J. Cryptol. (2008) 21: 52–96
DOI: 10.1007/s00145-007-9008-4

Journal of
CRYPTOLOGY

Simpler Session-Key Generation from Short Random Passwords*

Minh-Huyen Nguyen† and Salil Vadhan‡
Harvard University, 33 Oxford Street, Cambridge, MA 02138, USA
mnguyen@eecs.harvard.edu; salil@eecs.harvard.edu

Communicated by Oded Goldreich

Received 23 September 2004 and revised 11 November 2006
Online publication 18 September 2007

Abstract. Goldreich and Lindell (CRYPTO '01) recently presented the first protocol for password-authenticated key exchange in the standard model (with no common reference string or set-up assumptions other than the shared password). However, their protocol uses several heavy tools and has a complicated analysis.

We present a simplification of the Goldreich–Lindell (GL) protocol and analysis for the special case when the dictionary is of the form $\mathcal{D} = \{0, 1\}^d$ i.e., the password is a short string chosen uniformly at random (in the spirit of an ATM PIN number). The security bound achieved by our protocol is somewhat worse than the GL protocol. Roughly speaking, our protocol guarantees that the adversary can "break" the scheme with probability at most $O(\text{poly}(n)/|\mathcal{D}|)^{\Omega(1)}$, whereas the GL protocol guarantees a bound of $O(1/|\mathcal{D}|)$.

We also present an alternative, more natural definition of security than the "augmented definition" of Goldreich and Lindell, and prove that the two definitions are equivalent.

Key words. Human-memorizable passwords, Key exchange, Authentication, Cryptographic protocols, Secure two-party computation.

1. Introduction

What is the minimal amount of information that two parties must share in order to perform nontrivial cryptography? This fundamental question is at the heart of many of the major distinctions we draw in cryptography. Classical private-key cryptography assumes that the legitimate parties share a long random key. Public-key cryptography mitigates this by allowing the sharing of information to be done through public keys

---

* An extended abstract of this paper appeared in the *First Theory of Cryptography Conference* (*TCC '04*) [22].

† Supported by NSF grant CCR-0205423 and ONR grant N00014-04-1-0478.

‡ Supported by NSF grant CCR-0205423, a Sloan Research Fellowship, and ONR grant N00014-04-1-0478. Part of this work done while at the Radcliffe Institute for Advanced Study.

that need not be hidden from the adversary. However, in both cases, the amount of information shared by the legitimate parties (e.g., as measured by mutual information) needs to be quite large. Indeed, the traditional view is that security comes from the adversary's inability to exhaustively search the keyspace.

Thus it is very natural to ask: *can we do nontrivial cryptography using "low-entropy" keys* (i.e., using a keyspace that is feasible to exhaustively search)? In addition to being a natural theoretical question, it has clear relevance to the many "real-life" situations where we need security but only have a low-entropy key (e.g., an ATM PIN number, or human-chosen password on a website).

Public-key cryptography provides an initial positive answer to this question: key-exchange protocols, as in [10], do not require *any* prior shared information. However, this holds only for passive adversaries, and it is well known that without any prior shared information between the legitimate parties, an active adversary can always succeed through a person-in-the-middle attack. Thus, it remains an interesting question to achieve security against active adversaries using a low-entropy shared key. This has led researchers to consider the problem of *password-authenticated key exchange,* which we describe next.

Password-Authenticated Key Exchange The password-authenticated key exchange problem was first suggested as a topic for research by Bellovin and Merritt [4]. We assume that two parties, Alice and Bob, share a password w chosen uniformly at random from a dictionary $\mathcal{D} \subseteq \{0,1\}^n$. This dictionary can be very small, e.g., $|\mathcal{D}| = \mathrm{poly}(n)$, and in particular it may be feasible for an adversary to exhaustively search it. Our aim is to construct a protocol enabling Alice and Bob to generate a "random" session key $K \in \{0,1\}^n$, which they can subsequently use for standard private-key cryptography. We consider an active adversary that completely controls the communication channel between Alice and Bob. The adversary can intercept, modify, drop, and delay messages, and in particular can attempt to impersonate either party through a person-in-the-middle attack.

Our goal is that, even after the adversary mounts such an attack, Alice and Bob will generate a session key that is "indistinguishable" from uniform even given the adversary's view. However, our ability to achieve this goal is limited by two unpreventable attacks. First, since the adversary can block all communication, it can prevent one or both of the parties from completing the protocol and obtaining a session key. Second, the adversary can guess a password $\tilde{w}$ chosen uniformly from the dictionary $\mathcal{D}$ and attempt to impersonate one of the parties. With probability $1/|\mathcal{D}|$, the guess equals the real password (i.e., $\tilde{w} = w$), and the adversary will succeed in the impersonation and therefore learn the session key. Thus, we revise our goal to effectively limit the adversary to these two attacks. Various formalizations for this problem have been developed through several works [2,3,7,15,17,25]. We follow the definitional framework of Goldreich and Lindell [15], which is described in more detail in Sect. 2.

In addition to addressing what can be done with a minimal amount of shared information, the study of this problem is useful as another testbed for developing our understanding of *concurrency* in cryptographic protocols. The concurrency implicitly arises from the person-in-the-middle attack, which we can view as two simultaneous executions of the protocol, one between Alice and the adversary and the other between Bob and the adversary.

The first protocols for the password-authenticated key exchange problem were proposed in the security literature, based on informal definitions and heuristic arguments (e.g., [5,27]). The first rigorous proofs of security were given in the ideal cipher and random oracle models [2,7,19]. Only recently were rigorous solutions without random oracles given, in independent works by Goldreich and Lindell [15] and Katz, Ostrovsky, and Yung [18]. One of the main differences between these two protocols is that the KOY protocol (and the subsequent protocol of [13]) is in the "public parameters model," requiring a string to be generated and published by a trusted third party, whereas the GL protocol requires no set-up assumption other than the shared password. Thus, even though the KOY protocol has a number of practical and theoretical advantages over the GL protocol (which we will not enumerate here), the GL protocol is more relevant to our initial question about the minimal amount of shared information needed for nontrivial cryptography.

The Goldreich–Lindell Protocol As mentioned above, the Goldreich–Lindell protocol [15] is remarkable in that the only set-up assumption it requires is that the two parties share a password chosen at random from an arbitrary dictionary. Furthermore, their protocol can be based on general complexity assumptions (i.e., the existence of enhanced trapdoor permutations), can be implemented in a constant number of rounds (under stronger assumptions), and achieves a nearly optimal security bound (the adversary has probability only $O(1/|\mathcal{D}|)$ of "breaking" the scheme).

Despite giving such a strong result, the Goldreich–Lindell protocol does not leave us with a good understanding of the password-authenticated key-exchange problem. First, the protocol makes use of several "heavy" tools: secure two-party polynomial evaluation (building on [21], who observed that this yields a protocol for password-authenticated key exchange against passive adversaries), nonmalleable commitments (as suggested in [6]), and the specific concurrent zero-knowledge proof of Richardson and Kilian [24]. It is unclear whether all of these tools are really essential for solving the key-exchange problem. Second, the proof of the protocol's security is extremely complicated. Goldreich and Lindell do introduce nice techniques for analyzing concurrent executions (arising from the person-in-the-middle attack) of two-party protocols whose security is only guaranteed in the stand-alone setting (e.g. the polynomial evaluation), but these techniques are applied in an intricate manner that seems inextricably tied to the presence of the nonmalleable commitment and zero-knowledge proof. Finally, finding an efficient instantiation of the Goldreich–Lindell protocol would require finding efficient instantiations of all three of the heavy tools mentioned above, which seems difficult. In particular, the Richardson–Kilian zero-knowledge proof is used to prove an **NP** statement that asserts the consistency of a transcript of the nonmalleable commitment, a standard commitment, and the output of an iterated one-way permutation. For such an **NP** statement, it seems difficult to avoid using a generic zero-knowledge proof system for **NP**, which is almost always inefficient due to the use of Cook's theorem.

Our Protocol Our main result is a simplification of the Goldreich–Lindell protocol and analysis for the special case when the dictionary is of the form $\mathcal{D} = \{0, 1\}^d$, i.e., the password is a short string chosen uniformly at random from $\{0, 1\}^d$, for $d \geq 3 \log n$ where n is the security parameter. More generally, the password can be chosen uniformly at random from any fixed dictionary of size 2^d whose elements we can efficiently enumerate (because the enumeration provides a bijection with $\{0, 1\}^d$). Note that this special case still retains the main features of the problem: the person-in-the-middle attack and the resulting concurrency issues are still present, and the adversary can still exhaustively search the dictionary (since we allow the password length d to be as small as $O(\log n)$, where n is the security parameter). Our protocol is still far from practical (and cannot be used with passwords as short as 4-digit ATM numbers), but we view it as a theoretical step in our understanding of a natural and important cryptographic problem.

Though our protocol cannot be used directly for arbitrary dictionaries, it can be converted into one for arbitrary dictionaries in the *common reference string model* (using the common reference string as the seed of a randomness extractor [23]). For dictionaries $\mathcal{D} \subset \{0, 1\}^n$, the common reference string is a uniform string of only logarithmic length (specifically, $O(\log n + \log |\mathcal{D}|)$), and thus retains the spirit of minimizing the amount of shared information between the legitimate parties. In contrast, the previous protocols in the public parameters model [13,18] require a public string of length poly(n) with special number-theoretic structure.

The main way in which we simplify the GL protocol is that we remove the nonmalleable commitments and the Richardson–Kilian zero-knowledge proof. Instead, our protocol combines secure polynomial evaluation with a combinatorial tool (i.e., almost pairwise-independent hashing), in addition to using "lightweight" cryptographic primitives also used in [15] (one-way permutations, one-time MACs, standard commitments). Our analysis is also similarly simpler. While it has the same overall structure as the analysis in [15] and utilizes their techniques for applying the stand-alone properties of the polynomial evaluation in the concurrent setting, it avoids dealing with the nonmalleable commitments and the zero-knowledge proof (which is the most complex part of the GL analysis).

Removing the nonmalleable commitments and the RK zero-knowledge proof has two additional implications. First, finding an efficient implementation of our protocol only requires finding an efficient protocol for secure polynomial evaluation (in fact, only for linear polynomials).[1] Since this is a highly structured special case of secure two-party computation, it does not seem beyond reach to find an efficient protocol. Indeed, Naor and Pinkas [21] have already given an efficient polynomial evaluation protocol for passive adversaries. Second, our protocol can be implemented in a constant number of rounds assuming only the existence of trapdoor permutations, whereas implementing the Goldreich–Lindell protocol in a constant number of rounds requires additional assumptions, such as the existence of claw-free permutations (for a round-efficient version of the Richardson–Kilian zero-knowledge proof, see [15]) and some sort of exponential hardness assumption (to use [1] and obtain constant-round non-malleable commitments). Though our protocol is not efficient and cannot be used in practice, this theoretical work will hopefully help lead the way to protocols that are both simpler and efficient.

