Seemingly, we should be able to derive this directly from the security of the polynomial evaluation. However, we encounter a technical difficulty due to the fact that currently known composition theorems for secure computation do not apply to our specific scenario. We discuss the reason for this and then show how to bypass the problem in this particular case.

As mentioned above, the lemma holds even if C knows Q and this enables us to remove the (A_2, C) execution. Formally, consider the following experiment in which A_2 does *not* participate:

$$\begin{array}{l}\mathsf{Expt}_z^{B(w)}(C(Q)):\\ \quad s \leftarrow C_1^{B(w)}(Q, 1^n)\\ \quad \mathsf{return}\ C_2(s,z)\end{array}$$

We claim that for every ppt channel C, there exists a ppt machine C' with auxiliary input Q, such that for every randomized process $z = Z(Q, w)$

$$\left|Pr_{Q,w}[\text{Expt}_z^{B(w)}(C'(Q))=1] - Pr_{Q,w}[\text{Expt}_z^{A_2(Q,w),B(w)}=1]\right| < \frac{1}{poly(n)} \tag{13}$$

The party C' works by simply playing A_2's role to C; C' is able to do this because A_2's only input into the pre-key exchange is the polynomial Q (and the commit stage can be simulated sufficiently well without any input). Specifically, C''s simulation works by first non-malleably committing to a random value (instead of to (Q, w)). Then, C' continues exactly as A_2 would by sending an ordinary commitment to Q and participating in the polynomial evaluation with C, inputting Q. The only difference in C's view is with respect to the non-malleable commitment, and this can make only a negligible difference.[24] We thus obtain Equation (13). It remains to show that for every ppt C' interacting only with B, the channel C' can distinguish $Q(w)$ from $Q(\tilde{w})$ with probability less than $\epsilon + \frac{1}{poly(n)}$.

[24]Assume by contradiction that C behaves differently when he receives a commitment to (Q, w) or to U_{3n}. We now show that C can be used to distinguish such commitments. First notice that the non-malleable commitment is referred to by A only during the zero-knowledge proof in the validation stage. Since A_2 does not reach this stage, the commitment is not used at all. This is a crucial point enabling any non-negligible difference in C's behavior to be used to distinguish commitments to (Q, w) from random commitments. Now, let D be a distinguisher given a commitment to either (Q, w) or U_{3n}. Then, D can perfectly simulate $C_1^{A(Q,w),B(w)}(1^n)$ (he knows Q and w), except instead of simulating a commitment by A to (Q, w) in the (A, C) commit stage, D uses his input (challenge) commitment. As the rest of the simulation by D is independent of the value in this commitment, the only difference is with respect to this step. Therefore, any non-negligible difference in C's output implies distinguishability of the commitments. This argument is used a number times during our proof.

We note that this is in contrast to a situation where A does run her zero-knowledge proof (and specifically in a real execution), where the value of the commitment is crucial.

Relying on the Security of Two-Party Computation: As described, we now have a two-party setting in which C and B interact according to our protocol. In this setting, we wish to show that C can distinguish $Q(w)$ from $Q(\tilde{w})$ with probability at most negligibly greater than ϵ. Intuitively this is due to the security of the polynomial evaluation which ensures that C learns nothing from it. As for the rest of the (C, B) execution (i.e., the validation and decision stages), all that C can learn is B's accept/reject bit. The proof is thus based on showing that due to the design of the validation stage, C gains a distinguishing advantage of at most ϵ from the accept bit of B. We would therefore expect to proceed by replacing the polynomial evaluation by an ideal module computed by a trusted third party. By analyzing C's distinguishing probability in this ideal setting, we would then derive his distinguishing probability when the ideal polynomial evaluation is replaced by a (real) secure evaluation. Any difference in the probabilities would contradict the security of the polynomial evaluation and we would thus obtain the lemma.

However, this argument does *not* quite work here. First notice that if the polynomial evaluation was run by itself then, as a secure protocol, we know that C learns nothing of w from it. However, this is not the case; rather the evaluation is run as *part* of a larger protocol. The fact that in this larger setting, C learns nothing of w from the polynomial evaluation, must be formally justified. Loosely speaking, this is the objective of the Sequential Composition Theorem [14]. We begin by discussing the formulation of this theorem, upon which our above argument rests.

The starting point of the composition theorem is an arbitrary protocol π that involves an ideal subroutine call to a functionality f (the protocol with the ideal call is denoted π^f).[25] Now consider the protocol π^ρ that is derived by replacing the ideal call to f with a secure protocol ρ for computing f. The crucial point here to notice is that, outside of the ρ subprotocol, the protocol π^ρ depends *only* on the output of ρ, and not on any intermediate messages sent during its computation. This is because ρ directly takes the place of f, for which there *are no* intermediate messages (it is an ideal oracle call). The composition theorem states that, in this setting, attacking π^ρ is not more advantageous than attacking the ideal protocol π^f.

Unfortunately, our protocol does not fit into this scenario. In the validation stage of our protocol C' must send B a MAC of the entire message transcript, including the messages belonging to the secure polynomial evaluation. That is, the protocol *definition* depends on intermediate messages belonging to a secure two-party protocol. As discussed above, the composition theorem of [14] does not apply to such a case.[26] It is possible to generalize the Sequential Composition Theorem to include such protocols; we leave this for future work. For now, we show how to bypass this problem in our specific protocol.

"Removing" the Validation Stage: The above problem is caused by the MAC sent by C' during the validation stage. We therefore first show how we can remove the validation stage so that this affects C''s probability of distinguishing $Q(w)$ from $Q(\tilde{w})$ by at most ϵ (once the validation stage is removed, we can just apply a standard analysis). First notice that if C' can predict B's output bit perfectly, then the (C', B)-validation stage is meaningless with respect to C''s view. This is because B's only private output from this stage is his accept/reject bit (B also sends messages in the role of the verifer in a zero-knowledge proof; however, B is an honest verifier and his messages are thus simulatable). Therefore, C' can internally simulate B's role in this stage, using his perfect

[25] For simplicity we relate to the case of an arbitrary protocol that uses a single ideal call; the full theorem, however, relates to the more general case of many sequential ideal calls to $f_1, \ldots, f_m$.

[26] We stress that in our protocol, the message transcript of the secure polynomial evaluation is a vital part of the continuation of the protocol. In fact, if the MAC were only to be applied to the message transcript *excluding* the internal messages of the polynomial evaluation then the protocol would no longer be correct (see Appendix B).

prediction for B's accept/reject bit. Taking this a step further, if C' can predict B's output bit and be correct with probability at least $1-\epsilon$, then the (C', B)-validation stage can be removed, with a difference of at most ϵ to C''s view. We therefore proceed by showing how C' can indeed predict B's output bit by himself, with accuracy $1-\epsilon$. This is possible because when C' interacts with B in this non-concurrent setting, the probability that B accepts is at most negligibly more than ϵ. That is, C' can predict that B always rejects and he will be correct with probability $1-\epsilon$. Recall that in this scenario, A_2 is not involved and therefore C' must attempt to have B accept without any "help" from A_2.

Claim 6.3 *For every ppt channel C' playing in ${C'}_1^{B(w)}(Q, 1^n)$, for every polynomial $p(\cdot)$ and for all sufficiently large n's*

$$Pr[B = \text{acc}] < \epsilon + \frac{1}{p(n)}$$

Proof: We prove the claim by considering a modified party B' who executes everything in the same way as B except that in the validation stage he only checks that $y = f^{2n}(\tau_B)$ (ignoring the MAC and the zero-knowledge proof). Recall that τ_B is B's output from the polynomial evaluation. Since we only omitted checks that may make B reject, we have that

$$Pr[B = \text{acc}] \le Pr[B' = \text{acc}]$$

Next we show that B' accepts with probability at most $\epsilon + \frac{1}{poly(n)}$. This is based on the security of the polynomial evaluation which ensures that C learns nothing of w before the validation stage. That is, we show that the validation stage (which now amount to B' checking if $f^{2n}(\tau_B) = y$ where y is sent by C in this stage) ensures that if w is uniformly distributed in $\mathcal{D}$ (with respect to C's view), then party B' accepts with probability at most ϵ.

As this claim is due to the security of the polynomial evaluation, we analyze the probability that B' accepts in an ideal execution. Denote the ideal model parties by $\hat{C}'$ and $\hat{B}'$. We claim that for every ppt $\hat{C}'$, it holds that $Pr[\hat{B}' = \text{acc}] \le \epsilon$. The channel $\hat{C}'$'s view of the protocol is essentially *empty* (apart from his own randomness). This is because $\hat{C}'$ receives nothing from the polynomial evaluation and the only other messages sent by B are as the receiver of a non-malleable commitment. Since this involves no secrets from $\hat{B}'$'s part, party $\hat{C}'$ learns nothing from them.

Now, $\hat{B}'$ accepts only if $y = f^{2n}(Q_C(w))$ where Q_C is the polynomial input by $\hat{C}'$ to the secure evaluation (recall that if $\hat{B}'$ receives $\perp$ from the polynomial evaluation then he always rejects). Since $\hat{C}'$ learns nothing of w in the execution, with respect to his view the string $f^{2n}(Q_C(w))$ is uniformly distributed in the set $\{f^{2n}(Q_C(w')) \ : \ w' \in \mathcal{D}\}$. As f^{2n} and Q_C are 1–1 functions (Q_C is a non-constant linear polynomial), this set contains exactly $|\mathcal{D}|$ elements. We therefore conclude that for every $\hat{C}'$ interacting with $\hat{B}'$ in an ideal execution,

$$Pr[\hat{B}' = \text{acc}] \le \frac{1}{|\mathcal{D}|} = \epsilon$$

By the security definition of multi-party computation, for every adversary C' interacting with B' in the real model, there exists an adversary $\hat{C}'$ interacting with $\hat{B}'$ in the ideal model, such that the outputs of B' and $\hat{B}'$ are indistinguishable. This is the correctness requirement described in Section A.1. This implies that

$$\left| Pr[B' = \text{acc}] - Pr[\hat{B}' = \text{acc}] \right| < \frac{1}{poly(n)}$$

(Otherwise, one can distinguish the real and ideal executions with non-negligible probability by simply outputting the accept/reject bit.) We conclude that for every ppt adversary C' in a real execution

$$Pr[B' = \text{acc}] < \epsilon + \frac{1}{poly(n)}$$

and the lemma follows. ■

We are now ready to remove the *entire* validation stage from the protocol.[27] We do this formally by modifying B so that he does not output any accept/reject bit. We call the modified party B_2 (as with A_2, he only participates in the first 2 stages). Now, there exists a ppt channel C'' such that

$$Pr_{Q,w}[\text{Expt}_{Q(w)}^{B_2(w)}(C''(Q)) = 1] = Pr_{Q,w}[\text{Expt}_{Q(w)}^{B(w)}(C'(Q)) = 1 \mid B = \text{rej}] \tag{14}$$

$$Pr_{Q,w,\tilde{w}}[\text{Expt}_{Q(\tilde{w})}^{B_2(w)}(C''(Q)) = 1] = Pr_{Q,w,\tilde{w}}[\text{Expt}_{Q(\tilde{w})}^{B(w)}(C'(Q)) = 1 \mid B = \text{rej}] \tag{15}$$

The strategy for C'' is to simply run ${C'}_1^{B_2(w)}$ and continue by "assuming" that B_2 outputs reject. Thus in executions for which B rejects, the output of C'' (interacting with B_2) equals the output of C' (interacting with B).

Once the validation step is removed, C'' cannot distinguish $Q(\tilde{w})$ from $Q(w)$ since he obtains no output from the polynomial evaluation (and this is the only part of the protocol where B uses w). This is captured by the following.

Claim 6.4 *For every ppt channel C'', every polynomial $p(\cdot)$ and for all sufficiently large n's*

$$\left|Pr_{Q,w}[\text{Expt}_{Q(w)}^{B_2(w)}(C''(Q)) = 1] - Pr_{Q,w,\tilde{w}}[\text{Expt}_{Q(\tilde{w})}^{B_2(w)}(C''(Q)) = 1]\right| < \frac{1}{p(n)}$$

We now put everything together in order to show that C cannot distinguish $Q(w)$ from $Q(\tilde{w})$ with probability noticeably greater than ϵ (and so establish Lemma 4.4).

$$\begin{aligned}
&\left|Pr[\text{Expt}_{Q(w)}^{A_2(Q,w),B(w)} = 1] - Pr[\text{Expt}_{Q(\tilde{w})}^{A_2(Q,w),B(w)} = 1]\right| && (16)\\
\approx\ &\left|Pr[\text{Expt}_{Q(w)}^{B(w)}(C'(Q)) = 1] - Pr[\text{Expt}_{Q(\tilde{w})}^{B(w)}(C'(Q)) = 1]\right| && (17)\\
=\ &\Big|\Big(Pr[\text{Expt}_{Q(w)}^{B(w)}(C'(Q)) \mid B = \text{acc}] - Pr[\text{Expt}_{Q(\tilde{w})}^{B(w)}(C'(Q)) \mid B = \text{acc}]\Big) \cdot Pr[B = \text{acc}] && (18)\\
&+ \Big(Pr[\text{Expt}_{Q(w)}^{B(w)}(C'(Q)) = 1 \mid B = \text{rej}] - Pr[\text{Expt}_{Q(\tilde{w})}^{B(w)}(C'(Q)) = 1 \mid B = \text{rej}]\Big) \cdot Pr[B = \text{rej}]\Big| &&\\
\leq\ &Pr[B = \text{acc}] + \left|Pr[\text{Expt}_{Q(w)}^{B_2(w)}(C''(Q)) = 1] - Pr[\text{Expt}_{Q(\tilde{w})}^{B_2(w)}(C''(Q)) = 1]\right| && (19)\\
<\ &\epsilon + \frac{1}{poly(n)} && (20)
\end{aligned}$$

where the soft equality between lines 16 and 17 is as shown in Equation (13), the equality between lines 18 and 19 is by Equations (14) and (15), and the inequality between lines 19 and 20 is due to Claim 6.3 (for the first part), and Claim 6.4 (for the second part). ■

[27] This is in contrast to the *proof* of Claim 6.3 where we removed only the zero-knowledge proof and MAC from the validation stage.

7 On Simulating A's Zero-Knowledge Proof

In some of the proofs that follow, we wish to "remove" the (A, C) validation stage, which includes a zero-knowledge proof. 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 himself. 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 by B. Rewinding C would thus also require rewinding B. However, as 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 of Richardson and Kilian [40] in order to address the problem for our specific 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 be equal to the total number of rounds in our protocol (not including the zero-knowledge proof itself) *plus* $t = t(n)$, where $t(n)$ equals any non-constant function of the security parameter n (say $t(n) = \log n$).

To motivate how the proof simulation is done in our scenario, consider the following mental experiment in which the (C, B)-execution does not include the zero-knowledge proof (given by C to B). In such a case, the total number of rounds in the (C, B) execution equals $m - t$. 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 are not required to rewind B (which is not possible, since B is an outside party).

The above reasoning can be applied in the following scenario. Consider a *modified* party $B_{\not{zk}}$ who is exactly the same as B, except that his protocol definition does not include verifying a zero-knowledge proof from C. Then, as we have described in the above mental experiment, when C interacts with A and $B_{\not{zk}}$, the proof given by A to C can be simulated by C himself.

7.1 The Main Result

The Modified Parties $A_{\not{zk}}$ and $B_{\not{zk}}$: In our above description we described a modified party $B_{\not{zk}}$, whose protocol definition does not include verifying C's zero-knowledge proof. Furthermore, when we say that A's proof can be simulated by C himself, this means that A too can be modified to a party $A_{\not{zk}}$, whose protocol definition does not include proving any statement in the validation stage. Before continuing, we formally define what we mean by these modifications of A and B to $A_{\not{zk}}$ and $B_{\not{zk}}$ respectively. 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 (and likewise, C should send such a MAC to B). This message-transcript includes the messages of the zero-knowledge proof. Therefore, the protocols of $A_{\not{zk}}$ and $B_{\not{zk}}$ must be appropriately redefined to take this issue into account.

The $A_{\not zk}$ Modification: In the zero-knowledge proof with C, party A plays the prover. The essence of the modification of A to $A_{\not zk}$ is in replacing A's prover role in the (A, C)-proof by a simulator. This modification works only if C's view in a protocol execution with $A_{\not zk}$ is indistinguishable from his view in an execution with A. As mentioned, the MAC sent by A in the validation stage is computed on the entire message transcript, including messages from the zero-knowledge proof. Therefore, the MAC sent by $A_{\not zk}$ must also include messages from the simulated proof. However, $A_{\not zk}$ 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 $A_{\not zk}$ to be exactly the same as A, except that she has no 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 which she appends to her message transcript. This means that the only difference between A and $A_{\not zk}$'s message transcripts is that A's transcript includes messages from a zero-knowledge proof and $A_{\not zk}$'s transcript includes s instead. Intuitively, if s is the transcript of the simulated proof, then A and $A_{\not zk}$'s message transcripts are indistinguishable. This ensures that the MACs sent by A and $A_{\not zk}$ respectively are indistinguishable.

The $B_{\not zk}$ Modification: In the zero-knowledge proof with C, party B plays the verifier. We wish to modify B to $B_{\not zk}$ so that the only difference between the parties is that $B_{\not zk}$ does not participate in the zero-knowledge proof. The modification should be such that $B_{\not zk}$ has the same behavior as a party who plays the verifier in the zero-knowledge proof, but always considers the verification to be successful (irrespective of the real outcome). A problem arising in defining $B_{\not zk}$ is that the zero-knowledge proof has influence on B's protocol definition *beyond* the mere result of the verification procedure. Again, this "influence" is due to the MAC that B receives in the validation stage; this MAC is computed on the entire message transcript, including the messages from the zero-knowledge proof. Furthermore, this MAC is part of B's decision process in whether to output accept or reject. Therefore, our modification of B is such that the resulting message transcripts for B and $B_{\not zk}$ are identical. That is, similarly to $A_{\not zk}$, instead of playing the verifier in the proof, $B_{\not zk}$ expects to receive a string s which he then appends to his message transcript. Then, if s equals a valid proof transcript, the message transcripts of B and $B_{\not zk}$ are identical.

We begin by showing that with respect to the "distinguishing experiments" defined in Section 4.1, there is no difference if C interacts with A and B or with A and $B_{\not zk}$. Intuitively, this is because B always plays an honest-verifier in the zero-knowledge proof and C knows whether the proof succeeded or not. Therefore, C can simulate the proof and the affects of its result by himself.

Lemma 7.1 *Let $B_{\not zk}$ be the above-defined modified party. Then, for every ppt channel C there exists a ppt channel C' such that for every randomized process $z = Z(Q, w)$*

$$Pr_{Q,w}[\mathrm{Expt}_z^{A(Q,w),B(w)}(C) = 1] = Pr_{Q,w}[\mathrm{Expt}_z^{A(Q,w),B_{\not zk}(w)}(C') = 1]$$

Proof: The equality in the lemma is obtained by having the channel C' (who interacts with A and $B_{\not zk}$) simulate the scenario in which C interacts with A and B. This simulation is defined so that C's view is identical to the setting where C really interacts with A and B. Notice first that in both settings, C and C' interact with A (and not with a modified party). Therefore, with respect to the (A, C)-execution, channel C' need do nothing beyond forwarding all messages between A and C (without modification). Furthermore, until the zero-knowledge proof is reached in the (C, B)-execution stage, there is also no difference between B and $B_{\not zk}$. Therefore, the simulation of this part just involves C' forwarding all messages between C and $B_{\not zk}$.

The interesting part of the simulation is from the (C, B) zero-knowledge proof until the conclusion of the (C, B)-execution. This includes the zero-knowledge proof from C to B, and B's accept/reject bit. We stress that the simulation must ensure that C receives the same accept/reject bit from $B_{\not\approx k}$ that B would have output. (Notice that in general $B_{\not\approx k}$'s output-bit may not be the same as B's, because if the (C, B) zero-knowledge proof fails B always rejects. On the other hand, $B_{\not\approx k}$ does not have such a proof and may therefore accept in the same situation.) The simulation is thus as follows:

The Simulation:

1. *Zero-Knowledge with C:* C' emulates B's role as the verifier in the proof with C. The basis for this emulation is the fact that B plays an honest verifier. Therefore, C''s emulation consists of being an honest verifier in B's place.

   C' plays the verifier in this proof and therefore either accepts or rejects the proof. Let zk-accept be a random variable such that zk-accept $= 1$ if and only if C' accepts the proof.

2. *The String* s *Received by $B_{\not\approx k}$:* By the definition of $B_{\not\approx k}$, party $B_{\not\approx k}$ expects to receive a string s at the point of B's zero-knowledge proof. This string is then appended to $B_{\not\approx k}$'s message transcript. Channel C' sets s to equal the transcript of messages belonging to the internal zero-knowledge proof execution it had conducted in Step 1.

3. *The MAC from C:* In the last step of the protocol, C sends a MAC to B. The MAC forwarded by C' to C depends on whether or not C' accepted the zero-knowledge proof (i.e., if zk-accept $= 1$ or not).

   - *Case* zk-accept $= 1$*:* In this case, channel C' forwards (to $B_{\not\approx k}$) the MAC sent by C.
   - *Case* zk-accept $= 0$*:* In this case, channel C' sends an *invalid* string in place of the MAC. (This ensures that $B_{\not\approx k}$ will reject.)

4. *$B_{\not\approx k}$'s Output Bit:* C' receives $B_{\not\approx k}$'s accepts/reject bit and forwards it to C.

This concludes the simulation. We now show that C's view in this simulation is identical to his view in a real execution with B. As discussed above, we need only consider the last part of the (B, C) execution. Firstly, C's view of the zero-knowledge proof with B is identical to the view simulated by C', since C' emulates B perfectly. Next, note that C' accepts the zero-knowledge proof with the *same* probability that B would have; this is a central point in showing that C's view of the rest remains unchanged. Consider the following two cases:[28]

- **Case** zk-accept $= 1$**:** Channel C' accepted the proof and thus B would have accepted it (with the same probability). Party B therefore accepts if he received $y_B = f^{2n}(\tau_B)$ where τ_B is his output from the polynomial evaluation, and if the MAC is correct. The modification has no effect on the y-value and therefore this makes no difference. It remains to show that the probability that B accepts the MAC from C equals the probability that $B_{\not\approx k}$ accepts this same MAC. This is true if the message transcripts that both B and $B_{\not\approx k}$ hold are the same (by the "same", we mean that they are identically distributed). Apart the zero-knowledge proof, the transcripts are

[28] Let B-zk-accept be a random variable such that B-zk-accept $= 1$ if and only if B accepts his zero-knowledge proof from C. Then, formally we show below that for $b \in \{0, 1\}$, the probability that $\mathrm{Expt}_z^{A(Q,w),B(w)}(C) = 1$ *conditioned* on B-zk-accept $= b$ **equals** the probability that $\mathrm{Expt}_z^{A(Q,w),B_{\not\approx k}(w)}(C') = 1$ *conditioned* on zk-accept $= b$. Now, since for $b \in \{0, 1\}$, we have that $Pr[\text{B-zk-accept} = b] = Pr[\text{zk-accept} = b]$, we obtain the equality in the lemma.

identical by definition. Furthermore, $B_{\not\ni k}$ appends the messages from C''s internal emulation of the proof (with C) to his message transcript. As C' plays an honest verifier exactly as B would have, this means that their message transcripts are identically distributed. We conclude that the probability that $B_{\not\ni k}$ accepts in this case equals the probability that B accepts.

- **Case** $\mathsf{zk\text{-}accept} = 0$**:** Channel C' rejected the proof and thus B would have rejected it (with the same probability). In this case B's output-bit is always reject. Since C' sends $B_{\not\ni k}$ an invalid MAC value in this case, $B_{\not\ni k}$ also always rejects.

We have shown that when the result of C''s verification is the same as B, then $B_{\not\ni k}$'s accept/reject bit is the same as B's. Given that the probability that C' accepts the proof is equal to the probability that B accepts it, we have that B and $B_{\not\ni k}$'s output-bits equal accept with the same probability. This means that C's view is identical in both cases and this completes the proof. ■

We now show that when C interacts with $B_{\not\ni k}$, modifying A to $A_{\not\ni k}$ makes no difference to his view. This is done by showing how the proof from A can be simulated by C himself. A key observation regarding $B_{\not\ni k}$ is that the number of messages it sends is strictly less than the number of iterations of the zero-knowledge proof that takes place in the (A, C) execution.

Lemma 7.2 *Let $A_{\not\ni k}$ and $B_{\not\ni k}$ be the above-defined modified parties. Then, for every ppt channel C there exists a ppt channel C' such that for every randomized process $z = Z(Q, w)$, every polynomial $p(\cdot)$ and all sufficiently large n's*

$$\left| Pr_{Q,w}[\mathrm{Expt}_z^{A(Q,w),B_{\not\ni k}(w)}(C) = 1] - Pr_{Q,w}[\mathrm{Expt}_z^{A_{\not\ni k}(Q,w),B_{\not\ni k}(w)}(C') = 1] \right| < \frac{1}{p(n)}$$

Proof: In both experiments, $B_{\not\ni k}$ does not participate in the zero-knowledge proof. As we have described above (in the motivating discussion), this enables A's zero-knowledge proof to be simulated for C, who is the verifier. We now formally show how C' executes this simulation for C.

The key observation is that the number of iterations in the first part of the RK-proof is m, whereas the number of messages sent between C and $B_{\not\ni k}$ is $m - t$. Therefore, there are t iterations for which no message is sent between C and $B_{\not\ni k}$ (these iterations may not be fixed but rather can be determined by C' during the execution). In these iterations, since $B_{\not\ni k}$ is not active, C' is able to rewind C. The RK-proof is such that if the verifier can be rewound for any iteration during the first part, then a successful simulation of the proof is achieved.

To see why the above holds, recall that the RK-proof consists of two parts. The first part consists of m iterations, where in each iteration the verifier (who is C in this case) sends the prover a commitment to a random string v_i. The prover then sends a commitment to p_i and the verifier decommits. In the second part of the proof, the prover proves (with a witness-indistinguishable proof [20]) that either there exists an i such that $p_i = v_i$ or that the "target" statement is correct. In a real proof, the prover will not be able to set $p_i = v_i$ except with negligible probability. This then implies that the statement is correct. On the other hand, if there is just one iteration of the first part in which the simulator can rewind the verifier, he can then set $p_i = v_i$ (because he rewinds after obtaining the decommitment value v_i and can thus set his commitment p_i to equal v_i). In this case, he can successfully prove the witness-indistinguishable proof (without knowing a proof of the target 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. 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 the same as in a real execution. This prevents us from simply halting if after a rewind, C refuses to decommit (since before rewinding he did decommit).

Before we continue, we define the concepts of *promising* and *successful* iterations. Through this, we differentiate what happens during the first execution of a given iteration (i.e., before any rewinding when the verifier receives a commitment to a random string p_i) and during repeated executions after rewinding (when the verifier receives a commitment to p_i such that $p_i = v_i$). That is,

- An iteration i is called *promising* if when C receives a commitment to a random p_i, the iteration is such that no messages are sent to $B_{\not\approx k}$ and C decommits properly. (This occurs before any rewinding.)
- 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 $B_{\not\approx k}$ and C decommits properly. (This typically occurs after rewinding when p_i can be set to v_i.)

Now, notice that when an 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 $\mathcal{NP}$-witness) can prove the necessary witness-indistinguishable proof. Loosely speaking, the probability that a *promising* iteration is not *successful* must be negligible. This is because the only difference between the two cases is whether C receives a commitment to p_i or v_i. Now, assume that there is a verifier V^* for whom the probabilities that an iteration is promising or successful are non-negligibly far apart. Then, V^* can be used to distinguish a commitment to p_i from a commitment to v_i, contradicting the security of the commitment scheme. This point is crucial because unless C refuses to decommit before any rewinding, we know that there must be at least t promising iterations. We can conclude that with overwhelming probability, some of these are also 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 he 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 and in this case C' halts the simulation; (2) C sent a message to $B_{\not\approx k}$ during the iteration – in this case C' simply continues to the next iteration.

On the other hand, if an iteration *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 rewinded iteration is successful, then as we have shown C' can complete the simulation successfully. However, the iteration may not be successful after the rewinding. That is, C may refuse to decommit or may send messages to $B_{\not\approx k}$. As long as the rewinded iteration is not successful, C' continues to rewind up to N times (where $N = poly(n)$ and the exact polynomial taken is discussed in the analysis). If none of the rewinds were successful then he resends his original commitment to a random p_i and continues to the next iteration. We note that each rewinding is independent in that C' sends an independent random commitment to $p_i = v_i$ each time.