We note that the security bound achieved by our protocol is somewhat worse than in previous works. Roughly speaking, our protocol for a password chosen uniformly at random from the dictionary $\mathcal{D} = \{0, 1\}^d$ guarantees that the adversary can "break"

[1] Actually, we require a slightly augmented form of polynomial evaluation, in which one of the parties commits to its input beforehand and the protocol ensures consistency with this committed input.

the scheme with probability at most $O(n^3/|\mathcal{D}|)^{1/4}$, whereas previous works guarantee a bound of $O(1/|\mathcal{D}|)$. A security bound with linear dependency on the size of the dictionary is of course preferable for both conceptual and practical reasons, but again, our protocol should be viewed as a stepping stone towards more efficient protocols with better security bounds.

An additional result in our paper involves the definition of security in [15]. As pointed out by Rackoff (cf. [2]), it is important that a key exchange protocol provide security even if the party who completes the protocol first starts using the generated key in some application before the second party completes the protocol. In order to address this issue, Goldreich and Lindell [15] augmented their definition with a "session-key challenge", in which the adversary is given either the generated key or a uniform string with probability 1/2 upon the first party's completion of the protocol. We present an arguably more natural definition that directly models the use of the generated key in an arbitrary application, and prove its equivalence to the augmented definition of Goldreich and Lindell [15]. (This result is analogous to the result of Shoup [25] for non-password-based key exchange protocols.)

Organization In Sect. 2, we formalize the problem of session-key generation using human passwords. We first provide the basic security definition of Goldreich and Lindell [15] and give their augmented definition (which deals with the issue of one party using the generated key before the other party completes the protocol). We then present our alternative, more natural definition of security and prove its equivalence to the augmented definition of Goldreich and Lindell.

In Sect. 3, we provide an overview of our protocol. It was observed in [21] that a secure protocol for *polynomial evaluation* yields a protocol for session-key generation that is secure against passive adversaries. In [15], Goldreich and Lindell work from the intuition that by augmenting a secure protocol for polynomial evaluation with additional mechanisms, one can obtain a protocol for session-key generation that is secure against active adversaries. Our protocol also comes from this intuition, but the additional tools we are using are different.

In Sect. 4, we state our main security theorems and provide an overview of the proof of security. As in [15], the proof reduces the case of active adversaries to the case of passive adversaries by establishing a "Key-Match Property," which states that if certain "pre-keys" computed by the two honest parties are different, then one of the parties will reject with high probability.

Section 5 contains the proof of security against passive adversaries. In Sects. 6 and 7, we establish the Key-Match Property (each section handling a different 'scheduling' of the two concurrent interactions between the adversary and the honest parties). It is here that our analysis differs from and is significantly simpler than that of Goldreich and Lindell [15]. In Sect. 8, we adapt the proofs of [15] to show that the simulation-based definition of security against active adversaries is satisfied.

In Sect. 9, we state additional security theorems and show how the shared dictionary $\mathcal{D} = \{0,1\}^d$ can be realized from several other types of dictionaries. In Appendix A, we recall definitions of secure two-party computation and in Appendix B we give a construction of almost pairwise-independent hash functions that can be used in our protocol.

2. Definition of Security

We adopt the notation of Goldreich and Lindell and refer the reader to [15] for more details.

- C denotes the probabilistic polynomial time adversary through which the honest parties A and B communicate. We model this communication by giving C oracle access to a single copy of A and a single copy of B. Here the oracles A and B have memory and represent honest parties executing the session-key generation protocol. We denote by $C^{A(x),B(y)}(\sigma)$ an execution of C with auxiliary input σ when it communicates with A and B, with respective inputs x and y. The output of the channel C from this execution is denoted by $\text{output}(C^{A(x),B(y)}(\sigma))$.
- The security parameter is denoted by n. The password dictionary is denoted by $\mathcal{D} \subseteq \{0,1\}^n$ and we write $\epsilon = \frac{1}{|\mathcal{D}|}$.

We denote by U_n the uniform distribution over strings of length n, by $\text{neg}(n)$ an arbitrary negligible function, and write $x \overset{R}{\leftarrow} S$ when x is chosen uniformly from the set S.

For a function $\gamma : \mathbb{N} \to [0,1]$, we say that the probability ensembles $\{X_n\}$ and $\{Y_n\}$ are $(1-\gamma)$*-indistinguishable* (denoted by $\{X_n\} \overset{\gamma}{\equiv} \{Y_n\}$) if for every probabilistic polynomial-time distinguisher D, every polynomial p, every n, and every auxiliary input $\sigma_n \in \{0,1\}^{p(n)}$, we have

$$|\Pr[D(X_n, \sigma_n) = 1] - \Pr[D(Y_n, \sigma_n) = 1]| < \gamma(n) + \text{neg}(n).$$

We say that $\{X_n\}$ and $\{Y_n\}$ are *computationally indistinguishable*, which we denote by $X_n \overset{\text{comp}}{\equiv} Y_n$, if they are 1-indistinguishable. We say that $\{X_n\}$ is $(1-\gamma)$*-pseudorandom* if it is $(1-\gamma)$-indistinguishable from U_n.

We will also consider indistinguishability for probability ensembles $\{X_w\}_{w\in S}$ and $\{Y_w\}_{w\in S}$ that are indexed by a set of strings S in which case we require that for every probabilistic polynomial-time distinguisher D, every polynomial p, every $w \in S$, and every auxiliary input $\sigma \in \{0,1\}^{p(|w|)}$, we have:

$$|\Pr[D(X_w, w, \sigma) = 1] - \Pr[D(Y_w, w, \sigma) = 1]| < \gamma(|w|) + \text{neg}(|w|).$$

We will now formalize the problem of session-key generation using human passwords. We first provide the basic security definition of Goldreich and Lindell [15] in Sect. 2.1 and give their augmented definition (which deals with the issue of one party using the generated key before the other party completes the protocol) in Sect. 2.2. In Sect. 2.3, we present our alternative, more natural definition of security and prove its equivalence to the augmented definition of Goldreich and Lindell.

2.1. *The Initial Definition*

The definition in [15] follows the standard paradigm for secure computation: define an ideal functionality (using a trusted third party) and require that every adversary attacking the real protocol can be simulated by an ideal adversary attacking the ideal functionality. Note that in the real protocol, the active adversary C can prevent one or both of the parties A and B from having an output. Thus, in the ideal model, we will allow C_{ideal} to specify two input bits, dec_C^A and dec_C^B, that determine whether A and B obtain a session key or not.

Ideal model Let A, B be the honest parties and let $\mathrm{C_{ideal}}$ be any probabilistic polynomial-time ideal adversary with auxiliary input σ.

1. A and B receive $w \xleftarrow{R} \mathcal{D}$.
2. A and B both send w to the trusted party.
3. $\mathrm{C_{ideal}}$ sends $(\mathrm{dec}_C^A, \mathrm{dec}_C^B)$ to the trusted party.
4. The trusted party chooses $K \xleftarrow{R} \{0,1\}^n$. For each party $i \in \{A, B\}$, the trusted party sends K if $\mathrm{dec}_C^i = 1$ and sends $\perp$ if $\mathrm{dec}_C^i = 0$. Hence $\mathrm{output}(A) = K$ if $\mathrm{dec}_C^A = 1$, and $\perp$ otherwise ($\mathrm{output}(B)$ is defined similarly).

The ideal distribution is defined by:

$$\mathrm{IDEAL}_{\mathrm{C_{ideal}}}(\mathcal{D}, \sigma) = (w, \mathrm{output}(A), \mathrm{output}(B), \mathrm{output}(\mathrm{C_{ideal}}(\sigma))).$$

We note that this description of the ideal model differs slightly from the original definition in [15] since we allow B to finish first and A to reject in the ideal model (this is to take into account protocols in which no party is guaranteed to terminate with a session key). However, as described in Sect. 3.3, our protocol will guarantee that A always accepts. Moreover, we will show that any real adversary can be simulated by an ideal adversary who always chooses A to conclude first and accept.

Real model Let A, B be the honest parties and let C be any probabilistic polynomial-time real adversary with auxiliary input σ.

At some initialization stage, A and B receive $w \xleftarrow{R} \mathcal{D}$. The real protocol is executed by A and B communicating via C. We will augment C's view of the protocol with A and B's decision bits, denoted by dec_A and dec_B, where $\mathrm{dec}_A = \texttt{reject}$ if $\mathrm{output}(A) = \perp$, and $\mathrm{dec}_A = \texttt{accept}$ otherwise (dec_B is defined similarly). (Indeed, in typical applications, the decisions of A and B will be learned by the real adversary C: if A obtains a session key, then it will use it afterwards; otherwise, A will stop communication or try to re-initiate an execution of the protocol.) C's augmented view is denoted by $\mathrm{output}(C^{A(w),B(w)}(\sigma))$.

The real distribution is defined by:

$$\mathrm{REAL}_C(\mathcal{D}, \sigma) = (w, \mathrm{output}(A), \mathrm{output}(B), \mathrm{output}(C^{A(w),B(w)}(\sigma))).$$

One might want to say that a protocol for password-based session-key generation is secure if the above ideal and real distributions are computationally indistinguishable. Unfortunately, as pointed in [15], an active adversary can guess the password and successfully impersonate one of the parties with probability $\frac{1}{|\mathcal{D}|}$. This implies that the real and ideal distributions are always distinguishable with probability at least $\frac{1}{|\mathcal{D}|}$. Thus we will only require that the distributions be distinguishable with probability at most $O(\gamma)$ where the goal is to make γ as close to $\frac{1}{|\mathcal{D}|}$ as possible. In the case of a passive adversary, we require that the real and ideal distributions be computationally indistinguishable (for all subsequent definitions, this requirement will be implicit).