We stress that C' must block any message sent by C to $B_{\not\approx k}$ during a rewinding. This is because C cannot be rewound beyond a point in which he sent a message to $B_{\not\approx k}$. However, since C may refuse to decommit, further rewindings may be necessary. Thus, in the case that C sends a message to $B_{\not\approx k}$ during a rewinding, C' halts the iteration (without forwarding the message) and rewinds again, up to N times.

Motivation for the Analysis: As we have mentioned, if an iteration is not promising because C refused to decommit, then the simulation terminates successfully (as the prover would also simply halt in a real proof). On the other hand, we know that there are at least t iterations for which C does not send any messages to $B_{\not\ni k}$ (recall that there are only $m-t$ messages sent between C and $B_{\not\ni k}$). We therefore have at least t promising iterations (or C refused to decommit and anyway the simulation succeeds). The simulation fails only if all these promising iterations are not successful; we show that for a correct choice of N (the number of rewindings of a promising iteration), this occurs with at most negligible probability.

The Analysis: 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 $B_{\not\ni k}$ during iteration i when p_i is a random commitment (i.e., before rewinding).
- Let $Y_1, \ldots, Y_m$ be random variables such that $Y_i = 1$ if and only if C agrees to decommit during iteration i when p_i is a random commitment (i.e., before rewinding).

  We therefore have that an iteration i is *promising* if $X_i = Y_i = 1$. We now introduce similar notations for iterations after rewinding:

- Let $X'_1, \ldots, X'_m$ be random variables such that $X'_i = 1$ if and only if C sends no messages to $B_{\not\ni k}$ during iteration i when p_i is a commitment such that $p_i = v_i$ (i.e., typically after rewinding).
- Let $Y'_1, \ldots, Y'_m$ be random variables such that $Y'_i = 1$ if and only if C agrees to decommit during iteration i when p_i is a commitment such that $p_i = v_i$ (i.e., typically after rewinding).

  We therefore have that an iteration i is *successful* if $X'_i = Y'_i = 1$.

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

$$\Pr[Y_1 \cdots Y_{i-1} = 1^{i-1} \,\&\, X_1 \cdots X_{i-1} = \alpha] < \frac{1}{q(n)} \tag{21}$$

or

$$\begin{array}{ll} \text{if} & \Pr[X_i = Y_i = 1 \,|\, Y_1 \cdots Y_{i-1} = 1^{i-1} \,\&\, X_1 \cdots X_{i-1} = \alpha] \ge \frac{1}{n} \\ \text{then} & \Pr[X'_i = Y'_i = 1 \,|\, Y_1 \cdots Y_{i-1} = 1^{i-1} \,\&\, X_1 \cdots X_{i-1} = \alpha] > \frac{1}{2n} \end{array} \tag{22}$$

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 iterations, and obtains v_i from the verifier decommitment in the ith iteration, in case such an event has occured. The second algorithm is given the view of the first algorithm along with a challenge commitment and distinguishes the case in which this commitment is to a random value from the case this commitment is to the value v_i. ■

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. Denoting the simulator's failure event by `fail`, we have:

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

where S is any maximal prefix-free subset of $\{0,1\}^m$. (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 every $\alpha \in \{0,1\}^m$ has a prefix in S (or else this α could have been added to S without violating the prefix-free condition).) The last equality holds since the strings in $\{0,1\}^m$ can be partitioned to subsets such that the strings in each subset have a unique prefix in the set S.

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. 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. Then $S_1 \cup S_2$ is a maximal prefix-free subset of $\{0,1\}^m$, and so we have:

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

where the last equality follows since $\Pr[\mathtt{fail} \,\&\, X \in S_2] = 0$ (i.e., unless C refuses to properly decommit in some iteration, in which case the simulation never fails, there must be at least $t \geq k+1$ iterations/indices i in which $X_i = 1$ holds). Since $|H_k| < m^k$, we have

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

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

$$\Pr[\mathtt{fail} \,\&\, X_1 \cdots X_{|\alpha'|} = \alpha'] < \frac{1}{m^k \cdot p(n)} \tag{23}$$

which establishes our claim that the simulation fails with probability smaller than $1/p(n)$.

In order to establish Eq. (23), we fix an arbitrary $\alpha' \in H_k$, let $i = |\alpha'| + 1$, and we consider two cases:

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

Case 2: $\Pr[Y_1 \cdots Y_{i-1} = 1^{i-1} \,\&\, X_1 \cdots X_{i-1} = \alpha'] \geq \frac{1}{m^k \cdot p(n)}$. In this case, setting $q(n) = m^k \cdot p(n)$, we conclude that Eq. (22) holds. Furthermore, for every $j \leq i$, it holds that $\Pr[Y_1 \cdots Y_{j-1} = 1^{j-1} \,\&\, X_1 \cdots X_{j-1} = \alpha''] \geq \frac{1}{m^k \cdot p(n)}$ holds, where α'' is the the $(j-1)$-bit long prefix of α'. Thus, Eq. (22) holds for α'' too. We are particularly interested in prefices α'' such that $\alpha''1$ is a prefix of α'. We know that there are k such prefices $\alpha''1$ 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. (22)) 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 $\Pr[\mathtt{fail} \,\&\, X_1 \cdots X_{i-1} = \alpha'] < 2^{-n} < \frac{1}{m^k \cdot p(n)}$ 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 $j \in J$)

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

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

Thus, we have established the desired bound of Eq. (23) in all possible cases.

Concluding the (A, C) Simulation: Following the zero-knowledge proof, $A_{\not\approx k}$ sends a MAC of the entire session-transcript. The channel C' must ensure that C receives a MAC that is indistinguishable from the MAC that he would have received from A. Recall that by the definition of the $A_{\not\approx k}$ modification, the party $A_{\not\approx k}$ expects to receive a string s in place of the zero-knowledge proof. C' defines s to be the transcript of the zero-knowledge simulation. This means that $A_{\not\approx k}$'s resulting message-transcript 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 sent by $A_{\not\approx k}$ is indistinguishable from one that A would have sent. This completes the proof. ■

7.2 Corollaries and Remarks

The above proof is *identical* for a party B_2 who does not participate at all in the validation stage. We now restate Lemma 7.2 in this case (this is used in Section 10).

Lemma 7.4 *Let $A_{\not\ni k}$ be the above-defined modified party and let B_2 be a party who halts before the validation stage. Then, for every ppt channel C there exists a ppt channel C' such that for every randomized process $z = Z(Q,w)$, every polynomial $p(\cdot)$ and all sufficiently large n's*

$$\left|Pr_{Q,w}[\mathrm{Expt}_z^{A(Q,w),B_2(w)}(C) = 1] - Pr_{Q,w}[\mathrm{Expt}_z^{A_{\not\ni k}(Q,w),B_2(w)}(C') = 1]\right| < \frac{1}{p(n)}$$

An immediate corollary from Lemmas 7.1 and 7.2 is that if we modify *both* A and B to $A_{\not\ni k}$ and $B_{\not\ni k}$ respectively, then this has at most a negligible affect on C's output.

Corollary 7.5 *Let $A_{\not\ni k}$ and $B_{\not\ni k}$ be the modified parties defined above. Then, for every ppt channel C there exists a ppt channel C' such that for every randomized process $z = Z(Q,w)$, every polynomial $p(\cdot)$ and all sufficiently large n's*

$$\left|Pr_{Q,w}[\mathrm{Expt}_z^{A(Q,w),B(w)}(C) = 1] - Pr_{Q,w}[\mathrm{Expt}_z^{A_{\not\ni k}(Q,w),B_{\not\ni k}(w)}(C') = 1]\right| < \frac{1}{p(n)}$$

A Note on the Number of Rounds: Our simulator works given that the number of rounds in the first part of the RK-proof is any non-constant function of the security parameter n (say $\log\log n$). We note that if only an expected (rather than strictly) polynomial-time simulator is desired, then a single additional round suffices. This can be shown using the techniques of [25].

Pseudorandomness of $Q(w)$ Restated: We now restate Theorem 4.2 in the case that C interacts with A_2 and $B_{\not\ni k}$, rather than with A_2 and B (recall that A_2 is a party that halts before the validation stage). The restated theorem is used for the session-key secrecy (proved in Section 8), and is presented here only due to the definition of the modified party $B_{\not\ni k}$.

Theorem 7.6 (Pseudorandomness of $Q(w)$ with $B_{\not\ni k}$): *Let C be an arbitrary ppt adversary interacting with A_2 and $B_{\not\ni k}$. Then, for every polynomial $p(\cdot)$ and for all sufficiently large n's*

$$\left|Pr_{Q,w}[\mathrm{Expt}_{Q(w)}^{A_2(Q,w),B_{\not\ni k}(w)} = 1] - Pr_{Q,w,U_n}[\mathrm{Expt}_{U_n}^{A_2(Q,w),B_{\not\ni k}(w)} = 1]\right| < 2\epsilon + \frac{1}{p(n)}$$

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

The proof of this theorem is identical to the proof of Theorem 4.2.

8 Full Proof of Session-Key Secrecy

Theorem 4.2 states that $Q(w)$ is $(1-O(\epsilon))$-pseudorandom prior to the validation stage of the (A,C)-execution. In this section we prove that the session-key $k_2(Q(w))$ is $(1-O(\epsilon))$-pseudorandom at the conclusion of the entire protocol. 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 consider 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 itself. Therefore, this can only "help" C distinguish the session-key from a random string.

The proof relies 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, as in our case the generator is seeded by $Q(w)$ which is only $(1 - 2\epsilon)$-pseudorandom, whereas a generator is usually seeded by a uniformly random string. In the following proposition we show that if $Q(w)$ is $(1 - 2\epsilon)$-pseudorandom (as previously shown), then given $f^{2n}(Q(w))$ and $k_1(Q(w))$, the string $k_2(Q(w))$ is $(1 - 4\epsilon)$-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 the validation stage, the output session-key $k_2(Q(w))$ is still $(1 - O(\epsilon))$-pseudorandom. We also show that given $f^{2n}(Q(w))$, the string $k_1(Q(w))$ is $(1 - O(\epsilon))$-pseudorandom. This means that the MAC-key is $(1 - O(\epsilon))$-pseudorandom even after A sends $f^{2n}(Q(w))$. The validation stage also contains a zero-knowledge proof and we deal with this later.

Preliminaries

We model any information that C may have learned about Q and w during the protocol by a random process $I(\cdot)$. This can be seen by defining $I((Q, w))$ to equal the output of $C_1^{A_2(Q,w), B_{\not\approx k}(w)}(1^n)$. (The reason we define the random process over A_2 and $B_{\not\approx k}$, rather than A and B, will become evident later.) Now, we model the inputs Q and w by a random variable Y_n and the value $Q(w)$ by a *related* random variable X_n (i.e., for $Y_n = (\alpha, \beta)$, the random variable X_n is defined to be $\alpha(\beta)$). Then, the fact that $Q(w)$ is $(1-2\epsilon)$-pseudorandom with respect to C's view after interacting with A_2 and $B_{\not\approx k}$ (as stated in Theorem 7.6), is represented by the saying that X_n is $(1 - 2\epsilon)$-pseudorandom to a distinguisher given $I(Y_n)$.

Proposition 8.1 *Let $\{X_n\}$ and $\{Y_n\}$ be (possibly) related random variables such that $\{X_n\}$ is $(1 - \delta)$-pseudorandom to a distinguisher given $I(Y_n)$. Then*

- $(I(Y_n), f^{2n}(X_n), k_1(X_n), k_2(X_n))$ *is $(1-2\delta)$-indistinguishable from $(I(Y_n), f^{2n}(X_n), k_1(X_n), U_n)$, and*
- $(I(Y_n), f^{2n}(X_n), k_1(X_n))$ *is $(1 - 2\delta)$-indistinguishable from $(I(Y_n), f^{2n}(X_n), U_n)$.*

Proof: We begin by showing that $(I(Y_n), f^{2n}(X_n), k_1(X_n))$ is $(1 - 2\delta)$-indistinguishable from $(I(Y_n), f^{2n}(X_n), U_n)$. This is shown in three steps ($\overset{\delta}{\equiv}$ denotes $(1 - \delta)$-indistinguishability and $\overset{c}{\equiv}$ denotes computational indistinguishability):

1. $(I(Y_n), f^{2n}(X_n), k_1(X_n)) \overset{\delta}{\equiv} (I(Y_n), f^{2n}(U_n), k_1(U_n))$

   This is because by the hypothesis $(I(Y_n), X_n)$ is $(1 - \delta)$-indistinguishable from $(I(Y_n), U_n)$.

2. $(I(Y_n), f^{2n}(U_n), k_1(U_n)) \overset{c}{\equiv} (I(Y_n), f^{2n}(U_n^{(1)}), U_n^{(2)})$ (where $U_n^{(1)}$ and $U_n^{(2)}$ are two independent uniform distributions)

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

3. $(I(Y_n), f^{2n}(U_n^{(1)}), U_n^{(2)}) \overset{\delta}{\equiv} (I(Y_n), f^{2n}(X_n), U_n^{(2)})$

   As in the first step, this is because $(I(Y_n), X_n)$ is $(1 - \delta)$-indistinguishable from $(I(Y_n), U_n)$.

Putting it all together we have that $(I(Y_n), f^{2n}(X_n), k_1(X_n))$ and $(I(Y_n), f^{2n}(X_n), U_n)$ are $(1-2\delta)$-indistinguishable.

An analogous argument is used to show that $(I(Y_n), f^{2n}(X_n), k_1(X_n), k_2(X_n))$ is $(1-2\delta)$-indistinguishable from $(I(Y_n), f^{2n}(X_n), k_1(X_n), U_n)$. ■

Session-Key Secrecy w.r.t. $A_{\not k}$ and $B_{\not k}$

We now show that when C interacts with $A_{\not k}$ and $B_{\not k}$ (where $A_{\not k}$ and $B_{\not k}$ are as defined in Section 7), the session-key is $(1-O(\epsilon))$-pseudorandom.

Corollary 8.2 *Let C be an arbitrary ppt channel interacting with $A_{\not k}$ and $B_{\not k}$, as defined in Section 7. Then, for every polynomial $p(\cdot)$ and all sufficiently large n's*

$$\left|Pr_{Q,w}[\mathrm{Expt}_{k_2(Q(w))}^{A_{\not k}(Q,w),B_{\not k}(w)}=1]-Pr_{Q,w,U_n}[\mathrm{Expt}_{U_n}^{A_{\not k}(Q,w),B_{\not k}(w)}=1]\right|<4\epsilon+\frac{1}{poly(n)}$$

Proof: Theorem 7.6 (from Section 7) states that for any ppt channel C interacting with A_2 and $B_{\not k}$

$$\left|Pr_{Q,w}[\mathrm{Expt}_{Q(w)}^{A_2(Q,w),B_{\not k}(w)}=1]-Pr_{Q,w,U_n}[\mathrm{Expt}_{U_n}^{A_2(Q,w),B_{\not k}(w)}=1]\right|<2\epsilon+\frac{1}{poly(n)}$$

That is, the string $Q(w)$ is $(1-2\epsilon)$-pseudorandom with respect to C's view at the conclusion of the protocol execution. Now, the only difference between A_2 and $A_{\not k}$ is that $A_{\not k}$ sends the following two messages in the validation stage: $f^{2n}(Q(w))$ and the MAC-key $k_1(Q(w))$. Using the notation of Proposition 8.1, the channel C's view of the execution $C_1^{A_{\not k},B_{\not k}}(1^n)$ can be represented by $(I(Q,w),f^{2n}(Q(w)),k_1(Q(w)))$ (define $I(Q,w)\stackrel{\text{def}}{=}C_1^{A_2(Q,w),B_{\not k}(w)}(1^n)$). Now, Proposition 8.1 states that $(I(Q,w),f^{2n}(Q(w)),k_1(Q(w)),k_2(Q(w)))$ is $(1-4\epsilon)$-indistinguishable from $(I(Q,w),f^{2n}(Q(w)),k_1(Q(w)),U_n)$. In other words,

$$\left|Pr_{Q,w}[\mathrm{Expt}_{k_2(Q(w))}^{A_{\not k}(Q,w),B_{\not k}(w)}=1]-Pr_{Q,w,U_n}[\mathrm{Expt}_{U_n}^{A_{\not k}(Q,w),B_{\not k}(w)}=1]\right|<4\epsilon+\frac{1}{poly(n)}$$

That is, the corollary is obtained by combining Theorem 7.6 with Proposition 8.1. ■

Session-Key Secrecy w.r.t. A and B

It remains to show that when C interacts with A and B (and not the modified parties $A_{\not k}$ and $B_{\not k}$), then the session-key $k_2(Q(w))$ is $(1-O(\epsilon))$-pseudorandom. This is immediately derived from Corollary 7.5 that states that for every ppt channel C there exists a ppt channel C' such that for every randomized process $z=Z(Q,w)$

$$\left|Pr_{Q,w}[\mathrm{Expt}_z^{A(Q,w),B(w)}(C)=1]-Pr_{Q,w}[\mathrm{Expt}_z^{A_{\not k}(Q,w),B_{\not k}(w)}(C')=1]\right|<\frac{1}{poly(n)}$$

By applying Corollary 7.5 twice to Corollary 8.2, replacing z once with $k_2(Q(w))$ and once with U_n, we have the following theorem for *session-key secrecy*.

Theorem 8.3 (Theorem 4.5 restated): *Let C be an arbitrary ppt channel. Then,*

$$\left|Pr_{Q,w}[\mathrm{Expt}_{k_2(Q(w))}^{A(Q,w),B(w)}=1]-Pr_{Q,w,U_n}[\mathrm{Expt}_{U_n}^{A(Q,w),B(w)}=1]\right|<4\epsilon+\frac{1}{poly(n)}$$

Divertion: Security of the MAC Value

We now divert and prove a corollary needed for Sections 4.6 and 10 (password secrecy). We prove it here because its proof is almost identical to that of Theorem 8.3. Recall that the last message sent by A in the protocol is a MAC (implemented via a pseudorandom function keyed with $k_1(Q(w))$) applied to her entire message transcript. The corollary states that for any other string t, the value $MAC_{k_1(Q(w))}(t)$ is $(1-4\epsilon)$-pseudorandom with respect to C's view.

Corollary 8.4 (Corollary 4.9 restated): *Let C be an arbitrary ppt channel. Then, for every string t that differs from the (A,C)-message transcript, the value $MAC_{k_1(Q(w))}(t)$ is $(1-4\epsilon)$-pseudorandom with respect to C's view.*

Proof: In the proof of Theorem 8.3 we show that $k_2(Q(w))$ is $(1-4\epsilon)$-pseudorandom with respect to C's view. Using the same argument, we have that *before* A sends the MAC in the validation stage, the MAC-key $k_1(Q(w))$ is $(1-4\epsilon)$-pseudorandom with respect to C's view. Formally, consider a modified party $A_{\not{m}ac}$ who is exactly the same as A except that she does not send the MAC in the validation stage. Then, the same proof as above can be used to show that

$$\left|Pr_{Q,w}[\mathrm{Expt}_{k_1(Q(w))}^{A_{\not{m}ac}(Q,w),B(w)}=1]-Pr_{Q,w,U_n}[\mathrm{Expt}_{U_n}^{A_{\not{m}ac}(Q,w),B(w)}=1]\right|<4\epsilon+\frac{1}{poly(n)} \qquad (24)$$

(Recall that this is proved in two stages. We first consider a scenario in which C interacts with a party $A_{\not{k},\not{m}ac}$, who sends neither the MAC or zero-knowledge proof, and with $B_{\not{k}}$. In this scenario, we show that $k_1(Q(w))$ is $(1-4\epsilon)$-pseudorandom with respect to C's view. Then, by applying Corollary 7.5, we have that the above holds also when C interacts with $A_{\not{m}ac}$ and B.)

In the proposition below we show that Equation (24) implies that the MAC used is a $(1-4\epsilon)$-pseudorandom function. Then, even when C is given the MAC for the (A,C)-message transcript, all other MAC values are $(1-4\epsilon)$-pseudorandom with respect to his view. This completes the proof as the only difference between A and $A_{\not{m}ac}$ is that A sends a MAC on the (A,C)-message transcript in the validation stage.

It remains to prove that the MAC used is a $(1-4\epsilon)$-pseudorandom function. We use the same notation as in Proposition 8.1. (Recall that $I(\cdot)$ is a random process that represents the information learned by C during a protocol execution, Y_n is a random variable taking on pairs (Q,w) and X_n is a related random variable taking on values $Q(w)$.)

Proposition 8.5 *Let $\{X_n\}$ and $\{Y_n\}$ be (possibly) related random variables such that $\{X_n\}$ is $(1-\delta)$-pseudorandom to a distinguisher given $I(Y_n)$. Furthermore, let $g_r(\cdot)$ be a pseudorandom function when r is uniformly distributed. Then, given $I(Y_n)$, the function $g_{X_n}(\cdot)$ is $(1-\delta)$-pseudorandom.*

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

Let D be a ppt oracle machine who receives the output of the random process $I(Y_n)$ and oracle access to either g_{X_n} or a random function f. Then,

$$\begin{aligned}
&\left|Prob[D^{g_{X_n}}(I(Y_n),1^n)=1]-Prob[D^f(I(Y_n),1^n)=1]\right| \\
&\quad\leq\ \left|Prob[D^{g_{X_n}}(I(Y_n),1^n)=1]-Prob[D^{g_{U_n}}(I(Y_n),1^n)=1]\right| && (25)\\
&\qquad+\left|Prob[D^{g_{U_n}}(I(Y_n),1^n)=1]-Prob[D^f(I(Y_n),1^n)=1]\right| && (26)
\end{aligned}$$

Equation (26) is negligible by the definition of a pseudorandom function. On the other hand, Equation (25) must be less than $\delta + \frac{1}{poly(n)}$ because otherwise a ppt machine D' that, given $I(Y_n)$ attempts to distinguish X_n from U_n, can invoke D on input $(I(Y_n), 1^n)$ and answer all oracle queries according to its input string (which is either X_n or U_n). ■

■

9 Full Proof of the Key-Match Requirement

The key-match requirement captured in Theorem 4.6 states that the probability that A and B both accept, yet have different keys is at most $O(\epsilon)$. Recall that $\tau_A \stackrel{\text{def}}{=} Q(w)$ and that τ_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 $\tau_A \neq \tau_B$ is at most $O(\epsilon)$. (In fact, the first scheduling is such that B accepts with probability at most $O(\epsilon)$, irrespective of whether or not $\tau_A = \tau_B$.)

9.1 Proof of Lemma 4.7

Lemma 9.1 (Lemma 4.7 – 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*

$$Pr[B = \text{acc} \ \wedge \ \text{Case 1}] < 2\epsilon + \frac{1}{p(n)}$$

Proof: As in our previous proofs, we reduce the concurrent setting to a two-party stand-alone setting. However, before doing this we remove the zero-knowledge proofs and modify parties A and B to $A_{\not{z}k}$ and $B_{\not{z}k}$ respectively, as defined in Section 7 (loosely speaking, the modified parties act exactly as A and B but do not participate in the zero-knowledge proofs). Corollary 7.5 states that for every channel C interacting with A and B, there exists a channel C' interacting with $A_{\not{z}k}$ and $B_{\not{z}k}$ such that the channels' views in the two cases are indistinguishable. Since B's accept/reject bit is part of C's view, we have that the probability that $B_{\not{z}k}$ 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). We therefore continue by proving the theorem in the setting whereby C interacts with $A_{\not{z}k}$ and $B_{\not{z}k}$ (rather than with A and B).

As mentioned, our first step now is to reduce the concurrent setting to a two-party stand-alone setting. In previous proofs this was done by having C simulate one of the concurrent executions with A or B. For example, C would internally simulate the (C, B)-execution while interacting with A. The reduction here is different in that a party P will incorporate parts of *both* the (A, C) and (C, B) executions. 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 $(A_{\not{z}k}, C)$-evaluation terminates before the $(C, B_{\not{z}k})$-evaluation begins. Consider now a simplified case in which the entire $(A_{\not{z}k}, C)$-protocol consists only of a single polynomial evaluation; likewise for the $(C, B_{\not{z}k})$-protocol. Then, when the scheduling is as mentioned, a party P, can execute two sequential polynomial evaluations with C; in the first he plays $A_{\not{z}k}$'s role and in the second he plays $B_{\not{z}k}$'s role. That is, when this scheduling occurs the above two-party setting perfectly simulates the concurrent setting.

The actual reduction is more complex as the $(A_{\not k}, C)$ and $(C, B_{\not k})$ 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 are internally simulated by C and some are played by 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 $B_{\not k}$'s role in the $(C, B_{\not k})$-validation stage. What remains is $B_{\not k}$'s role in the $(C, B_{\not k})$-commitment stage and $A_{\not k}$'s role in the $(A_{\not k}, C)$-validation stage; these are internally simulated by C. The following table shows which party (P or C) simulates $A_{\not k}$ and $B_{\not k}$'s respective roles.

Stage	$A_{\not k}$	$B_{\not k}$
1. Commitment	P	C
2. Pre-Key Exchange	P	P
3. Validation	C	P

Party P's input consists of Q and w and this therefore enables him to play $A_{\not k}$ and $B_{\not k}$'s roles, as required. We also give C some auxiliary input that enables him to internally simulate the remaining parts of the execution.

This reduction makes sense when the scheduling of Case 1 occurs. Loosely speaking, we show that according to this scheduling, the two-party protocol between P and C accurately simulates our concurrent setting. (When Case 1 does not occur, then nothing can be said about the (P, C) protocol. However, for the lemma we need to bound $B_{\not k}$'s accepting probability in Case 1 only. This is therefore enough.)

We now present the protocol for parties P and C; the protocol is specifically designed to simulate the concurrent $C_1^{A_{\not k}(Q,w),B_{\not k}(w)}$ setting, according to the scheduling of Case 1. What we show is that every adversary C_1 in the concurrent setting can be "simulated" (in some adequate sense) by an adversary C' to the following protocol.

Protocol-(P, C):

Input:

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

The Protocol:

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 $c_1 = \text{Commit}(Q) = C(Q, r_1)$ 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. 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 $(C, B_{\not k})$-execution (pre-key exchange):*
   - C sends P a commitment $c_2 = C(Q_C, r_2)$ for some polynomial Q_C and a random r_2.

- 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 τ (which equals either $Q_C(w)$ or $\perp$) from the evaluation.

4. *Partial Emulation of Stage 3 of the $(C, B_{\not\approx k})$-execution (validation stage):*

   - C sends a string y to P, and P outputs accept if and only if $y = f^{2n}(\tau)$.

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 $B_{\not\approx k}$ accept in the concurrent protocol with the scheduling of Case 1, can be used by a party C' to succeed with at least the same probability in the above protocol with P.

Claim 9.2 *Let C be a ppt channel interacting with $A_{\not\approx k}$ and $B_{\not\approx k}$. Then there exists a ppt party C' interacting with P in Protocol-(P, C') such that*

$$Pr_{Q,w}[P = \text{acc}] \geq Pr_{Q,w}[B_{\not\approx k} = \text{acc} \;\wedge\; \text{Case 1}]$$

Proof: We begin by considering a modification of party $B_{\not\approx k}$ to B' who ignores the MAC sent to him in the validation stage. That is, B' is the same as B except that he decides whether to accept or reject based solely on the y-string he receives in the validation stage. As B' only omits checks, we have that

$$Pr_{Q,w}[B' = \text{acc} \;\wedge\; \text{Case 1}] \geq Pr_{Q,w}[B_{\not\approx k} = \text{acc} \;\wedge\; \text{Case 1}]$$

We continue by proving that for every C interacting with $A_{\not\approx k}$ and B', there exists a C' interacting with P such that

$$Pr_{Q,w}[B' = \text{acc} \;\wedge\; \text{Case 1}] = Pr_{Q,w}[P = \text{acc}]$$

The party C' incorporates C internally and *perfectly* simulates the concurrent setting with $A_{\not\approx k}$ and B' for C (i.e., $C_1^{A_{\not\approx k}, B'}$). First notice that Step (4) of the (P, C') protocol constitutes the full validation stage of the (C, B')-protocol (whereas it is only partial for the $(C, B_{\not\approx k})$-protocol). This means that the (P, C') protocol contains all stages of the $(A_{\not\approx k}, C)$ and $(C, B_{\not\approx k})$ protocols *except* for the the first stage of the $(C, B_{\not\approx k})$-protocol and the third stage of the $(A_{\not\approx k}, C)$-protocol. As mentioned, 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 $C^{A_{\not\approx k}(Q,w), B'(w)}$ setting for him. This involves separately simulating the $(A_{\not\approx k}, C)$ and (C, B') executions. This is done as follows (recall that C fully controls the scheduling):

- *The $(A_{\not\approx k}, C)$ 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 $A_{\not\approx k}$ to P and likewise, messages from P are forwarded to C.
  2. *Stage 3:* C' internally emulates $A_{\not\approx k}$'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 $(A_{\not\approx k}, C)$ session-transcript keyed by $k_1(Q(w))$. Party C' can send these messages since he knows $Q(w)$ and can therefore 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. 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.
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:
   - *Failure Case:* If this first message was sent **before** the completion of Stage 2 of the $(A_{\not k}, C)$ execution, then C' halts (the simulation fails).
   - *Success Case:* If this first message was sent **after** the completion of Stage 2 of the $(A_{\not k}, C)$ execution, 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. We begin by noting 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 not successful (i.e., Case 1 did not occur) then P never accepts. On the other hand, when the simulation succeeds P accepts with the same probability as B' would have. 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. ■

It remains to bound the probability that P accepts in the above (P, C')-protocol.