Definition 2.1 (Initial definition). A protocol for password-based authenticated session-key generation is $(1-\gamma)$-*secure* for the dictionary $\mathcal{D} \subseteq \{0,1\}^n$ (where γ is a function of the dictionary size $|\mathcal{D}|$ and n) if:

1. For every real *passive* adversary, there exists an ideal adversary $\mathrm{C}_{\mathrm{ideal}}$ that always sends (1,1) to the trusted party such that for every auxiliary input $\sigma \in \{0,1\}^{\mathrm{poly}(n)}$,
$$\{\mathrm{IDEAL}_{\mathrm{C}_{\mathrm{ideal}}}(\mathcal{D},\sigma)\} \stackrel{\mathrm{comp}}{\equiv} \{\mathrm{REAL}_C(\mathcal{D},\sigma)\}.$$
2. For every real adversary C, there exists an ideal adversary $\mathrm{C}_{\mathrm{ideal}}$ such that for every auxiliary input $\sigma \in \{0,1\}^{\mathrm{poly}(n)}$,
$$\{\mathrm{IDEAL}_{\mathrm{C}_{\mathrm{ideal}}}(\mathcal{D},\sigma)\} \stackrel{O(\gamma)}{\equiv} \{\mathrm{REAL}_C(\mathcal{D},\sigma)\}.$$

By the discussion above, the best we can hope for is $\gamma = \frac{1}{|\mathcal{D}|}$. Note that in [15], their definition and protocol refer to any dictionary $\mathcal{D} \subseteq \{0,1\}^n$ and $\gamma = \frac{1}{|\mathcal{D}|}$. In contrast, our protocol will be $(1-\gamma)$-secure for dictionaries of the form $\mathcal{D} = \{0,1\}^d$ and $\gamma = (\frac{\mathrm{poly}(n)}{|\mathcal{D}|})^{\Omega(1)}$.

Following [15] (but unlike some of the other previous work), the above definition refers to only a single execution of the session-key generation protocol. As in [15], it can be shown that security is maintained for multiple sequential executions, but security is not guaranteed for concurrent executions using the same password.

2.2. *Security with Respect to the Session-Key Challenge*

The above definition is actually not completely satisfying because of a subtle point raised by Rackoff: the adversary controls the scheduling of the interactions (A, C) and (C, B) so the honest parties do not necessarily end at the same time. Assume that A completes the protocol first. Then A might use its session key K_A before the interaction (C, B) is completed: A's use of K_A leaks information which C might use in its interaction with B to learn K_A, K_B or the password w.

In [15], Goldreich and Lindell augment the above definition with a *session-key challenge* to address this issue. Suppose that A completes the protocol first and outputs a session key K, then the adversary is given a session key challenge K_β, which is the session key K with probability $1/2$ (i.e., $\beta = 1$) or a truly random string K_0 with probability $1/2$ (i.e., $\beta = 0$). The adversary C will be given the session-key challenge in both the ideal and real models, as soon as the first honest party outputs a session key K. We call the resulting definition *security with respect to the session-key challenge*.

Ideal model Let A, B be the honest parties and let $\mathrm{C}_{\mathrm{ideal}}$ be any probabilistic polynomial-time ideal adversary with auxiliary input σ.

1. A and B receive $w \stackrel{\mathrm{R}}{\leftarrow} \mathcal{D}$.
2. A and B both send w to the trusted party.
3. $\mathrm{C}_{\mathrm{ideal}}$ decides which party $i \in \{A, B\}$ concludes first and whether it is a successful execution or not, i.e., $\mathrm{C}_{\mathrm{ideal}}$ sends a pair (i, dec^i_C) to the trusted party.

4. The trusted party chooses $K \xleftarrow{R} \{0,1\}^n$. If $\mathrm{dec}_C^i = 1$, the trusted party sends K to party i; otherwise it sends $\perp$.
5. Session-key challenge: if party i received $\perp$, then the trusted party gives $\perp$ to $\mathrm{C}_{\mathrm{ideal}}$. Otherwise, the trusted party chooses $\beta \xleftarrow{R} \{0,1\}$ and gives $\mathrm{C}_{\mathrm{ideal}}$ the string K_β where $K_1 = K$ and $K_0 \xleftarrow{R} \{0,1\}^n$.
6. $\mathrm{C}_{\mathrm{ideal}}$ decides whether the second party's execution is successful or not, i.e., $\mathrm{C}_{\mathrm{ideal}}$ sends dec_C^j for $j \neq i$ to the trusted party.
7. If $\mathrm{dec}_C^j = 1$, the trusted party sends K to party j. Otherwise, it sends $\perp$.

The augmented ideal distribution is defined by:

$$\text{IDEAL-SK}_{\mathrm{C}_{\mathrm{ideal}}}(\mathcal{D}, \sigma) = (w, \mathrm{output}(A), \mathrm{output}(B), \mathrm{output}(\mathrm{C}_{\mathrm{ideal}}(\sigma, K_\beta)), \beta).$$

Real model At some initialization stage, A and B receive $w \xleftarrow{R} \mathcal{D}$. C has oracle access to a single copy of $A(w)$ and a single copy of $B(w)$. The adversary C controls which party (A or B) concludes first. If the first party concludes with $\perp$, then C is given $\perp$. If the first party concluding outputs locally a session key K, then a bit $\beta \xleftarrow{R} \{0,1\}$ is chosen and C is given the session-key challenge K_β where $K_1 = K$ and $K_0 \xleftarrow{R} \{0,1\}^n$. C completes its interaction with the other party.

The augmented real distribution is defined by:

$$\text{REAL-SK}_C(\mathcal{D}, \sigma) = (w, \mathrm{output}(A), \mathrm{output}(B), \mathrm{output}(C^{A(w),B(w)}(\sigma, K_\beta)), \beta).$$

Definition 2.2 (Security with respect to the session-key challenge [15]). A protocol for password-based authenticated session-key generation is $(1-\gamma)$-*secure with respect to the session-key challenge* for the dictionary $\mathcal{D} \subseteq \{0,1\}^n$ if for every real adversary C, there exists $\mathrm{C}_{\mathrm{ideal}}$ such that for every auxiliary input $\sigma \in \{0,1\}^{\mathrm{poly}(n)}$,

$$\{\text{IDEAL-SK}_{\mathrm{C}_{\mathrm{ideal}}}(\mathcal{D}, \sigma)\} \stackrel{O(\gamma)}{\equiv} \{\text{REAL-SK}_C(\mathcal{D}, \sigma)\}.$$

Goldreich and Lindell give some intuition as to why the session-key challenge solves the flaw mentioned earlier. First, note that the ideal adversary cannot distinguish between the case $\beta = 0$ and the case $\beta = 1$ since in the ideal model, both K_0 and K are truly uniform strings. Consider the real adversary who has been given the session-key challenge: if C has been given K_0, then the session-key challenge does not help C in attacking the protocol, since C could generate K_0 on its own. Suppose that instead C has been given K and that C can somehow use it to attack the protocol (this corresponds to the situation where A uses the session-key K; $C(K)$ can simulate A's use of the key), then it would mean that C can tell whether $\beta = 0$ or $\beta = 1$, which is not possible if the protocol is secure with respect to the session-key challenge.

2.3. *Security with Respect to the Environment*

Suppose that A completes the protocol first and outputs a session key K. Our intuitive notion of security is that *no matter how A uses its session-key K* before the execution (C, B) is completed, the ideal and real distributions should be $(1 - O(\gamma))$-indistinguishable. It is not immediate that the session-key challenge captures this. Thus we propose an alternative augmentation to Definition 2.1 that corresponds more directly to this goal.

We model the different ways the party A could use its session-key K by considering an arbitrary probabilistic polynomial time machine Z that is given the key K (as soon as A outputs a session-key K) and interacts with the adversary in both the ideal and real models. This is similar to the "application" queries in Shoup's model for (non-password-based) secure key-exchange [25], which was later extended to password protocols in [7]. Z can also be thought of in terms of "environment" as in Canetti's notion of UC security [8]: Z models an arbitrary environment (or application) in which the key generated by the session-key generation protocol is used (note that this is not as general as the definition of Canetti since the environment Z is only given the session-key and not the password w).

Examples of environments follow:

1. $Z(K) = \perp$: A does not use its session-key.
2. $Z(K) = K$: A publicly outputs its session-key.
3. $Z(K) = \begin{cases} K & \text{with probability } 1/2, \\ U_n & \text{with probability } 1/2. \end{cases}$
   This corresponds to the session-key challenge.
4. $Z(K) = \mathrm{Enc}_K(0^n)$: A uses its session-key for secure private-key encryption.
5. C sends a query m_1, $Z(K)$ answers with $\mathrm{Enc}_K(m_1)$, C sends a query m_2, $Z(K)$ answers with $\mathrm{Enc}_K(m_2)$ and so on. This corresponds to an interactive environment Z which models a chosen plain-text attack.

We call the definition obtained by adding (in both the ideal and real models) the environment Z *security with respect to the environment*. Informally, a real protocol is secure with respect to the environment if every adversary attacking the real protocol and interacting with an arbitrary environment can be simulated, with probability $1 - O(\gamma)$, by an ideal adversary attacking the ideal functionality and interacting with the same environment in the ideal model. (More precisely, for every real adversary, there should be a single ideal adversary that simulates it well for *every* environment.)

Ideal model Let A and B be the honest parties, $\mathrm{C}_{\mathrm{ideal}}$ any probabilistic polynomial-time ideal adversary with auxiliary input σ and Z any probabilistic polynomial-time with auxiliary input τ.

1. A and B receive $w \stackrel{\mathrm{R}}{\leftarrow} \mathcal{D}$.
2. A and B both send w to the trusted party.
3. $\mathrm{C}_{\mathrm{ideal}}$ decides which party $i \in \{A, B\}$ concludes first and whether it is a successful execution or not, i.e., $\mathrm{C}_{\mathrm{ideal}}$ sends (i, dec_C^i) to the trusted party.
4. The trusted party chooses $K \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^n$. If $\mathrm{dec}_C^i = 1$, it sets $L_1 = K$; otherwise, $L_1 = \perp$. The trusted party sends L_1 to party i and Z.
5. $\mathrm{C}_{\mathrm{ideal}}$ interacts with $Z(L_1, \tau)$.
6. $\mathrm{C}_{\mathrm{ideal}}$ decides whether the second party's execution is successful or not, i.e., $\mathrm{C}_{\mathrm{ideal}}$ sends dec_C^j for $j \neq i$ to the trusted party.
7. If $\mathrm{dec}_C^j = 1$, the trusted party sets $L_2 = K$; otherwise, $L_2 = \perp$. It sends L_2 to party j.