Claim 9.3 *For every ppt party C' interacting with P in Protocol-(P, C') we have that for every polynomial $p(\cdot)$ and all sufficiently large n's*

$$Pr[P = \text{acc}] < 2\epsilon + \frac{1}{p(n)}$$

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 can be analyzed by directly considering the security of the different modules. (We stress that this protocol's connection to the concurrent execution of our protocol with A,B and C has already been established in Claim 9.2 and is not relevant in the analysis here.)

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 (this replacement is possible since the commitment is not used anywhere in the continuation of the protocol). 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, or $\perp$. By the security of the evaluation, C' receives $Q(w_C)$ (or $\perp$) and nothing more.
- *(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 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 he (possibly) received from the first evaluation (and his auxiliary input $Q(w)$). Now, if $w_C \neq w$, then the only thing that party C' learns about w is that it does not equal w_C. This is due to the pairwise independence 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 as nothing is revealed of w during the protocol. (Note that C can indeed determine whether or not $w_C = w$ by comparing $Q(w_C)$ to his auxiliary input $Q(w)$.)

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 [14], 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. We differentiate between the case that $w_C = w$ and $w_C \neq w$ and separately upper bound the following probabilities:

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

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 sending w_C is a commitment to the polynomial Q. That is, $\hat{C}'$'s entire view at this point consists of his auxiliary input $Q(w)$ and Commit(Q). Due to the hiding property of the commitment, Commit(Q) can be replaced by 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 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 $Pr[w_C = w] < \epsilon + \mu$. Therefore

$$Pr[\hat{P} = \text{acc} \ \wedge \ w_C = w] \leq Pr[w_C = w] \leq \epsilon + \mu$$

Bounding the probability that $\hat{P} = \mathsf{acc}$ and $w_C \neq w$: We first 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 his 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 $w_C \neq w$.

The continuation of the protocol involves $\hat{C}'$ sending a polynomial Q_C for the second polynomial evaluation and then 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 his 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 $(Q_C, f^{2n}(Q_C(w)))$ is at most the probability that he 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,

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

We therefore conclude that in an ideal execution

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

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

The lemma follows by combining Claims 9.2 and 9.3. ■

9.2 Proof of Lemma 4.8

Lemma 9.4 (Lemma 4.8 – 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 for every polynomial $p(\cdot)$ and for all sufficiently large n's,*

$$Pr[B = \text{acc} \ \wedge \ \text{Case 2} \ \wedge \ \tau_A \neq \tau_B] < \epsilon + \frac{1}{p(n)}$$

Proof: The proof of this lemma relies on the non-malleability of the commitment sent in the commitment stage of the protocol. As was described in the proof sketch, in the case that $\tau_A \neq \tau_B$, 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. (We note that in the case that $(Q', w') = (Q, w)$, party B rejects with overwhelming probability, unless $\tau_A = \tau_B$. This is because the validation stage enforces that $\tau_B = Q'(w')$ and by the hypothesis $Q'(w') = Q(w) = \tau_A$.) 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.

As described, unless $\tau_A = \tau_B$, the channel C can only make B accept if he generates a non-malleable commitment to (Q', w) where $Q' \neq Q$. To instantiate the above intuition, we define a relation R as follows (recall that the non-malleable commitment value sent by A is (Q, w) and denote the one received by B by (Q', w')). Define $R \subset \{0,1\}^{3n} \times \{0,1\}^{3n}$ such that $((Q, w), (Q', w')) \in R$ if and only if $(Q', w') \neq (Q, w)$ and $w' = w$. That is, C "succeeds" with respect to R (and thus B may accept) if C does not copy A's commitment and yet the second element of the commitment is the correct password.

We consider the probability that B accepts in Case 2 and $\tau_A \neq \tau_B$ 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): $Pr[B = \text{acc} \ \wedge \ \text{Case 2} \ \wedge \ \tau_A \neq \tau_B \ \wedge \ ((Q, w), (Q', w')) \in R] < \epsilon + \frac{1}{poly(n)}$
2. (Fail Case): $Pr[B = \text{acc} \ \wedge \ \text{Case 2} \ \wedge \ \tau_A \neq \tau_B \ \wedge \ ((Q, w), (Q', w')) \notin R] < \frac{1}{poly(n)}$

The lemma follows by combining B's accepting probability in the above two cases. 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 9.5 (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. Then for every polynomial $p(\cdot)$ and all sufficiently large n's*

$$Pr[\text{Case 2} \ \wedge \ ((Q, w), (Q', w')) \in R] < \epsilon + \frac{1}{p(n)}$$

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, the (A, C) pre-key exchange may run concurrently to the (B, C) commitment. The key point in this 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 is comprised of the polynomial Q only. Therefore, if C has Q, then he can perfectly emulate this stage himself (this is true irrespective of the security of the modules making up the pre-key exchange stage of the protocol). Fortunately, Claim 9.5 holds *even* if C is explicitly given Q. Thus, we prove that for every ppt channel C given auxiliary input Q, it holds that

$$Pr[\text{Case 2} \ \wedge \ ((Q, w), (Q', w')) \in R] < \epsilon + \frac{1}{poly(n)}$$

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. Since the (A, C) pre-key exchange is simulatable (by C), 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.[29] 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

$$\left|Pr[((Q,w),(Q',w')) \in R] - Pr[((Q,w),(\hat{Q}',\hat{w}')) \in R]\right| < \frac{1}{poly(n)}$$

Since $\hat{C}$ is given no information about w, the probability that $\hat{C}$ generates a commitment to $(\hat{Q}',w)$ is at most ϵ (by the fact that w is uniformly distributed in $\mathcal{D}$). Therefore, the probability that C generates a commitment to (Q',w) where $Q' \neq Q$ is less than $\epsilon + \frac{1}{poly(n)}$ as required. ■

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

Claim 9.6 (Failure w.r.t R): *For every ppt channel C, every polynomial $p(\cdot)$ and all sufficiently large n's*

$$Pr[B = \text{acc} \;\wedge\; \tau_A \neq \tau_B \;\wedge\; ((Q,w),(Q',w')) \notin R] < \frac{1}{p(n)}$$

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 is $Q'(w)$. Thus, with overwhelming probability B rejects unless $Q'(w') = Q'(w)$. As we will show, this implies that $\tau_A = \tau_B$, in contradiction to the claim hypothesis. In the following fact, we formally show that with overwhelming probability, B's output from the polynomial evaluation equals $Q'(w)$.

Fact 9.7 *For every ppt channel C,*

$$Pr[B = \text{acc} \;\wedge\; \tau_B \neq Q'(w)] < \frac{1}{poly(n)}$$

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 his 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.

[29] 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.

Now, let Q_C be the ordinary commitment sent by C to B before the polynomial evaluation. Then, by the definition of the augmented polynomial evaluation, B's output τ_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 $\tau_B \in \{Q_C(w), \perp\}$.

It remains to 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, 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 $\tau_B \in \{Q_C(w), \perp\}$. However, since B checks if $y = f^{2n}(\tau_B)$ and this never holds when $\tau_B = \perp$, B always rejects if $\tau_B = \perp$. Thus,

$$Pr[B = \text{acc} \ \wedge \ \tau_B \neq Q_C(w)] < \frac{1}{poly(n)}$$

The proof is completed by noticing that the zero-knowledge proof states (among other things) that $Q_C = Q'$. Thus by the soundness of the zero-knowledge proof (which also holds in our setting), the probability that B accepts and $Q_C \neq Q'$ is negligible. We conclude that

$$Pr[B = \text{acc} \ \wedge \ \tau_B \neq Q'(w)] < \frac{1}{poly(n)}$$

■

On the other hand, we now show that with overwhelming probability, $\tau_B = Q'(w')$.

Fact 9.8 *For every ppt channel C,*

$$Pr[B = \text{acc} \ \wedge \ \tau_B \neq Q'(w')] < \frac{1}{poly(n)}$$

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, B rejects unless $y = f^{2n}(\tau_B)$. Since f^{2n} is a 1–1 function, we conclude that with overwhelming probability, B rejects unless $\tau_B = Q'(w')$. ■

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 9.7 (or equivalently by Fact 9.8), we have that with overwhelming probability, B rejects unless $\tau_A = Q(w) = Q'(w') = \tau_B$, in contradiction to the hypothesis that $\tau_A \neq \tau_B$.
- *Case $w' \neq w$:* From Facts 9.7 and 9.8 we have that if B accepts then with overwhelming probability $Q'(w') = Q'(w)$. 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 9.6. ■

Lemma 4.8 is obtained by combining Claims 9.5 and 9.6. ■

10 Full Proof of Password Secrecy

In this section we prove the password secrecy requirement which states that at the conclusion of the protocol execution, the password w is $(1-O(\epsilon))$-indistinguishable from a random $\tilde{w} \in_R \mathcal{D}$, with respect to the channel's view.

Theorem 10.1 (Theorem 4.10 – restated): *For every ppt channel C, every polynomial $p(\cdot)$ and all sufficiently large n's*

$$\left| Pr_w[\mathrm{Expt}_w^{A(w),B(w)}(C)=1] - Pr_{w,\tilde{w}}[\mathrm{Expt}_{\tilde{w}}^{A(w),B(w)}(C)=1] \right| < 26\epsilon + \frac{1}{p(n)}$$

Proof: As described in the proof sketch, the theorem is proved by first "removing" the entire (C,B) execution. Loosely speaking, we show that the (C,B) execution can be simulated by C himself (while interacting only with A), such that his view in the simulated setting is $(1-O(\epsilon))$-indistinguishable from his view in a full execution with both A and B. The proof then continues by showing that for every channel C interacting with A alone, the password w is $(1-O(\epsilon))$-indistinguishable from $\tilde{w} \in_R \mathcal{D}$ (with respect to C's view). Putting these together, we have that when C interacts with both A and B, he can distinguish w from $\tilde{w} \in_R \mathcal{D}$ with probability at most $O(\epsilon)$.

The prove is divided into two lemmas: in the first we remove the (C,B) execution and in the second we upper bound the "amount of information" C can learn about w in a non-concurrent execution with A only. We denote an analogous experiment in which C' interacts only with A by $\mathrm{Expt}_z^{A(w)}(C')$ (this experiment is formally defined in Section 6.1).

Lemma 10.2 (Removing 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 for every randomized process $z=Z(w)$, every polynomial $p(\cdot)$ and all sufficiently large n's*

$$\left| Pr_w[\mathrm{Expt}_z^{A(w)}(C')=1] - Pr_w[\mathrm{Expt}_z^{A(w),B(w)}(C)=1] \right| < 7\epsilon + \frac{1}{p(n)}$$

Proof: The proof of the lemma is in two steps. In the first step we remove the (C,B) validation stage. Following this, we remove the remaining (first two stages) of the (C,B) execution. Let B_2 be a party who participates in only the first two stages of the protocol (i.e., he halts before the validation stage). Then, the fact that we can remove the (C,B) validation stage is stated as follows.

Claim 10.3 (Removing the (C,B) Validation): *Let C be a ppt channel and let B_2 be a party who does not participate in the validation stage. Then, there exists a ppt channel $\tilde{C}$ interacting with A and B_2 such that for every randomized process $z=Z(w)$*

$$\left| Pr_w[\mathrm{Expt}_z^{A(w),B(w)}(C)=1] - Pr_w[\mathrm{Expt}_z^{A(w),B_2(w)}(\tilde{C})=1] \right| < 7\epsilon + \frac{1}{poly(n)}$$

Proof: This proof involves showing how B's role in the (C,B) validation stage can be simulated by $\tilde{C}$ (for C). Basically, this simulation is made possible due to the MAC sent in the last step of the protocol. Recall that the (C,B) validation stage involves four steps: (1) B receives a y-string (which should equal $f^{2n}(\tau_B)$); (2) B verifies a zero-knowledge proof; (3) B receives a string that should equal a MAC of his entire session-transcript; and (4) B outputs an accept/reject bit.

First note that the only messages *sent* by B during the validation stage are his messages as a verifer for the zero-knowledge proof and his accept/reject bit. Therefore, only these messages need to be simulated. However, as B is an honest verifier, his messages in the zero-knowledge proof can be perfectly simulated by $\tilde{C}$, who emulates the verifier in C's proof to B. It thus remains for $\tilde{C}$ to simulate B's accept/reject bit. We show that the MAC sent in the validation stage is such that if C was not reliable, then B rejects with probability $1 - O(\epsilon)$. This enables $\tilde{C}$ to "predict" B's output-bit based on whether or not C was reliable.