The ideal distribution is defined by:

$$\mathrm{IDEAL}_{Z,\tau,\mathrm{C}_{\mathrm{ideal}}}(\mathcal{D},\sigma) = (w, \mathrm{output}(A), \mathrm{output}(B), \mathrm{output}(Z(L_1,\tau)), \mathrm{output}({\mathrm{C}_{\mathrm{ideal}}}^{Z(L_1,\tau)}(\sigma))).$$

Real model At some initialization stage, A and B receive $w \xleftarrow{R} \mathcal{D}$. C has oracle access to a single copy of $A(w)$ and a single copy of $B(w)$. The adversary C controls which party (A or B) concludes first. Let $M_1 \in \{0,1\}^n \cup \perp$ be the output of the first concluding party. C interacts with $Z(M_1,\tau)$ and completes its interaction with the other party.

The real distribution is defined by:

$$\mathrm{REAL}_{Z,\tau,C}(\mathcal{D},\sigma) = (w, \mathrm{output}(A), \mathrm{output}(B), \mathrm{output}(Z(M_1,\tau)), \mathrm{output}(C^{A(w),B(w),Z(M_1,\tau)}(\sigma))).$$

Definition 2.3 (Security with respect to the environment). A protocol for password-based authenticated session-key generation is $(1-\gamma)$-*secure with respect to the environment* for the dictionary $\mathcal{D} \subseteq \{0,1\}^n$ if for every probabilistic polynomial-time C, there exists $\mathrm{C}_{\mathrm{ideal}}$ such that for every auxiliary input $\sigma \in \{0,1\}^{\mathrm{poly}(n)}$ and every probabilistic polynomial-time Z with every auxiliary input $\tau \in \{0,1\}^{\mathrm{poly}(n)}$,

$$\{\mathrm{IDEAL}_{Z,\tau,\mathrm{C}_{\mathrm{ideal}}}(\mathcal{D},\sigma)\} \overset{O(\gamma)}{\equiv} \{\mathrm{REAL}_{Z,\tau,C}(\mathcal{D},\sigma)\}.$$

Note that security with respect to the environment implies security with respect to the session-key challenge since it suffices to consider the probabilistic polynomial-time $Z(K)$ which generates $\beta \xleftarrow{R} \{0,1\}$ and outputs the key K if $\beta = 1$ or a truly random string K_0 if $\beta = 0$. We show that the two definitions are actually equivalent:

Theorem 2.4. *A protocol (A,B) is $(1-\gamma)$-secure with respect to the session-key challenge iff it is $(1-\gamma)$-secure with respect to the environment.*

This is similar to a result of Shoup [25] showing the equivalence of his definition and the Bellare–Rogaway [3] definition for non-password-based key exchange. The "application" queries in Shoup's definition are analogous to our environment Z, and the "test" queries in [3] are analogous to the session-key challenge. Though both of these definitions have been extended to password-authenticated key exchange [2,7], it is not immediate that Shoup's equivalence result extends directly to our setting. For example, the definitions of [2,3] are not simulation-based and do not directly require that the password remain pseudorandom, whereas here we are relating two simulation-based definitions that do ensure the password's secrecy.

Given Theorem 2.4, the relationship between security with respect to the environment and security with respect to the session-key challenge is analogous to the relationship between semantic security and indistinguishability for encryption schemes [16,20]. Though both are equivalent, the former captures our intuitive notion of security better, but the latter is typically easier to establish for a given protocol (as it involves only taking into account a specific environment Z).

The intuition for the proof of Theorem 2.4 is that for every adversary C against security with respect to the environment, there exists an adversary C' against security with respect to the session-key challenge that can simulate both the environment Z and C. Indeed, C' is given the session-key challenge K_β and can simulate both $Z(K_\beta)$ and C (interacting with $Z(K_\beta)$) on its own using K_β. Note that here Z is only run with the actual session key with probability $1/2$ (namely, when $\beta = 1$), whereas the definition of security with respect to the environment always refers to Z run with the true session key. Intuitively, however, this difference should not matter, because the two cases $\beta = 0$ and $\beta = 1$ are indistinguishable in the ideal model for security with respect to the session-key challenge.

Proof. For conciseness of notation in the proof, we omit "output" in the distributions.

Let (A, B) be a protocol that is secure with respect to the session-key challenge. To prove the theorem, it suffices to prove that Definition 2.3 holds that is, for every probabilistic polynomial-time C, there exists a probabilistic polynomial-time $\mathrm{C}_{\text{ideal}}$ such that for every Z and every auxiliary input τ:

$$\{w, A, B, Z(M_1, \tau), C^{A(w),B(w),Z(M_1,\tau)}(\sigma)\} \overset{O(\gamma)}{\equiv} \{w, A, B, Z(L_1, \tau), {\mathrm{C}_{\text{ideal}}}^{Z(L_1,\tau)}(\sigma)\},$$

where M_1 is the output of the first concluding party in the real execution $C^{A(w),B(w)}$ and L_1 is the output of the first concluding party in the ideal execution.

We denote by M_2 the output of the second concluding party in the real execution $C^{A(w),B(w)}$ and by L_2 the output of the second concluding party in the ideal execution. Hence, we want to prove that for every probabilistic polynomial-time C, there exists a probabilistic polynomial-time $\mathrm{C}_{\text{ideal}}$ such that for every Z and every auxiliary input τ:

$$\begin{aligned}&\{w, M_1, M_2, Z(M_1, \tau), C^{A(w),B(w),Z(M_1,\tau)}(\sigma)\}\\ &\overset{O(\gamma)}{\equiv} \{w, L_1, L_2, Z(L_1, \tau), {\mathrm{C}_{\text{ideal}}}^{Z(L_1,\tau)}(\sigma)\}\end{aligned}$$

where we also require that if $\mathrm{C}_{\text{ideal}}$ sets $i \in \{A, B\}$ as the first concluding party, then i concludes first in the simulated view $\mathrm{C}_{\text{ideal}}$ outputs.

Let us consider the session-key challenge given to the adversary in both the ideal and real models of Definition 2.2:

- In the ideal model of Definition 2.2, the trusted party gives the adversary $\mathrm{C}_{\text{ideal}}$ the session-key challenge K_β

$$K_\beta = \begin{cases} L_1 & \text{if } \beta = 1 \text{ or } L_1 = \perp, \\ U_n & \text{if } \beta = 0 \text{ and } L_1 \neq \perp. \end{cases}$$

- In the real model of Definition 2.2, the trusted party gives the adversary C the session-key challenge K_β

$$K_\beta = \begin{cases} M_1 & \text{if } \beta = 1 \text{ or } M_1 = \perp, \\ U_n & \text{if } \beta = 0 \text{ and } M_1 \neq \perp. \end{cases}$$

We fix the real adversary C (against security with respect to the environment) and define the real adversary C' (against security with respect to the session-key challenge) that, on auxiliary input (σ, τ) and upon receiving K from the first concluding party, simulates $Z(K, \tau)$ and C on its own. Hence $C'^{A(w),B(w)}(\sigma, \tau, K) \equiv \{K, \tau, C^{A(w),B(w),Z(K,\tau)}(\sigma)\}$. We now apply Definition 2.2 for the real adversary C'.

By security with respect to the session-key challenge, there exists an ideal adversary $\mathrm{C}'_{\mathrm{ideal}}$ such that

$$\text{IDEAL-SK}_{\mathrm{C}'_{\mathrm{ideal}}} \overset{O(\gamma)}{\equiv} \text{REAL-SK}_{C'}$$

$$\Rightarrow \{w, L_1, L_2, \mathrm{C}'_{\mathrm{ideal}}(\sigma, \tau, K_\beta), \beta\} \overset{O(\gamma)}{\equiv} \{w, M_1, M_2, C'^{A(w),B(w)}(\sigma, \tau, K_\beta), \beta\}.$$

Moreover, as β has only two possible values, we know that:

$$\{w, L_1, L_2, \mathrm{C}'_{\mathrm{ideal}}(\sigma, \tau, L_1)\} \overset{O(\gamma)}{\equiv} \{w, M_1, M_2, C'^{A(w),B(w)}(\sigma, \tau, M_1)\}, \tag{1}$$

$$\{w, L_1, L_2, \mathrm{C}'_{\mathrm{ideal}}(\sigma, \tau, K_0)\} \overset{O(\gamma)}{\equiv} \{w, M_1, M_2, C'^{A(w),B(w)}(\sigma, \tau, K_0)\}. \tag{2}$$

We will first prove that the real outputs of the honest parties are indistinguishable from ideal outputs, even when the environment Z is present. This is formalized by the following claim:

Claim 2.5. *For every Z, every τ and every σ,*

$$\{w, M_2, M_1, \tau, C^{A(w),B(w),Z(M_1,\tau)}(\sigma)\} \overset{O(\gamma)}{\equiv} \{w, L_2, L_1, \tau, C^{A(w),B(w),Z(L_1,\tau)}(\sigma)\},$$

where M_i is the output of the ith concluding party in the real execution $C^{A(w),B(w)}$ and L_1, L_2 are defined as follows:

- *if the first party in the real execution $C^{A(w),B(w)}$ accepts, then $L_1 = U_n$. Otherwise, $L_1 = \perp$.*
- *if the second party in the real execution $C^{A(w),B(w)}$ accepts and $L_1 \neq \perp$, then $L_2 = L_1$. If the second party accepts and $L_1 = \perp$, then $L_2 = U_n$. If the second party rejects, then $L_2 = \perp$.*

Proof of claim. By definition of C' and (1), we know that

$$\begin{aligned}\{w, M_2, M_1, \tau, C^{A(w),B(w),Z(M_1,\tau)}(\sigma)\} &\equiv \{w, M_2, C'^{A(w),B(w)}(\sigma, \tau, M_1)\} \\ &\overset{O(\gamma)}{\equiv} \{w, L_2, \mathrm{C}'_{\mathrm{ideal}}(\sigma, \tau, L_1)\}.\end{aligned}$$

Again, by definition of C', we have

$$\{w, L_2, L_1, \tau, C^{A(w),B(w),Z(L_1,\tau)}(\sigma)\} \equiv \{w, L_2, C'^{A(w),B(w)}(\sigma, \tau, L_1)\}.$$

Note that in both the real and ideal models, the string K_0 is distributed identically to L_1, hence by (2), we have

$$\{w, C'^{A(w),B(w)}(\sigma, \tau, L_1)\} \overset{O(\gamma)}{\equiv} \{w, \mathrm{C}'_{\text{ideal}}(\sigma, \tau, L_1)\}$$
$$\Rightarrow \{w, L_2, C'^{A(w),B(w)}(\sigma, \tau, L_1)\} \overset{O(\gamma)}{\equiv} \{w, L_2, \mathrm{C}'_{\text{ideal}}(\sigma, \tau, L_1)\}. \qquad \square$$

We will now prove that if the real outputs of the honest parties are replaced by ideal outputs, then the protocol leaks no information about the password w.

Claim 2.6. *For every Z, every τ and every σ,*

$$\{w, L_2, L_1, \tau, C^{A(w),B(w),Z(L_1,\tau)}(\sigma)\} \overset{O(\gamma)}{\equiv} \{\tilde{w}, L_2, L_1, \tau, C^{A(w),B(w),Z(L_1,\tau)}(\sigma)\},$$

where $\tilde{w} \overset{R}{\leftarrow} \mathcal{D}$ and L_1, L_2 are defined as follows:

- *if the first party in the real execution $C^{A(w),B(w)}$ accepts, then $L_1 = U_n$. Otherwise, $L_1 = \perp$.*
- *if the second party in the real execution $C^{A(w),B(w)}$ accepts and $L_1 \neq \perp$, then $L_2 = L_1$. If the second party accepts and $L_1 = \perp$, then $L_2 = U_n$. If the second party rejects, then $L_2 = \perp$.*

Proof of claim. We define the real adversary C'' (against security with respect to the session-key challenge) that, on auxiliary input (σ, τ) and upon receiving L_1 from the first concluding party, simulates $Z(L_1, \tau)$ and C on its own such that $C''^{A(w),B(w)}(\sigma, \tau, L_1) \equiv \{L_2, L_1, \tau, C^{A(w),B(w),Z(L_1,\tau)}\}$, where L_2 is computed according to the above rule. Since L_1 is distributed identically to K_0, by security with respect to the session-key challenge (for the case where $\beta = 0$) there exists $\mathrm{C}''_{\text{ideal}}$ such that

$$\{w, \mathrm{C}''_{\text{ideal}}(\sigma, \tau, L_1)\} \overset{O(\gamma)}{\equiv} \{w, C''^{A(w),B(w)}(\sigma, \tau, L_1)\}, \tag{3}$$

which in turn implies (by non-uniform indistinguishability or samplability of $\mathcal{D}$)

$$\{\tilde{w}, \mathrm{C}''_{\text{ideal}}(\sigma, \tau, L_1)\} \overset{O(\gamma)}{\equiv} \{\tilde{w}, C''^{A(w),B(w)}(\sigma, \tau, L_1)\}, \tag{4}$$

where $\tilde{w} \overset{\mathrm{R}}{\leftarrow} \mathcal{D}$. Note that in the ideal model, the adversary $\mathrm{C}''_{\text{ideal}}(\sigma, \tau, L_1)$ learns nothing about the password w since L_1 is independent of the password w. Hence we have

$$\{\tilde{w}, \mathrm{C}''_{\text{ideal}}(\sigma, \tau, L_1)\} \equiv \{w, \mathrm{C}''_{\text{ideal}}(\sigma, \tau, L_1)\}. \tag{5}$$

From (3), (4), (5) and transitivity of indistinguishability, we conclude that

$$\{w, C''^{A(w),B(w)}(\sigma, \tau, L_1)\} \overset{O(\gamma)}{\equiv} \{\tilde{w}, C''^{A(w),B(w)}(\sigma, \tau, L_1)\}. \qquad \square$$

Note that the distributions $\{\tilde{w}, L_2, L_1, \tau, C^{A(w),B(w),Z(L_1,\tau)}(\sigma)\}$ and $\{w, L_2, L_1, \tau, C^{A(\tilde{w}),B(\tilde{w}),Z(L_1,\tau)}(\sigma)\}$, where $\tilde{w} \overset{\mathrm{R}}{\leftarrow} \mathcal{D}$, are equivalent. Combining Claims 2.5 and 2.6, we obtain

$$\{w, M_1, M_2, \tau, C^{A(w),B(w),Z(M_1,\tau)}(\sigma)\} \overset{O(\gamma)}{\equiv} \{w, L_1, L_2, \tau, C^{A(\tilde{w}),B(\tilde{w}),Z(L_1,\tau)}(\sigma)\}.$$

We now describe the ideal adversary ${\mathrm{C}_{\mathrm{ideal}}}^{Z(L_1,\tau)}(\sigma)$ that simulates $C^{A(\tilde{w}),B(\tilde{w}),Z(L_1,\tau)}(\sigma)$:

1. $\mathrm{C}_{\mathrm{ideal}}$ generates a random password $\tilde{w} \stackrel{\mathrm{R}}{\leftarrow} \mathcal{D}$ and simulates the honest parties A and B in the interaction $(A(\tilde{w}), B(\tilde{w}))$.
2. Let $i \in \{A, B\}$ be the first party to conclude in the simulated execution. Party i outputs a decision bit dec_i and a key L_1 in the simulated execution. As soon as the first party in the simulated execution concludes, $\mathrm{C}_{\mathrm{ideal}}$ sends (i, dec_i) to the trusted party.
3. $\mathrm{C}_{\mathrm{ideal}}$ interacts with $Z(L_1, \tau)$ and continue the simulated execution.
4. Let $j \in \{A, B\}$ be the second party to conclude in the simulated execution. Party j outputs a decision bit dec_j and a key L_2 in the simulated execution. As soon as the second party in the simulated execution concludes, $\mathrm{C}_{\mathrm{ideal}}$ sends (j, dec_j) to the trusted party.

Hence, for any probabilistic polynomial-time C, there exists $\mathrm{C}_{\mathrm{ideal}}$ such that for every Z and every τ:

$$\{w, M_1, M_2, Z(M_1, \tau), C^{A(w),B(w),Z(M_1,\tau)}(\sigma)\}$$
$$\stackrel{O(\gamma)}{\equiv} \{w, L_1, L_2, Z(L_1, \tau), {\mathrm{C}_{\mathrm{ideal}}}^{Z(L_1,\tau)}(\sigma)\}.$$

Note that the $O(\cdot)$ in $O(\gamma)$ hides a constant factor lost in the proof. Specifically, the proof shows that if the real and ideal distributions in Definition 2.3 are $(1-\gamma)$-indistinguishable, then the real and ideal distributions in Definition 2.2 are $(1-3\gamma)$-indistinguishable. □

3. An Overview of the Protocol

Before presenting our protocol, we introduce the polynomial evaluation functionality, which is an important tool for the rest of the paper. In [21], it is observed that a secure protocol for polynomial evaluation immediately yields a protocol for session-key generation that is secure against *passive* adversaries. In [15], Goldreich and Lindell work from the intuition (from [6]) that by augmenting a secure protocol for polynomial evaluation with additional mechanisms, one can obtain a protocol for session-key generation that is secure against *active* adversaries. Our protocol also comes from this intuition, but the additional tools we are using are different.

3.1. *Secure Polynomial Evaluation*

In a secure polynomial evaluation, a party A knows a polynomial Q over some field $\mathbb{F}$ and a party B wishes to learn the value $Q(x)$ for some element $x \in \mathbb{F}$ such that A learns nothing about x and B learns nothing else about the polynomial Q but the value $Q(x)$. More specifically, for our problem, we will assume that $\mathbb{F} = \mathrm{GF}(2^n) \approx \{0,1\}^n$, Q is a non-constant linear polynomial over $\mathbb{F}$, and x is a string in $\{0,1\}^n$.

Definition 3.1 (Polynomial evaluation). The polynomial evaluation functionality is defined as:

Inputs The input of A is a non-constant linear polynomial Q over $\mathrm{GF}(2^n)$. The input of B is a value $x \in \mathrm{GF}(2^n)$.
Outputs B receives $Q(x)$. A receives nothing.

As observed in [21], a secure protocol for polynomial evaluation yields immediately a protocol for session-key generation that is secure against passive adversaries as follows: A chooses a random linear non-constant polynomial Q, and A and B engage in a secure polynomial evaluation protocol, where A inputs Q and B inputs w, so that B obtains $Q(w)$. Since A has both Q and w, A can also obtain $Q(w)$, and the session key is set to be $K = Q(w)$.

This protocol is secure against passive adversaries because the key K is a random string (since Q is a random polynomial), and it can be shown that an eavesdropper learns nothing about w or $Q(w)$ (due to the security of the polynomial evaluation).

However, the protocol is not secure against active adversaries. As illustrated in Fig. 1, an active adversary C can input a fixed polynomial Q_C in its interaction with B, say the identity polynomial id, and a fixed password w_C in its interaction with A. A outputs the session key $Q_A(w)$ and B outputs the session key $Q_C(w) = w$. With probability $1 - 2^{-n}$, the two session keys are different, whereas the definition of security requires them to be equal with probability $1 - O(\gamma)$.

3.2. *Motivation for Our Protocol*

The main deficiency of the secure polynomial evaluation protocol against active adversaries is that it does not guarantee that A and B output the same random session key. Somehow, the parties have to check that they computed the same random session key before starting to use it. It can be shown that A's session key $K_A = Q_A(w)$ is pseudorandom to the adversary, so A can start using it without leaking information. However, B cannot use its key $K_B = Q_C(w)$ because it might belong to a set of polynomial size (for example, if $Q_C = id$, then $Q_C(w) \in \mathcal{D}$ where the dictionary is by definition a small set). Hence Goldreich and Lindell added a validation phase in which A sends a message to B so that B can check if it computed the same session key, say A sends $f^{2n}(K_A)$ where f is a one-way permutation. Since f^{2n} is a 1–1 map, this uniquely defines K_A (the session-key used now consists of hardcore bits of $f^i(K_A)$, for $i = 0, \ldots, n-1$): B will compute $f^{2n}(K_B)$ and compare it with the value it received.

But it is still not clear that this candidate protocol is secure. Recall that the security of the polynomial evaluation protocol applies only in the stand-alone setting and guarantees nothing in the concurrent setting. In particular, it might be that C inputs a

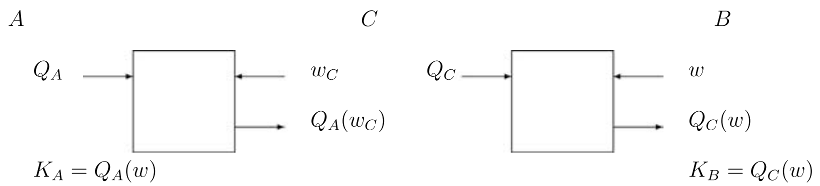

Fig. 1. Protocol that is insecure against active adversaries

polynomial Q_C in the polynomial evaluation between C and B such that the polynomials Q_A and Q_C are related in some manner, say for any $w \in \mathcal{D}$, it is easy to compute the correct validation message $f^{2n}(Q_C(w))$ given the value of $f^{2n}(Q_A(w))$; yet B's key does not equal A's key.