Formally, $\tilde{C}$ runs the protocol (with A and B_2) by passing all messages via C and by playing the verifier in the zero-knowledge proof of the (C, B) validation stage. 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, easily verifiable by $\tilde{C}$ (as he views all the communication). If C was reliable then $\tilde{C}$ outputs accept for B, otherwise he outputs reject for B. This completes the simulation of C's interaction with A and B. Let $\chi_{\tilde{C}}$ denote the simulated accept/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 in the claim equals the probability that $\tilde{C}$'s prediction is wrong (i.e., the probability that $B = \mathsf{acc}$ and $\chi_{\tilde{C}} = \mathsf{rej}$ or visa versa). Noticing that $\chi_{\tilde{C}} = \mathsf{acc}$ if and only if C is reliable, we have that:

$$\begin{aligned} &\left| Pr_w[\text{Expt}_z^{A(w),B(w)}(C) = 1] - Pr_w[\text{Expt}_z^{A(w),B_2(w)}(\tilde{C}) = 1] \right| \\ &\quad = Pr[B = \text{acc} \ \wedge \ C \text{ not reliable}] + Pr[B = \text{rej} \ \wedge \ C \text{ reliable}] \end{aligned}$$

First notice that when C is reliable B *always* accepts. That is, $Pr[B = \text{rej} \ \wedge \ C \text{ reliable}] = 0$. We now show that $Pr[B = \text{acc} \ \wedge \ C \text{ not reliable}]$ is at most negligibly more than 7ϵ and this completes the proof of Claim 10.3.

Claim 10.4 *For every ppt channel C,*

$$Pr[B = \text{acc} \ \wedge \ C \text{ not reliable}] < 7\epsilon + \frac{1}{poly(n)}$$

Proof: The proof of this claim 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 requirement ensures that this happens (undetectably by B) with probability at most $O(\epsilon)$.

The security of the MAC, shown in Corollary 4.9 and proven at the end of Section 8, states the following. Let t_A be A's message transcript. Then for every $t \neq t_A$, the string $MAC_{k_1(\tau_A)}(t)$ is $(1 - 4\epsilon)$-pseudorandom with respect to C's view. By the definition of reliability, if C is not reliable then B's message transcript (denoted t_B) is not equal to t_A. That is, if C is not reliable we have that $MAC_{k_1(\tau_A)}(t_B)$ is $(1 - 4\epsilon)$-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 $MAC_{k_1(\tau_B)}(t_B)$, where $k_1(\tau_B)$ is the MAC-key. Notice that the key used by B for the MAC is $k_1(\tau_B)$, whereas Corollary 4.9 refers to a MAC keyed by $k_1(\tau_A)$. However, if $\tau_A = \tau_B$ then $k_1(\tau_A) = k_1(\tau_B)$. Therefore, if $\tau_A = \tau_B$, then the Corollary holds and the probability that C generates the correct MAC-value is at most negligibly greater than 4ϵ. That is,

$$Pr[B = \text{acc} \ \wedge \ C \text{ not reliable} \ \wedge \ \tau_A = \tau_B] < 4\epsilon + \frac{1}{poly(n)}$$

On the other hand, if $\tau_A \neq \tau_B$ then irrespective of the MAC, the probability that B accepts is at most negligibly more than 3ϵ. This is due to the key-match requirement proven in Theorem 4.6. We conclude that

$$\begin{aligned} & Pr[B = \text{acc} \ \wedge \ C \text{ not reliable}] \\ & \quad = \ Pr[B = \text{acc} \ \wedge \ C \text{ not reliable} \ \wedge \ \tau_A \neq \tau_B] + Pr[B = \text{acc} \ \wedge \ C \text{ not reliable} \ \wedge \ \tau_A = \tau_B] \\ & \quad < \ 3\epsilon + 4\epsilon + \frac{1}{poly(n)} \end{aligned}$$

■

As stated above, this completes the proof of Claim 10.3. ■

It remains now to remove the rest of the execution between C and B_2. That is,

Claim 10.5 (Removing the Remaining (C, B_2) Execution): *For every ppt channel $\tilde{C}$ interacting with A and B_2, there exists a ppt channel C' interacting only with A such that for every randomized process $z = Z(w)$*

$$\left| Pr_w[\text{Expt}_z^{A(w),B_2(w)}(\tilde{C}) = 1] - Pr_w[\text{Expt}_z^{A(w)}(C') = 1] \right| < \frac{1}{poly(n)}$$

Proof: Intuitively, B_2's role can be simulated without any knowledge of w. Loosely speaking, this is because B_2 only uses w in the $(\tilde{C}, B_2)$ polynomial evaluation, and in this evaluation $\tilde{C}$ receives no output. Formally, this is shown by proving that if B_2 were to input an independently chosen $\tilde{w} \in_R \mathcal{D}$ (into the polynomial evaluation), instead of w, then $\tilde{C}$ would not be able to tell the difference. That is, for every randomized process $z = Z(w)$

$$\left| Pr_w[\text{Expt}_z^{A(w),B_2(w)}(\tilde{C}) = 1] - Pr_w[\text{Expt}_z^{A(w),B_2(\tilde{w})}(\tilde{C}) = 1] \right| < \frac{1}{poly(n)} \tag{27}$$

(Observe that in the second experiment, B_2's input is $\tilde{w}$.) We prove Equation (27) even when $\tilde{C}$ is given w as auxiliary input. Now, since w constitutes all of A's input, the channel $\tilde{C}(w)$ can perfectly simulate the entire (A, C) execution. It thus remains to show that for $z = Z(w)$,

$$\left| Pr_w[\text{Expt}_z^{B_2(w)}(\tilde{C}(w)) = 1] - Pr_w[\text{Expt}_z^{B_2(\tilde{w})}(\tilde{C}(w)) = 1] \right| < \frac{1}{poly(n)}$$

However, this is derived directly from the security of the polynomial evaluation. This is because $\tilde{C}$ obtains no output from the polynomial evaluation, and this is the only part of the protocol where B_2 uses his input (of w or $\tilde{w}$). Thus, $\tilde{C}$ can distinguish the cases that B_2 has input w or $\tilde{w}$ with at most negligible probability, and Equation (27) follows.

The proof of the current claim (i.e., Claim 10.5) is completed by noting that when B_2 inputs $\tilde{w} \in_R \mathcal{D}$ (chosen independently of w), then the entire $(\tilde{C}, B_2)$ execution can be perfectly simulated by a channel C'. That is, there exists a channel C' such that for every randomized process $z = Z(w)$

$$Pr_w[\mathrm{Expt}_z^{A(w)}(C') = 1] = Pr_w[\mathrm{Expt}_z^{A(w),B_2(\tilde{w})}(\tilde{C}) = 1] \tag{28}$$

The claim follows from Equations (27) and (28). ■

Combining Claims 10.3 and 10.5, we complete the proof of Lemma 10.2. ■

We now remain with a (non-concurrent) execution between A and C'. In this setting, we show that C' can distinguish w from $\tilde{w} \in_R \mathcal{D}$ with probability at most 12ϵ. That is:

Lemma 10.6 *For every ppt channel C' interacting only with A, every polynomial $p(\cdot)$ and all sufficiently large n's*

$$\left| Pr_w[\mathrm{Expt}_w^{A(w)}(C') = 1] - Pr_{w,\tilde{w}}[\mathrm{Expt}_{\tilde{w}}^{A(w)}(C') = 1] \right| < 12\epsilon + \frac{1}{p(n)}$$

Proof: As in the previous lemma, we first show that the (A, C)-validation stage can be "removed".

Claim 10.7 (Removing the (A, C) Validation): *Let A_2 be a party that does not participate in the validation stage. Then for every ppt channel C' interacting with A, there exists a ppt channel C'' interacting with A_2 such that for every randomized process $z = Z(w)$*

$$\left| Pr_w[\mathrm{Expt}_z^{A(w)}(C') = 1] - Pr_w[\mathrm{Expt}_z^{A_2(w)}(C'') = 1] \right| < 6\epsilon + \frac{1}{poly(n)}$$

Proof: The (A, C) validation stage consists of A sending $y = f^{2n}(Q(w))$, proving a statement in zero-knowledge and sending a MAC of her entire session-transcript. In this proof we show how all parts of this stage can be simulated by C'' (for C').

- *The y-value sent by A:* By Theorem 4.2, at the completion of Stage 2 by A, the string $Q(w)$ is $(1-2\epsilon)$-pseudorandom. Since f^{2n} is 1–1 (and polynomial-time computable), the string $f^{2n}(Q(w))$ is also $(1-2\epsilon)$-pseudorandom. Thus, C'' can simulate this step by choosing a uniformly distributed string instead of $f^{2n}(Q(w))$, and C' can distinguish the simulation from a real interaction with probability at most negligibly greater than 2ϵ.
- *The zero-knowledge proof:* Here we remove a zero-knowledge proof in a standard stand-alone setting.[30] Therefore, by the definition of zero-knowledge, there exists a simulator that generates transcripts indistinguishable from real proofs. Thus, C'' simply runs this simulator (with C' as the verifier) and produces a "fake transcript" that is computationally indistinguishable from a real proof (with respect to C''s view).
- *The MAC:* As with the y-value, C'' simulates the MAC by sending a random string instead. We claim that the MAC value sent by A is $(1 - 4\epsilon)$-pseudorandom. Therefore C' can distinguish a random string from a correct MAC value with probability at most negligibly greater than 4ϵ. The proof of the fact that A's MAC is $(1 - 4\epsilon)$-pseudorandom is derived directly from the proof of Corollary 8.4. This is based on the fact that $k_1(Q(w))$ is a $(1 - 4\epsilon)$-pseudorandom string and thus $MAC_{k_1(Q(w))}(\cdot)$ is a $(1 - 4\epsilon)$-pseudorandom function.

[30] We stress that there is no concurrent execution here, and so replacing a zero-knowledge interactive proof by a simulated transcript is straightforward.

Putting the above together we have that C' can distinguish C''''s simulation from real messages sent by A with probability at most negligibly greater than 6ϵ. This completes the proof of Claim 10.7. ■

What remains now is a scenario where a channel C'' interacts with A_2. The protocol thus consists only of A_2 committing to (Q, w) followed by a pre-key exchange stage in which A_2 inputs Q (recall that w is not used by A_2 in this stage). Then, by the hiding property of the commitment, it is immediate that C'' can distinguish w from $\tilde{w} \in_R \mathcal{D}$ with at most negligible probability. That is,

$$\left|Pr_w[\mathrm{Expt}_w^{A_2(w)}(C'') = 1] - Pr_{w,\tilde{w}}[\mathrm{Expt}_{\tilde{w}}^{A_2(w)}(C'') = 1]\right| < \frac{1}{poly(n)} \tag{29}$$

Combining Equation (29) with Claim 10.7 (taking $z = w$ once and $z = \tilde{w} \in_R \mathcal{D}$ a second time), we conclude that

$$\left|Pr_w[\mathrm{Expt}_w^{A(w)}(C') = 1] - Pr_{w,\tilde{w}}[\mathrm{Expt}_{\tilde{w}}^{A(w)}(C') = 1]\right| < 2 \cdot 6\epsilon + \frac{1}{poly(n)} = 12\epsilon + \frac{1}{poly(n)}$$

■

Combining Lemma 10.6 with Lemma 10.2 (applied twice, once for $z = w$ and once for $z = \tilde{w}$) we have

$$\begin{aligned}
&\left|Pr_w[\mathrm{Expt}_w^{A(w),B(w)}(C) = 1] - Pr_{w,\tilde{w}}[\mathrm{Expt}_{\tilde{w}}^{A(w),B(w)}(C) = 1]\right| \\
<\ & \left|Pr_w[\mathrm{Expt}_w^{A(w)}(C') = 1] - Pr_w[\mathrm{Expt}_{\tilde{w}}^{A(w)}(C') = 1]\right| + 2 \cdot 7\epsilon + \frac{1}{poly(n)} \\
<\ & 12\epsilon + 14\epsilon + \frac{1}{poly(n)} = 26\epsilon + \frac{1}{poly(n)}
\end{aligned}$$

■

11 Proof of Multi-Session Security

The claims in this section are derived from our definition for session-key generation protocols only and are correct for *any* protocol fulfilling this definition.

We focus on the case where the adversary sequentially invokes m sessions of our protocol with the same pair of parties, A and B. In each of these invocations, A and B use the same password $w \in \mathcal{D}$. Recall that neither A nor B will agree to participate in a new session before it has locally terminated the previous sessions. (We ignore other pairs of parties that share independently selected passwords; these are easily simulated by the adversary.)

We refrain from presenting formal definitions of security for m sessions (these are easy extensions of the single session case), and confine ourselves to showing how to reduce the security of m sessions to the security of a single session. Throughout the discussion, the dictionary $\mathcal{D}$ is fixed and implicit in all notations.

11.1 Password secrecy after m sessions

We start by considering the case of $m = 2$. In order to allow a generalization to arbitrary m, we consider the execution of a protocol that (as stand-alone) has "password security" $1 - \epsilon_1$, followed by an execution of a protocol that (as stand-alone) has "password security" $1 - \epsilon_2$. For starters, one may think of $\epsilon_1 = \epsilon_2 = O(\epsilon)$ (i.e., each protocol is a single-session protocol). During the induction, the protocols become the sequential composition of a number of single-session protocols together and ϵ_1, ϵ_2 are adjusted appropriately. By *password insecurity* (i.e., the ϵ_i's above) we mean an upper bound on the distinguishability-gap, from the channel's point of view, between the password used in the execution and a uniformly chosen password in $\mathcal{D}$.

Let $X_C(w_1, w_2, w_3)$ denote the probability that the channel C outputs 1 in the following experiment: First, the channel invokes the first protocol between A and B, when they both use the password $w_1 \in \mathcal{D}$ and the adversary modifies their interaction (and effectively interacts concurrently with each of them). Next, the channel invokes the second protocol between A and B, when they both use the password $w_2 \in \mathcal{D}$ (and again it effectively interacts concurrently with each of them). Finally, the channel is presented with a challenge $w_3 \in \mathcal{D}$.

Lemma 11.1 *Let $X_C(w_1, w_2, w_3)$, ϵ_1 and ϵ_2 be as above. Then, for any probabilistic polynomial-time channel C, it holds that $|\mathrm{E}[X_C(W, W, W)] - \mathrm{E}[X_C(W, W, W')]| \leq 2\epsilon_1 + \epsilon_2$, where W and W' are independent and uniformly distributed in $\mathcal{D}$.*

Proof: We consider several hybrid executions, and relate some pairs so to derive the above claim. Below, W_1, W_2 and W_3 represent random variables that are independent and uniformly distributed in $\mathcal{D}$.

Claim 1: $|\mathrm{E}[X_C(W_1, W_2, W_3)] - \mathrm{E}[X_C(W_1, W_1, W_3)]| \leq \epsilon_1$.