To prevent this from happening, Goldreich and Lindell force the polynomial Q input in the polynomial evaluation phase to be consistent with the message sent in the validation phase (which is supposedly $f^{2n}(Q(w))$). The parties have to commit to their inputs at the beginning and then prove in a zero-knowledge manner that the messages sent in the validation phase are consistent with these commitments. Because of the person-in-the-middle attack and the concurrency issues mentioned earlier, Goldreich and Lindell cannot use standard commitment schemes and standard zero-knowledge proofs but rather they use non-malleable commitments and the specific zero-knowledge proof of Richardson and Kilian.

Our approach is to allow C to input a polynomial Q_C related to Q_A, but to prevent C from being able to compute a correct validation message with respect to B's session-key, even given A's validation message. Suppose that the parties have access to a family of pairwise-independent hash functions $\mathcal{H}$. In the validation phase, we require A to send $h(f^{2n}(K_A)) = h(f^{2n}(Q_A(w)))$ for some function $h \xleftarrow{R} \mathcal{H}$. Then, even if $K_A = Q_A(w)$ and $K_B = Q_C(w)$ are related (but distinct), the values $h(f^{2n}(K_A))$ and $h(f^{2n}(K_B))$ will be independent and C cannot do much better than randomly guess the value of $h(f^{2n}(K_B))$.

One difficulty arises at this point: the parties have to agree on a common random hash function $h \xleftarrow{R} \mathcal{H}$. But the honest parties A and B only share the randomness coming from the password w so this password w has to be enough to agree on a random hash function. To make this possible, we assume that the password is of the form (w, w') where w and w' are chosen independently of one another: w is chosen at random from an arbitrary dictionary $\mathcal{D} \subseteq \{0,1\}^n$ and w' is uniformly distributed in $\mathcal{D}' = \{0,1\}^{d'}$. (For example, these can be obtained by splitting a single random password from $\{0,1\}^{d''}$ into two parts.) The first part of the password, w, will be used in the polynomial evaluation protocol whereas the second part of the password, w', will be used as the index of a hash function. Indeed, if we assume that $\mathcal{D}' = \{0,1\}^{d'}$, there exists a family of almost pairwise-independent hash functions $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$, where each hash function is indexed by a password $w' \in \mathcal{D}'$ and $m = \Omega(d')$ (see proof in Appendix B).

We formalize these ideas in the protocol described below.

3.3. *Tools Used in Our Protocol*

As in [15], we will need a secure protocol for an augmented version of polynomial evaluation. We refer the reader to Appendix A for more details on secure two-party computation.

Definition 3.2 (Augmented polynomial evaluation). The augmented polynomial evaluation functionality is defined as:

Initial phase A sends a commitment $c_A = \text{Commit}(Q_A, r_A)$ to a linear non-constant polynomial Q_A for a randomly chosen r_A. B receives a commitment c_B. We assume that the commitment scheme used is perfectly binding and computationally hiding.

Inputs The input of A is a linear non-constant polynomial Q_A, a commitment c_A to Q_A and a corresponding decommitment r_A. The input of B is a value $x \in \mathrm{GF}(2^n)$ and a commitment c_B.

Outputs

- In the case of correct inputs, i.e., $c_A = c_B$ and $c_A = \mathrm{Commit}(Q_A, r_A)$, B receives $Q_A(x)$ and A receives nothing.
- In the case of incorrect inputs, i.e., $c_A \neq c_B$ or $c_A \neq \mathrm{Commit}(Q_A, r_A)$, B receives a special failure symbol $\perp$ and A receives nothing.

The other cryptographic tools we will need are:

Commitment scheme: Let Commit be a perfectly binding, computationally hiding string commitment.

Seed-committed pseudorandom generator: similarly to [15], we will use the seed-committed pseudorandom generator $G(s) = (b(s)b(f(s))\dots b(f^{n+\ell-1}(s))f^{n+\ell}(s))$ where f is a one-way permutation with hardcore bit b.

One-time MAC with pseudorandomness property: Let MAC be a message authentication code for message space $\{0,1\}^{p(n)}$ (for a polynomial $p(n)$ to be specified later) using keys of length $\ell = \ell(n)$ that is secure against one query attack, i.e., a probabilistic polynomial-time A that queries the tagging algorithm MAC_K on *at most one* message of its choice cannot produce a valid forgery on a different message. Additionally, we will require the following pseudorandomness property:

- Let K be a uniform key of length ℓ.
- The adversary queries the tagging algorithm MAC_K on the message m of its choice.
- The adversary selects $m' \neq m$. We require that the value $\mathrm{MAC}_K(m')$ be pseudorandom with respect to the adversary's view.

Two examples of such a MAC are:

- $\mathrm{MAC}_s(m) = f_s(m)$ where $\{f_s\}_{s\in\{0,1\}^\ell}$ is a pseudorandom function family.
- $\mathrm{MAC}_{a,b}(m) = am + b$ where $\ell(n) = 2p(n)$ and $a, b \in \mathrm{GF}(2^{\ell/2})$.

Almost pairwise-independent hash functions: The family of functions $\mathcal{H} = \{h_{w'} : \{0,1\}^n \to \{0,1\}^m\}_{w'\in\{0,1\}^{d'}}$ is said to be *almost pairwise-independent with parameter* μ if:

1. (Uniformity) $\forall x \in \{0,1\}^n$, $h_{w'}(x)$ is uniform over $\{0,1\}^m$.
2. (Pairwise independence) $\forall x_1 \neq x_2 \in \{0,1\}^n$, $\forall y_1, y_2 \in \{0,1\}^m$,

$$\Pr_{w'\in\{0,1\}^{d'}}[h_{w'}(x_2) = y_2 | h_{w'}(x_1) = y_1] \leq \mu.$$

We also require that for a fixed $w' \in \{0,1\}^{d'}$ the function $h_{w'}$ be regular i.e., it is 2^{n-m} to 1. In other words, $h_{w'}(U_n) \equiv U_m$.

Lemma 3.3 (Appendix B). *For $\mathcal{D}' = \{0,1\}^{d'}$ there exists a family of almost pairwise-independent hash functions $\mathcal{H} = \{h_{w'} : \{0,1\}^n \to \{0,1\}^m\}_{w'\in\mathcal{D}'}$ with parameter $\mu = O(\frac{n}{|\mathcal{D}'|^{1/3}\log|\mathcal{D}'|})$.*

3.4. *Description of Our Protocol*

The formal description of the protocol follows (see Fig. 2 for an overview).

Protocol 3.4.

1. **Inputs:** The parties A and B have a joint password (w, w'), where w is chosen at random from an arbitrary dictionary $\mathcal{D} \subseteq \{0,1\}^n$ and w' is uniformly distributed in $\mathcal{D}' = \{0,1\}^{d'}$. (Throughout, we will view $\mathcal{D}'$ as a subset of $\{0,1\}^n$ after appropriate padding for consistency with Sect. 2 where the security parameter is defined to be the length of the password). w and w' are chosen independently.
2. **Commitment:** A chooses a random non-constant linear polynomial Q_A over $\mathrm{GF}(2^n)$ and random coins r_A and sends $c_A = \mathrm{Commit}(Q_A, r_A)$. B receives some commitment c_B.
3. **Augmented polynomial evaluation**
   (a) A and B engage in a polynomial evaluation protocol: A inputs the polynomial Q_A, the commitment c_A and the random coins r_A it used for the commitment; B inputs the commitment c_B it received and the password w viewed as an element of $\mathrm{GF}(2^n)$.
   (b) The output of B is denoted Π_B, which is supposed to be equal to $Q_A(w)$.
   (c) A internally computes $\Pi_A = Q_A(w)$.
4. **Validation**
   (a) A sends the string $y_A = h_{w'}(f^{n+\ell}(\Pi_A))$, where f is a one-way permutation and $\mathcal{H} = \{h_{w'}\}_{w' \in \{0,1\}^{d'}}$ is a family of almost pairwise-independent hash functions.
   (b) Let t_A be the session transcript so far as seen by A. A computes $k_1(\Pi_A) = b(\Pi_A)\ldots b(f^{\ell-1}(\Pi_A))$ and sends the string $z_A = \mathrm{MAC}_{k_1(\Pi_A)}(t_A)$.

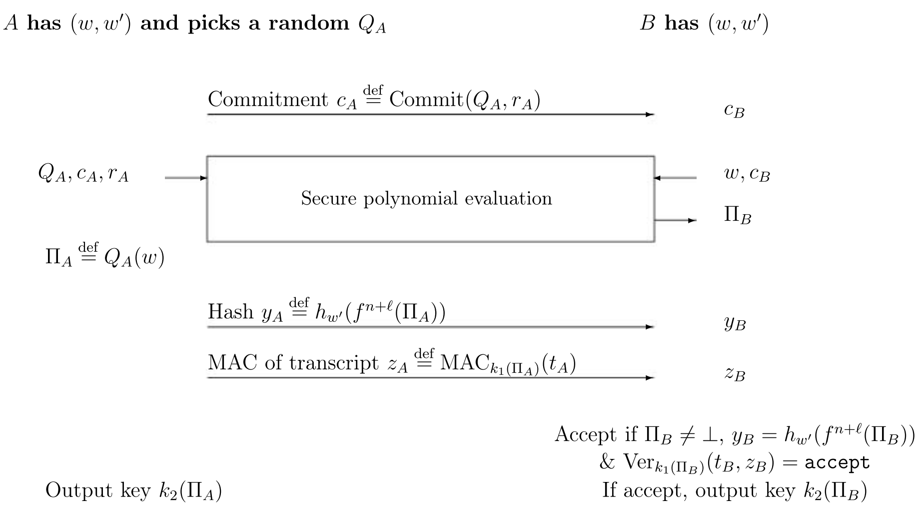

Fig. 2. Overview of our protocol

5. **Decision**
   (a) A always accepts and outputs $k_2(\Pi_A) = b(f^{\ell}(\Pi_A)) \dots b(f^{\ell+n-1}(\Pi_A))$
   (b) B accepts (this event is denoted by $\mathrm{dec}_B = \texttt{accept}$) if the strings y_B and z_B it received satisfy the following conditions:

   - $y_B = h_{w'}(f^{n+\ell}(\Pi_B))$
   - $\mathrm{Ver}_{k_1(\Pi_B)}(t_B, z_B) = \texttt{accept}$, where t_B is the session transcript so far as seen by B and $k_1(\Pi_B)$ is defined analogously to $k_1(\Pi_A)$.