Proof: In order to prove this claim, we use the password security of the first protocol. Intuitively, a channel succeeding in the above experiment can be used to distinguish the password used in the first protocol. First, recall that the requirement of password secrecy is that of indistinguishability after the protocol execution. That is, the channel is given a challenge c which either equals w_1 (the password used in the protocol) or $w_2 \in_R \mathcal{D}$.

Now, let C' be a channel for the first protocol, that behaves as follows. First C' emulates the interaction of C with $A(w_1)$ and $B(w_1)$ by actually interacting with $A(w_1)$ and $B(w_1)$ (i.e., C' simply forwards all messages between A,C and B for this interaction). Next, C' obtains a challenge c (which is either w_1 or $w_2 \in_R \mathcal{D}$) and uses it to emulate the execution $C^{A(c),B(c)}$. (We stress that C' does not interact with parties A and B, which have already terminated, but rather runs their programs internally using c as their password.) Finally, C' passes C a uniformly selected challenge $w_3 \in_R \mathcal{D}$, and outputs whatever C does.

Clearly, in case $c = w_1 \in_R \mathcal{D}$, C's view is exactly that of the event $X_C(W_1, W_1, W_3)$ (W_3 is independent of the first two passwords and therefore C''s emulation is perfect). Therefore, the expected value of the output of C' equals $\mathrm{E}[X_C(W_1, W_1, W_3)]$. Similarly, in case $c \in_R \mathcal{D}$ and is independent of w_1, the expected value of the output of C' equals $\mathrm{E}[X_C(W_1, W_2, W_3)]$. Therefore, C' distinguishes w_1 from w_2 with exactly the probability gap $|\mathrm{E}[X_C(W_1, W_1, W_3)] - \mathrm{E}[X_C(W_1, W_2, W_3)]|$. The claim follows. □

Claim 2: $|\mathrm{E}[X_C(W_1, W_1, W_1)] - \mathrm{E}[X_C(W_1, W_2, W_2)]| \leq \epsilon_1$.

Proof: The proof is similar to the proof of Claim 1. Channel C' emulates the two interactions in exactly the same way. Then, C' passes his challenge c (that either equals w_1 or w_2) to C for the challenge, instead of selecting a uniform one. In case $c = w_1 \in_R \mathcal{D}$, the expected value of the output of C' equals $\mathrm{E}[X_C(W_1, W_1, W_1)]$. On the other hand, in case c and w_1 are independently distributed in $\mathcal{D}$, the expected value of the output of C' equals $\mathrm{E}[X_C(W_1, W_2, W_2)]$. As before, this implies the claim. □

Claim 3: $|\mathrm{E}[X_C(W_1, W_2, W_2)] - \mathrm{E}[X_C(W_1, W_2, W_3)]| \leq \epsilon_2$.

Proof: Here we use the password security of the second protocol. Intuitively, the first protocol here is run with a password independent of the second protocol and the challenge. It can therefore be emulated and what remains is the standard password secrecy setting of the second protocol.

Consider a channel C' for the second protocol, that behaves as follows. First C' uniformly selects $w_1 \in_R \mathcal{D}$ and uses it to emulate the execution $C^{A(w_1),B(w_1)}$. (We stress that C' does not interact with parties A and B, but rather runs their programs internally using w_1 as their password.) Next, C' emulates the interaction of C with $A(w_2)$ and $B(w_2)$ by actually interacting with $A(w_2)$ and $B(w_2)$. Finally, C' obtains a challenge c and passes it to C, and outputs whatever C does.

Clearly, in case $c = w_2 \in_R \mathcal{D}$, the expected value of the output of C' equals $\mathrm{E}[X_C(W_1, W_2, W_2)]$. On the other hand, in case c and w_2 are independently distributed in $\mathcal{D}$, the expected value of the output of C' equals $\mathrm{E}[X_C(W_1, W_2, W_3)]$. $\square$

Combining the three claims, we have

$$\begin{aligned}
|\mathrm{E}[X_C(W,W,W)] - \mathrm{E}[X_C(W,W,W')]| &= |\mathrm{E}[X_C(W_1,W_1,W_1)] - \mathrm{E}[X_C(W_1,W_1,W_3)]| \\
&\leq |\mathrm{E}[X_C(W_1,W_1,W_1)] - \mathrm{E}[X_C(W_1,W_2,W_2)]| \\
&\quad + |\mathrm{E}[X_C(W_1,W_2,W_2)] - \mathrm{E}[X_C(W_1,W_2,W_3)]| \\
&\quad + |\mathrm{E}[X_C(W_1,W_2,W_3)] - \mathrm{E}[X_C(W_1,W_1,W_3)]| \\
&\leq \epsilon_1 + \epsilon_2 + \epsilon_1
\end{aligned}$$

where in the last inequality Claims 2, 3 and 1 respectively are applied. ■

Using the proof of Lemma 11.1 (i.e., paying close attention to one aspect of it), and using the password secrecy requirement of the protocol, we obtain the following.

Theorem 11.2 *From the point of view of any probabilistic polynomial-time channel that handles m sessions of our protocol, the password is $(1-O(m\epsilon))$-indistinguishable from the uniform distribution over $\mathcal{D}$.*

Proof: The theorem is proved by induction on the number of sessions, m. We consider two protocols: the first protocol consists of the first (among the m executions) execution of our basic protocol, and the second consists of the remaining $m-1$ executions of our basic protocol.

We wish to prove the theorem for a number of session that may grow as a function of the security parameter, and not merely for a constant number of executions. Still, let us first consider how the proof would go for a constant m. If we denote by $I(i)$ the password insecurity of i sessions, then by Lemma 11.1 we have $I(i) \leq 2 \cdot I(1) + I(i-1)$, and $I(m) \leq 2m \cdot I(1) = O(m\epsilon)$ follows (as desired).

However, this notation hides the actual running-time of the adversarial channels. Let us then denote by $I_T(i)$ the password insecurity of i sessions with respect to adversaries running in time T. Then, Lemma 11.1 says that if for every polynomial function T it holds that $I_T(1) \leq \epsilon_1$ and $I_T(i-1) \leq \epsilon_2$ then for every polynomial T' it holds that $I_{T'}(i) \leq 2\epsilon_1 + \epsilon_2$. Recall that, *in general*, it is not possible to applying induction on such a claim *for a non-constant of times.* This is because the running time of the adversary may become non-polynomial.

However, looking at the proof of Lemma 11.1, we observe that it actually establishes that for every function T' if $I_{T'}(i) > 2\epsilon_1 + \epsilon_2$ then either $I_{T_1}(1) > \epsilon_1$ or $I_{T_{i-1}}(i-1) > \epsilon_2$, where $T_1(n) = T'(n) + (i-1) \cdot p(n)$ and $T_{i-1}(n) = T'(n) + p(n)$, and where p is a fixed polynomial denoting the time it takes to emulate the actions of A and B (in a single session of our protocol). The running-time $T_1(n)$ is obtained by noticing that the adversary for Claims 1 and 2 in the proof of the lemma works by invoking the "original" adversary (taking time $T'(n)$) and emulating A and B for $i-1$ invocations (taking time $(i-1)p(n)$). On the other hand, $T_{i-1}(n)$ is obtained from Claim 3 where only the first protocol invocation need be emulated at a cost of $p(n)$ more than $T'(n)$.[31]

Now, for every T, we have $I_T(i) \leq 2 \cdot I_{T+i \cdot p}(1) + I_{T+p}(i-1)$, and $I_T(m) \leq 2m \cdot I_{T+m \cdot p}(1)$ follows. Thus, the contradicting adversary remains polynomial time even for any polynomial number of invocations. Using the fact that $I_{T'}(1) = O(\epsilon)$ for any polynomial T' (as well as the fact that both t and m are polynomials), the theorem follows. ■

11.2 Session-key secrecy after m sessions

Combining Theorem 11.2 and the session-key secrecy for a single session, we prove that the m^{th} session-key is $1-O(m\epsilon)$ pseudorandom from the point of view of a channel that conducts m sessions.

Again, we consider the sequential execution of two protocols, the first having (as stand alone) "password security" $1-\epsilon_1$, and the second having (as stand alone) "session-key security" $1-\epsilon_2$. We stress that the second protocol must be a single session of our protocol (since we refer to the way it generates the session-key). We redefine X_C so that it refers to an experiment in which a candidate session-key is presented as a challenge (and so that it refers explicitly to the first polynomial selected by A in the second protocol).[32] That is, we let $X_C(w_1, (q, w_2), c)$ denote the probability that the output of channel C equals 1 in the following experiment: As before, first C interacts concurrently with A_1 and B_1, where each party uses password w_1, next C interacts with A_2 and B_2 where each party uses password w_2 *but A_2 uses the polynomial q instead of Q* (so to obtain the session-key $k_2(q(w_2))$; by our protocol definition $k_2(q(w_2))$ is the output session-key), and finally C is presented a (session-key) challenge $c \in \{0,1\}^n$.

Lemma 11.3 *Let $X_C(w_1, (q, w_2), c)$, ϵ_1 and ϵ_2 be as above. Then, for any probabilistic polynomial-time channel C, it holds that $|\mathrm{E}[X_C(W, (W, Q), k_2(Q(W)))] - \mathrm{E}[X_C(W, (W, Q), U_n)]| \leq 2\epsilon_1 + \epsilon_2$, where Q is a uniformly distributed linear polynomial.*

Proof: We first prove a claim that will allow us to disregard the first protocol (at a cost of $2 \cdot \epsilon_1$). We do this by showing that if an *independent* password is used for the first protocol instead, then this can make at most a difference of ϵ_1.

Claim 1: Let $R : \{0,1\}^* \to \{0,1\}^*$ be a probabilistic polynomial-time algorithm satisfying $|R(x)| = |x|/3$ for all x's. Then $|\mathrm{E}[X_C(W_1, (W_1, Q), R(W_1, Q))] - \mathrm{E}[X_C(W_1, (W_2, Q), R(W_2, Q))]| \leq \epsilon_1$.

Proof: The claim follows from the password security of the first protocol, similarly to the proof of Claim 2 in Lemma 11.1: We consider a channel C' that first interacts with $A_1(w)$ and $B_1(w)$. Next, upon receiving a *password challenge* c, the adversary C' uniformly selects a linear polynomial q and emulates the actions of $A_2(c, q)$ and $B_2(c)$. Finally, C' presents C with the challenge $R(c, q)$, and outputs whatever C does.

In case $c = w \in_R \mathcal{D}$, the expected value of the output of C' equals $\mathrm{E}[X_C(W_1, (W_1, Q), R(W_1, Q))]$. On the other hand, in case c and w are independently distributed in $\mathcal{D}$, the expected value of the output of C' equals $\mathrm{E}[X_C(W_1, (W_2, Q), R(W_2, Q))]$. □

[31] The question of which case (i.e., the first session or the remaining $i-1$ sessions) to use is ignored here and below. A trivial solution is to specify the choice via a (non-uniform) auxiliary input. If fact, unraveling the induction (below), one may find an adequate choice (i.e., auxiliary input) by a preprocessing in which each of the m (not 2^m) possibilities is evaluated. (The induction tree has m leaves, and each determines the path to it.)

[32] As we explained above, these claims are based solely on our definition and are not related to any specific protocol. However, for ease of presentation here, we consider the key generated from our protocol. This is not a necessity, and the result holds for any protocol fulfilling the requirements of Definition 2.4.

Applying the above claim to R that uniformly selects a string of length $|w| = |q|/2$, we have

$$|\mathrm{E}[X_C(W_1, (W_1, Q), U_n)] - \mathrm{E}[X_C(W_1, (W_2, Q), U_n)]| \leq \epsilon_1 \tag{30}$$

Applying the claim to $R(w, q) = k_2(q(w))$, we have

$$|\mathrm{E}[X_C(W_1, (W_1, Q), k_2(Q(W_1)))] - \mathrm{E}[X_C(W_1, (W_2, Q), k_2(Q(W_2)))]| \leq \epsilon_1 \tag{31}$$

We next prove that in the case that the first protocol is run with an independent password, C distinguishes the cases with probability at most ϵ_2.

Claim 2: $|\mathrm{E}[X_C(W_1, (W_2, Q), U_n)] - \mathrm{E}[X_C(W_1, (W_2, Q), k_2(Q(W_2)))]| \leq \epsilon_2$.

Proof: The claim follows by the session-key security of the second protocol. Since the password used in the first protocol is independent of that used in the second protocol, the first protocol can be perfectly emulated for C (as in the proof of Claim 3 in Lemma 11.1). □

Combining Equations (30) and (31) with Claim 2, the lemma follows. ■

Considering the first $m - 1$ sessions as one protocol and the last session as a second, and applying Lemma 11.3 (using Theorem 11.2 and the session-key secrecy requirement of the protocol), we immediately obtain:

Theorem 11.4 *From the point of view of any probabilistic polynomial-time channel that handles m sessions of our protocol, the last session-key is $(1 - O(m\epsilon))$-indistinguishable from the uniform distribution over $\{0,1\}^n$.*

11.3 Undetected session-key mismatch after m sessions

By *undetected session-key mismatch* we refer to the event in which the legitimate communicators end-up with different session-keys without detecting this fact (this is the key-match requirement in the definition). Combining Theorem 11.2 and the bound on undetected session-key mismatch in a (stand-alone) single session, we prove that the probability that an undetected session-key mismatch occurs in the last session, after $m - 1$ prior sessions, is at most $O(m\epsilon)$.

Again, we consider the sequential execution of two protocols, the first having (as stand alone) "password security" $1 - \epsilon_1$, and the second having (as stand alone) "undetected mismatch security" $1 - \epsilon_2$. We redefine X_C so that $X_C(w_1, w_2)$ denotes the probability of the event 'undetected mismatch' *for the second session* occuring in the following experiment: As before, first C interacts concurrently with A_1 and B_1, where each party uses password w_1, next it interacts with A_2 and B_2 where each party uses password w_2.

Lemma 11.5 *Let $X_C(w_1, w_2)$, ϵ_1 and ϵ_2 be as above. Then, for any probabilistic polynomial-time channel C, it holds that $\mathrm{E}[X_C(W, W)] \leq \epsilon_1 + \epsilon_2$.*

Proof: We first claim that $|\mathrm{E}[X_C(W_1, W_1)] - \mathrm{E}[X_C(W_1, W_2)]| \leq \epsilon_1$. The claim follows from the password security of the first protocol, similarly to the proof of Claim 1 in Lemma 11.3: We consider a channel C' that first interacts with $A_1(w)$ and $B_1(w)$. Next, upon receiving a password challenge c, the adversary C' emulates the actions of $A_2(c)$ and $B_2(c)$. Finally, C' outputs 1 if and only if the 'undetected mismatch' event has occured in this emulation. (Surely, C' can determine this bit since it emulates all parties in the second protocol.) Therefore, if C's success in causing a mismatch in the second protocol can be used by C' to distinguish the password in the first protocol.