   If $\Pi_B = \perp$, then B will immediately reject.
   If B accepts, it outputs $k_2(\Pi_B) = b(f^{\ell}(\Pi_B)) \dots b(f^{\ell+n-1}(\Pi_B))$.

4. Main Security Theorems

We begin by stating our protocol's security against passive adversaries.

Theorem 4.1. *Protocol 3.4 is secure for the dictionary $\mathcal{D} \times \mathcal{D}' = \mathcal{D} \times \{0, 1\}^{d'}$ against passive adversaries. More formally, for every passive probabilistic polynomial-time real adversary C, there exists an ideal adversary $\mathrm{C_{ideal}}$ that always sends $(\mathrm{dec}_C^A, \mathrm{dec}_C^B) = (1, 1)$ to the trusted party such that for every auxiliary input $\sigma \in \{0, 1\}^{\mathrm{poly}(n)}$:*

$$\{\mathrm{IDEAL}_{\mathrm{C_{ideal}}}(\mathcal{D} \times \mathcal{D}', \sigma)\} \overset{\mathrm{comp}}{\equiv} \{\mathrm{REAL}_C(\mathcal{D} \times \mathcal{D}', \sigma)\}.$$

The proof of Theorem 4.1 is given in Sect. 5.

Next we state the basic security theorem against active adversaries, in the plain model with a dictionary of the form $\mathcal{D} \times \{0, 1\}^{d'}$.

Theorem 4.2. *Protocol 3.4 is $(1 - \gamma)$-secure with respect to the environment (equivalently, with respect to the session-key challenge) for the dictionary $\mathcal{D} \times \mathcal{D}' = \mathcal{D} \times \{0, 1\}^{d'}$, for $\gamma = \max\{\frac{1}{|\mathcal{D}|}, (\frac{\mathrm{poly}(n)}{|\mathcal{D}'|})^{\Omega(1)}\}$. More precisely, $\gamma = \max\{\frac{1}{|\mathcal{D}|}, O(\frac{n}{|\mathcal{D}'|^{1/3}})\}$.*

In Sect. 9 we show how the shared dictionary of the form $\mathcal{D} \times \{0, 1\}^d$ required in Theorem 4.2 can be realized from several other types of dictionaries $\mathcal{D}''$, achieving security bounds of the form $(\mathrm{poly}(n)/|\mathcal{D}''|)^{\Omega(1)}$ in all cases.

4.1. *Overview of the Proof of Theorem 4.2*

Notations

- Without loss of generality, we will assume that the real adversary's output equals its view of the execution (since the output is efficiently computable from the view). We will also often omit the auxiliary input σ of the adversary.
- Recall that we denote by $C^{A(w,w'),B(w,w')}$ an execution of C when it communicates with A and B, with common input (w, w'). We denote by $C^{A(Q_A,w,w'),B(w,w')}$ the execution of C with A and B where Q_A specifies the random non-constant linear polynomial to be used by A.

- 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. If C was reliable in the given execution, we denote this event by $\mathrm{reliable}_C = \mathtt{true}$; otherwise, we write $\mathrm{reliable}_C = \mathtt{false}$.

Structure of the proof We will mostly focus on the basic GL definition (Definition 2.1), but after each step we will describe the modifications needed to handle the session-key challenge of Definition 2.2. (This is easier than directly proving security for an arbitrary environment as in Definition 2.3 because it only requires taking into account a specific environment Z corresponding to the session-key challenge.)

Similarly to [15], the proof of Theorem 4.2 is in four steps:

1. **Key-Match Property:** In Sects. 6 and 7, we show that if $\Pi_A \neq \Pi_B$, then B will reject with probability $1 - O(\gamma)$.
2. **Simulation of the** (C, B) **interaction:** In Sect. 8.2, we show that if the Key-Match Property holds, then the interaction (C, B) can be simulated by an adversary C' interacting only with A, even if the interaction (A, C) is concurrent.
3. **Simulation of the** (A, C') **interaction:** In Sect. 8.1, we show that the interaction (A, C') as a stand-alone can be simulated.
4. In Sect. 8.3, we combine the above steps and obtain a proof of security against active adversaries. The real adversary's view of the concurrent interactions (A, C) and (C, B) can be simulated by a probabilistic polynomial-time C'' that is non-interactive and can therefore be simulated by an ideal adversary with no input.

As in [15], the main part of the proof of Theorem 4.2 is the *Key-Match Property*. Once the Key-Match Property is established, we can easily adapt the proofs in [15] to our specific protocol to build an ideal adversary that simulates the real adversary's view.

Theorem 4.3 (Key-Match Property). *For every probabilistic polynomial-time real adversary C and all sufficiently large values of n*

$$\Pr[\mathrm{dec}_B = \mathtt{accept} \wedge \Pi_A \neq \Pi_B] < 2\mu + \epsilon + \mathrm{neg}(n)$$

where $\epsilon = \frac{1}{|\mathcal{D}|}$ and $\mu = O(\frac{n}{|\mathcal{D}'|^{1/3} \log |\mathcal{D}'|})$.

The main part of our proof that is new (and simpler than [15]) is the Key-Match Property. As noted in the introduction, the adversary C has total control over the scheduling of the two interactions (A, C) and (C, B). Hence the Key-Match Property will be proved for every possible scheduling case, including those for which these interactions are concurrent. Nevertheless, the Key-Match Property will be established by tools of secure two-party computation, which *a priori* only guarantee security in the stand-alone setting.

For each scheduling, we want to bound from above the probability $[\mathrm{dec}_B = \mathtt{accept} \wedge \Pi_A \neq \Pi_B]$. Recall that B accepts iff two conditions are satisfied: the string y_B received must equal $h_{w'}(f^{n+\ell}(\Pi_B))$ and the MAC z_B received must be a valid MAC, i.e., $\mathrm{Ver}_{k_1(\Pi_B)}(t_B, z_B) = \mathtt{accept}$. Hence, to obtain an upper bound we can omit the verification of the MAC by B and only consider the probability that C succeeds in

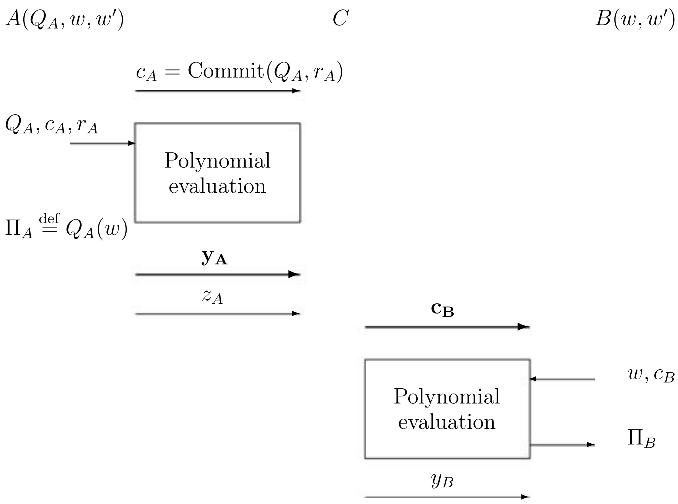

Fig. 3. First scheduling

sending the value $h_{w'}(f^{n+\ell}(\Pi_B))$ when $\Pi_A \neq \Pi_B$. (As in [15], the MAC is only used to reduce the simulation of active adversaries to the simulation of passive adversaries plus the key-match property.) For convenience, we will decompose the adversary into two algorithms.

- The first algorithm is denoted by C. C is the channel through which A and B communicate. For a given execution, we denote by $C^{A(Q_A,w,w'),B(w,w')}$ the view of C when it communicates with A and B with respective inputs (Q_A, w, w') and (w, w') *until just before C sends a string y_B to B*.
- The second algorithm is denoted by $\mathrm{C_{hash}}$. $\mathrm{C_{hash}}$ takes as an input the above view $C^{A(Q_A,w,w'),B(w,w')}$ and tries to compute the hash value $h_{w'}(f^{n+\ell}(\Pi_B))$.

Hence to establish the Key-Match Property, for each scheduling, we will bound from above the probability

$$\Pr[\mathrm{C_{hash}}(C^{A(Q_A,w,w'),B(w,w')}) = h_{w'}(f^{n+\ell}(\Pi_B)) \wedge \Pi_A \neq \Pi_B].$$

Note that since B always rejects if $\Pi_B = \perp$, we can adopt the convention that

$$\Pr[\mathrm{C_{hash}}(C^{A(Q_A,w,w'),B(w,w')}) = h_{w'}(f^{n+\ell}(\Pi_B)) \wedge \Pi_B = \perp] = 0.$$

We consider two scheduling cases (see Figs. 3 and 4):

Scheduling 1: C sends the commitment c_B to B *after* A sends the hash value y_A.
The intuition for this case is that we have two sequential executions (A, C) and (C, B). Using the security of the polynomial evaluation (A, C), we show that even if C receives y_A, the hash index w' is $(1-\epsilon)$-pseudorandom with respect to the adversary's view. Hence, by the uniformity property of the hash functions, C cannot do much better than randomly guess the value of $h_{w'}(f^{n+\ell}(\Pi_B))$. The full proof for this scheduling is in Sect. 6.

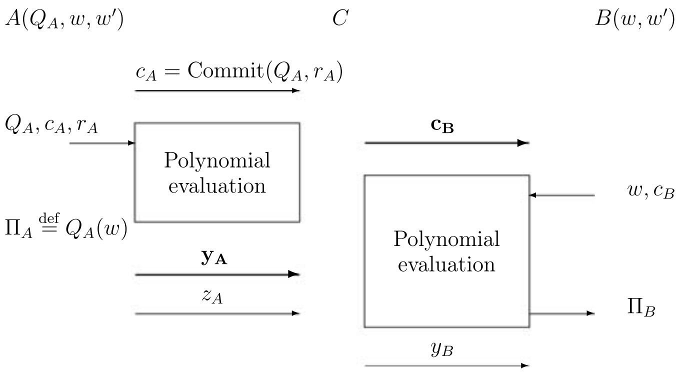

Fig. 4. Second scheduling

Scheduling 2: C sends the commitment c_B to B *before* A sends the hash value y_A.
The almost pairwise independence property means that for fixed values $x_1 \neq x_2 \in \{0,1\}^n$, if the index w' is chosen at random and independently of x_1 and x_2, then being given the value $h_{w'}(x_1)$ does not help one guess the value $h_{w'}(x_2)$. Before y_A is sent, the hash index w' is random (since it has not been used by A or B). Thus, if we show that the values Π_A and Π_B are determined before y_A is sent, then w' is independent of $x_1 = f^{n+\ell}(\Pi_A)$ and $x_2 = f^{n+\ell}(\Pi_B)$ and the adversary cannot guess $h_{w'}(x_2)$ even given $y_A = h_w(x_1)$. Π_A is determined before y_A is sent by the definition of the protocol. Π_B is determined because the commitment c_B is a perfectly binding commitment to some value Q_C, and thus the security of augmented polynomial evaluation implies that Π_B equals $Q_C(w)$ (or $\perp$) except with negligible probability. The full proof for this scheduling is in Sect. 7.