We next show that $\mathrm{E}[X_C(W_1, W_2)] \leq \epsilon_2$. This follows by the 'undetected mismatch' security of the second protocol, by emulating the first protocol (as in the proof of Claim 3 in Lemma 11.1). Combining the two claims, the lemma follows. ■

As in the previous subsection, we immediately obtain

Theorem 11.6 *For any probabilistic polynomial-time channel that handles m sessions of our protocol, the probability that in the last session the parties output different session-keys without detecting this fact is at most $O(m\epsilon)$.*

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 discussion. Finally, we would like to thank Ran Canetti, Shai Halevi and Tal Rabin for discussion that led to a significant simplification of the protocol.

References

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

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

[3] M. Bellare and P. Rogaway. Provably Secure Session Key Distribution: The Three Party Case. In *27th STOC*, pages 57–66, 1995.

[4] M. Bellare, R. Canetti and H. Krawczyk. Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols. In *30th STOC*, pages 419–428, 1998.

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

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

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

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

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

[10] 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.

[11] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SICOMP*, Vol. 13, pages 850–864, 1984. Preliminary version in *23rd FOCS*, 1982.

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

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

[14] R. Canetti. Security and Composition of Multi-party Cryptographic Protocols. *Journal of Cryptology*, Vol. 13, No. 1, pages 143–202, 2000.

[15] R. Canetti, Private Communication.

[16] Ran Canetti, Oded Goldreich, and Shai Halevi. The Random Oracle Methodology, Revisited. In *Proc. of the 30th STOC*, pages 209–218, 1998.

[17] G. Di Crescenzo, Y. Ishai and R. Ostrovsky. Non-Interactive and Non-Malleable Commitment. In *Proc. of the 30th STOC*, pages 141-150, 1998.

[18] W. Diffie, and M.E. Hellman. New Directions in Cryptography. *IEEE Trans. on Info. Theory*, IT-22 (Nov. 1976), pages 644–654.

[19] D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. *SIAM Journal on Computing*, January 2000. A preliminary version appeared in *23rd STOC*, pages 542–552, 1991.

[20] U. Feige and A. Shamir. Witness Indistinguishability and Witness Hiding Protocols. In *22nd STOC*, pages 416–426, 1990.

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

[22] O. Goldreich. *Foundation of Cryptography – Fragments of a Book.* February 1995. Available from `http://www.wisdom.weizmann.ac.il/~oded/foc-book.html`.

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

[24] O. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions. *JACM*, Vol. 33, No. 4, pages 792–807, 1986.

[25] O. Goldreich and A. Kahan. How To Construct Constant-Round Zero-Knowledge Proof Systems for NP. Journal of Cryptology, Vol. 9, pages 167–189, 1996.

[26] O. Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing but their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *JACM*, Vol. 38, No. 1, pages 691–729, 1991.

[27] O. Goldreich, S. Micali and A. Wigderson. How to Play any Mental Game – A Completeness Theorem for Protocols with Honest Majority. In *19th STOC,* pages 218–229, 1987. For details see [23].

[28] S. Goldwasser and S. Micali. Probabilistic Encryption. *JCSS*, Vol. 28, No. 2, pages 270–299, 1984.

[29] S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SICOMP*, Vol. 18, pages 186–208, 1989. Preliminary version in *17th STOC*, 1985.

[30] Shai Halevi and Hugo Krawczyk. Public-Key Cryptography and Password Protocols. In *ACM Conference on Computer and Communications Security*, 1998.

[31] D. P. Jablon. Strong password-only authenticated key exchange. *SIGCOMM Comput. Commun. Rev.*, Vol 26, No. 5, pages 5–26, 1996.

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

[33] J. Kilian. Basing Cryptography on Oblivious Transfer. In *20th STOC*, pages 20–31, 1988.

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

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

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

[37] M. Naor. Bit Commitment using Pseudorandom Generators. *Jour. of Crypto.*, Vol. 4, pages 151–158, 1991.

[38] Moni Naor and Benny Pinkas. Oblivious Transfer and Polynomial Evaluation. In *31st STOC*, pages 245-254, 1999.

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

[40] R. Richardson and J. Kilian. On the Concurrent Composition of Zero-Knowledge Proofs. In *EuroCrypt99*, Springer-Verlag (LNCS 1592), pages 415–431.

[41] R. Rivest, A. Shamir and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *CACM*, Vol. 21, Feb. 1978, pages 120–126.

[42] V. Shoup. On Formal Models for Secure Key Exchange. *Theory of Cryptography Library* Record 99-12, `http://philby.ucsd.edu/cryptolib/`.

[43] M. Steiner, G. Tsudi and M. Waidner. Refinement and extension of encrypted key exchange. *ACM SIGOPS Oper. Syst. Rev.*, Vol. 29, No. 3, pages 22–30, 1995.

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

[45] A.C. Yao. Theory and Application of Trapdoor Functions. In *23rd FOCS*, pages 80–91, 1982.

[46] A.C. Yao. How to Generate and Exchange Secrets. In *27th FOCS*, 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 [23].

A two-party protocol problem is casted 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 alway 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.[33] Thus, an execution in the ideal model proceeds as follows

[33]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.

(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.[34]

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),

$$\begin{array}{lll} (C_1(x,\perp)\,,\,\perp) & \text{if } C_1(x)=\perp & (32)\\ (C_1(x,f_1(C_1(x),y),\perp)\,,\,\perp) & \text{if } C_1(x)\neq\perp \text{ and } C_1(x,f_1(C_1(x),y))=\perp & (33)\\ (C_1(x,f_1(C_1(x),y))\,,\,f_2(C_1(x),y)) & \text{otherwise} & (34)\end{array}$$

- *In case* $C_1(I) = I$ *and* $C_1(I,O) = O$ (i.e., Party 1 is honest),

$$\begin{array}{lll} (\perp\,,\,C_2(y,\perp)) & \text{if } C_2(y)=\perp & (35)\\ (f_1(x,y)\,,\,C_2(y,f_2(x,C_2(y)))) & \text{otherwise} & (36)\end{array}$$

Equation (32) 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 (33) 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 (34) represents

[34] 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.

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 (35) and Equation (36) 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 (33).

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* $\text{real}_{\Pi,\overline{C}}(x, y)$, *is defined as the output pair resulting of the interaction between $C_1(x)$ and $C_2(y)$.*

We 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*

$$\{\text{ideal}_{f,\overline{B}}(x, y)\}_{x,y \text{ s.t. } |x|=|y|} \stackrel{\text{c}}{\equiv} \{\text{real}_{\Pi,\overline{A}}(x, y)\}_{x,y \text{ s.t. } |x|=|y|}$$

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 $\text{real}_{\Pi,\overline{A}}(x, y)$ 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 [14] regarding this issue.

General plausibility results: Assuming the existence of trapdoor permutations, one may provide secure protocols for ANY two-party computation (allowing abort) [46], as well as for ANY multi-party computations with honest majority [27]. 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. The concept of non-malleability was introduced by Dolev et. al. in [19], where they also provide a perfectly binding, (interactive) non-malleable commitment scheme based on any one-way function.

We now bring 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). The sender S commits to a string $\alpha \in_R D$ for some distribution D, and $\mathcal{A}$ wishes to cause T to receive a commitment to β where $\beta \neq \alpha$. ($\mathcal{A}$ is allowed to copy S's commitment and this is not considered a breach of security.) For a given polynomial-time computable relation R, we denote by $\Pi(\mathcal{A}, R)$, the probability that $\mathcal{A}$ generates β such that $(\alpha, \beta) \in R$.

On the other hand, we consider an adversarial simulator $\mathcal{A}'$ who *does not* participate as the receiver in a commitment protocol with S. Rather, $\mathcal{A}'$ sends T a 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, $\mathcal{A}'$ must generate a "related" commitment without any help.

We say that a string commitment scheme is non-malleable if for every 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. Therefore, the fact that $\mathcal{A}$ "saw" a commitment to α did not noticeably help her generate a commitment to β; she could do it by herself anyway. This formalization is conceptually similar to that of semantic security for encryptions.

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 verifer 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. 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 [29]): *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. $\{(P,V^*)(x)\}_{x\in S} \stackrel{\rm def}{=}$ *the output of V^* when interacting with P on common input $x \in S$; and*
2. $\{M^*(x)\}_{x\in S} \stackrel{\rm def}{=}$ *the output of M^* on input $x \in S$.*

Note that whereas P and V^* above are interactive strategies, M^* is a non-interactive computation. The above definition does NOT account for auxiliary information which an adversary may have prior to entering the interaction. Accounting for such auxiliary information is essential for using zero-knowledge proofs as subprotocols inside larger protocols.

A general plausibility result [26]: 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 [40]

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* [8] are sequentially executed where m is a parameter (to be discussed below). Specifically, the first part consists of m iterations, where the $i^{\rm th}$ 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 $i^{\rm th}$ 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 [20] 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 a run will be indistinguishable from a run in which an NP-witness for the membership of the common input (in the language) is used.)

A.5 Seed-Committed Pseudorandom Generators

A seed-committed pseudorandom generator is an efficiently computable deterministic function G mapping a seed to a (commitment,sequence) 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 ([11, 10]) of a seed-committed generator. Let f be a 1–1 one-way function and b a hard-core of f. Then define

$$G(s) = \langle f^{2n}(s), b(s)b(f(s)) \ldots b(f^{2n-1}(s)) \rangle$$

This generator clearly fulfills the requirements: $f^{2n}(s)$ is the commitment and $b(s) \cdots b(f^{2n-1}(s))$ is the sequence.

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 $(s, r) \mapsto (C(s, r), G(s))$ 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 any non-negligible success in generating t such that $f(m) = t$ (for an "unseen" m), must mean that f is not random.

B A Password Attack on the Protocol without the MAC

In this appendix, we describe an attack on our protocol *without* the MAC, such that C can learn a bit of w in every invocation. The main idea is to utilize the (possible) "malleability" of secure two-party protocols. That is, assuming that C is reliable, B's output from the augmented polynomial evaluation is always $Q(w_1 \cdots w_n)$, where $w = w_1 \cdots w_n$. If the polynomial evaluation is malleable, then it may be possible for C to launch a man-in-the-middle attack in which he causes B to receive $Q(0w_2 \cdots w_n)$ instead of $Q(w_1 \cdots w_n)$. However, notice that if $w_1 = 0$, then $Q(0w_2 \cdots w_n) = Q(w_1 \cdots w_n)$ and the parties should notice no difference between this malicious execution and one where C does nothing. That is, the session-key protocol should succeed and the parties should both output accept. On the other hand, if $w_1 = 1$, then A and B have different pre-keys (in fact, by the pairwise independence of Q, the pre-keys $Q(w_1 \cdots w_n)$ and $Q(0w_2 \cdots w_n)$ are independently distributed). Therefore, the session-key protocol should fail and at least one of the parties should output reject. Since C receives the parties' accept/reject output bits, he can infer whether $w_1 = 0$ or $w_1 = 1$. We now show that this attack is indeed possible, and that a well-known secure two-party computation protocol is malleable in the above sense.

The attack is based on an implementation of the polynomial evaluation using Yao's protocol for secure two-party computation [46]. Loosely speaking, in Yao's protocol party A (with input Q) generates an "encrypted" circuit computing $Q(\cdot)$ and sends it to party B. The circuit is such that it reveals nothing in its encrypted form and therefore the value of Q is not learned by B. Furthermore, given a certain series of "keys", the circuit may be decrypted revealing a single value $Q(w)$. Specifically, in order for B to learn $Q(w)$, party A defines $2n$ keys k_i^0 and k_i^1 ($1 \leq i \leq n$) such that given keys $k_1^{w_1}, \ldots, k_n^{w_n}$, the value $Q(w_1 \cdots w_n)$ (and only this value) may be computed from the circuit. Then for every i, party B obtains $k_i^{w_i}$ by oblivious transfer. That is, if $w_i = 0$ then B will obtain k_i^0, otherwise k_i^1. Given these keys and the encrypted circuit, B is able to compute $Q(w)$ (and only $Q(w)$) as required. On the other hand, A learns nothing from the protocol, as B obtains the keys by oblivious transfer. The crucial point regarding the malleability of the protocol is that B executes an *independent* oblivious transfer for each bit of his input.

We now show that Yao's protocol is malleable as previously described. Specifically, we show that channel C can cause B to receive $Q(0w_2 \cdots w_n)$ instead of $Q(w)$. C's strategy is to pass (almost) all messages of the protocol untouched between A and B. The only exception is that C causes B to receive k_1^0 (instead of $k_1^{w_1}$) in the first oblivious transfer described above. C can easily do this by playing the *receiver* in the first oblivious transfer with A and obtaining k_1^0. Next, C plays the *sender* in the first oblivious transfer with B where he defines both the 0 and 1 strings to be k_1^0. In this way, whatever B inputs into the oblivious transfer, he receives k_1^0. This can be generalized so that for any i, party B receives $Q(w_1 \cdots w_{i-1} 0 w_{i+1} \cdots w_n)$ instead of $Q(w)$ (i.e., the i'th bit is always 0).

Now, let A and B be parties running a modified version of our session-key protocol via C, where no MAC is sent by A in the validation stage. Furthermore, let the protocol implementation be such that the polynomial evaluation is executed using Yao's protocol. Then, the above strategy can be used by C to learn any single bit of w in each invocation of the protocol. C does this by reliably passing all messages between A and B during the protocol execution, *except* that C causes B to receive $Q(w_1 \cdots w_{i-1} 0 w_{i+1} \cdots w_n)$ in the polynomial evaluation. Then, at the conclusion of the protocol, C receives B's accept/reject bit. As we have described, if B accepts then w_i must equal 0, because otherwise A and B have different pre-keys. On the other hand, if B rejects, then w_i must equal 1 (because C did not interfere in any other part of the protocol) . The above is therefore a strategy for which C can learn one bit of the password in every invocation. Notice also that the attack *always* succeeds and that C can specify the bit that he wishes to learn. This clearly contradicts the password secrecy requirement of the protocol.