5. Proof of Security against Passive Adversaries

Theorem 5.1. *Protocol 3.4 is secure for the dictionary $\mathcal{D} \times \mathcal{D}' = \mathcal{D} \times \{0,1\}^{d'}$ against passive adversaries. More formally, for every passive probabilistic polynomial-time real adversary C, there exists an ideal adversary $\mathrm{C}_{\mathrm{ideal}}$ that always sends $(\mathrm{dec}_C^A, \mathrm{dec}_C^B) = (1, 1)$ to the trusted party such that for every auxiliary input $\sigma \in \{0,1\}^{\mathrm{poly}(n)}$:*

$$\{\mathrm{IDEAL}_{\mathrm{C}_{\mathrm{ideal}}}(\mathcal{D} \times \mathcal{D}', \sigma)\} \stackrel{\text{comp}}{\equiv} \{\mathrm{REAL}_C(\mathcal{D} \times \mathcal{D}', \sigma)\}.$$

Recall that a passive adversary just eavesdrops on the interaction between the honest parties so in this case, the parties A and B output the same session-key ($\mathrm{output}(A) = \mathrm{output}(B)$) and both accept. In the ideal model, the session-key $\mathrm{K}_{\mathrm{ideal}}$ is distributed according to U_n.

Thus, to prove Theorem 5.1, it suffices to prove the following proposition:

Proposition 5.2. *For every passive probabilistic polynomial-time real adversary C, there exists an ideal adversary* $\mathrm{C}_{\mathrm{ideal}}$ *such that*

$$\{w, w', \mathrm{output}(A), \mathrm{output}(C^{A(w,w'),B(w,w')})\} \overset{\mathrm{comp}}{\equiv} \{w, w', U_n, \mathrm{output}(\mathrm{C}_{\mathrm{ideal}})\}.$$

Proof. The view of the real adversary consists of a transcript of the execution of the protocol by A and B. We can think of this transcript as the concatenation of:

- The commitment to Q_A and the transcript of the augmented polynomial evaluation. We denote these by $T(Q_A, w)$.
- The hash value $y_A \overset{\mathrm{def}}{=} h_{w'}(f^{n+\ell}(\Pi_A))$ where $\Pi_A \overset{\mathrm{def}}{=} Q_A(w)$.
- The MAC-key $k_1(\Pi_A)$ (it suffices to include the MAC-key rather than the MAC itself, since the latter is easily computable from the MAC-key and the transcript so far).

Claim 5.3.

$$\{w, Q_A, T(Q_A, w)\} \overset{\mathrm{comp}}{\equiv} \{w, Q_A, T(\tilde{Q}_A, \tilde{w})\},$$

where Q_A and $\tilde{Q}_A$ are random non-constant linear polynomials and $w, \tilde{w}$ are taken uniformly at random (*and independently*) *from $\mathcal{D}$.*

Proof of sketch. The claim follows from the security of the augmented polynomial evaluation.

The commitment scheme we consider is computationally hiding hence a commitment to Q_A is indistinguishable from a commitment to $\tilde{Q}_A$. Note that non-constant linear polynomials are connected i.e., for every Q_A and $\tilde{Q}_A$, there exists $\hat{Q}_A$ and values x_1 and x_2 such that $Q_A(x_1) = \hat{Q}_A(x_1)$ and $\tilde{Q}_A(x_2) = \hat{Q}_A(x_2)$. Combining this connectedness property with the security of the augmented polynomial evaluation, we know (see Claim 5.2 in [15]) that $\forall w, Q_A, \tilde{w}, \tilde{Q}_A$,

$$T(Q_A, w) \overset{\mathrm{comp}}{\equiv} T(\tilde{Q}_A, \tilde{w}).$$

This implies that $\{w, Q_A, T(Q_A, w)\} \overset{\mathrm{comp}}{\equiv} \{w, Q_A, T(\tilde{Q}_A, \tilde{w})\}$. □

Claim 5.3 implies that

$$\begin{aligned}\{w, Q_A(w), T(Q_A, w)\} &\overset{\mathrm{comp}}{\equiv} \{w, Q_A(w), T(\tilde{Q}_A, \tilde{w})\}\\ &\equiv \{w, U_n, T(\tilde{Q}_A, \tilde{w})\}\end{aligned} \tag{6}$$

where (6) comes from the fact that for a random Q_A, $\Pi_A = Q_A(w)$ is uniformly distributed in $\{0,1\}^n$ and Q_A is independent of $T(\tilde{Q}_A, \tilde{w})$.

Note that w' is independent from the variables in (6) hence we have:

$$\{w, w', Q_A(w), T(Q_A, w)\} \overset{\mathrm{comp}}{\equiv} \{w, w', U_n, T(\tilde{Q}_A, \tilde{w})\}.$$

We can then apply the deterministic polytime function $G(\cdot) = (f^{n+\ell}(\cdot), k_1(\cdot), k_2(\cdot))$ to the third component of each distribution to obtain:

$$\{w, w', k_2(\Pi_A), f^{n+\ell}(\Pi_A), k_1(\Pi_A), T(Q_A, w)\}$$
$$\stackrel{\text{comp}}{\equiv} \{w, w', k_2(U_n), f^{n+\ell}(U_n), k_1(U_n), T(\tilde{Q}_A, \tilde{w})\}.$$

Since $G(s) = (f^{n+\ell}(s), k_1(s), k_2(s))$ is a pseudorandom generator, we have:

$$\{w, w', k_2(\Pi_A), f^{n+\ell}(\Pi_A), k_1(\Pi_A), T(Q_A, w)\} \stackrel{\text{comp}}{\equiv} \{w, w', U_n^1, U_n^2, U_\ell, T(\tilde{Q}_A, \tilde{w})\}$$
$$\Rightarrow \{w, w', k_2(\Pi_A), h_{w'}(f^{n+\ell}(\Pi_A)), k_1(\Pi_A), T(Q_A, w)\}$$
$$\stackrel{\text{comp}}{\equiv} \{w, w', U_n^1, h_{w'}(U_n^2), U_\ell, T(\tilde{Q}_A, \tilde{w})\}.$$

For a fixed $w' \in \mathcal{D}'$, $h_{w'}$ is a regular map, so we obtain

$$\{w, w', k_2(\Pi_A), h_{w'}(f^{n+\ell}(\Pi_A)), k_1(\Pi_A), T(Q_A, w)\}$$
$$\stackrel{\text{comp}}{\equiv} \{w, w', U_n^1, U_m, U_\ell, T(\tilde{Q}_A, \tilde{w})\}.$$

The ideal adversary $\mathrm{C}_{\text{ideal}}$ will do the following:

1. Generate a random password $\tilde{w} \in \mathcal{D}$ and a random non-constant linear polynomial $\tilde{Q}_A$.
2. Simulate the honest parties in the augmented polynomial evaluation to produce the transcript $T(\tilde{Q}_A, \tilde{w})$.
3. Generate random strings U_m and U_ℓ.
4. Output $(U_m, U_\ell, T(\tilde{Q}_A, \tilde{w}))$. □

6. Key-Match Property for the First Scheduling

Scheduling 1 is defined as "C sends the commitment c_B to B *after* A sends y_A". Without loss of generality we can assume that C sends the commitment c_B to B after A sends z_A (since obtaining z_A can only help C). As outlined in Sect. 4, we want to upperbound the probability that B accepts and $\Pi_A \neq \Pi_B$ for this scheduling.

The intuition for the Key-Match Property for Scheduling 1 is that we have two sequential executions (A, C) and (C, B). Using the security of the polynomial evaluation (A, C), we show that even if C receives y_A, the hash index w' is $(1-\epsilon)$-pseudorandom with respect to the adversary's view. Hence, by the uniformity property of the hash functions, C cannot do much better than randomly guess the value of $h_{w'}(f^{n+\ell}(\Pi_B))$, and thus B will reject this hash value with high probability. Note that this argument only uses the uniformity of the hash functions (rather than their pairwise independence) and does not explicitly rely on the condition $\Pi_A \neq \Pi_B$. (Nevertheless, the analysis implies that $\Pi_A \neq \Pi_B$ with high probability in this scheduling; otherwise, the adversary could compute the hash value by just copying.) In the second scheduling, we will directly exploit both the pairwise independence and the condition $\Pi_A \neq \Pi_B$.

Proposition 6.1. *For every probabilistic polynomial-time real adversary C and all sufficiently large values of n*

$$\Pr[\mathrm{dec}_B = \mathtt{accept} \wedge \Pi_A \neq \Pi_B \wedge \mathrm{Sch1}] < \epsilon + \mu + \mathrm{neg}(n)$$

where $\epsilon = \frac{1}{\mathcal{D}}$ *and* $\mu = O(\frac{n}{|\mathcal{D}'|^{1/3}\log|\mathcal{D}'|})$. Sch1 *denotes the event that the execution follows the first scheduling.*

Proof. From the discussion in Sect. 4, recall that:

$$\begin{aligned}&\Pr[\mathrm{dec}_B = \mathtt{accept} \wedge \Pi_A \neq \Pi_B \wedge \mathrm{Sch1}]\\ &\quad\le \Pr\big[\mathrm{C_{hash}}(C^{A(Q_A,w,w'),B(w,w')}) = h_{w'}(f^{n+\ell}(\Pi_B)) \wedge \Pi_A \neq \Pi_B \wedge \mathrm{Sch1}\big],\end{aligned}$$

where we denote by $C^{A(Q_A,w,w'),B(w,w')}$ the view of the channel C when it communicates with A and B with respective inputs (Q_A, w, w') and (w, w') *until just before C sends a string y_B to B* and $\mathrm{C_{hash}}$ is a probabilistic polynomial-algorithm that takes as an input the above view $C^{A(Q_A,w,w'),B(w,w')}$ and tries to compute the hash value $h_{w'}(f^{n+\ell}(\Pi_B))$.

We decompose the adversary into two algorithms:

- C_1 refers to the adversary until just before the commitment c_B is sent. Let (τ, y_A, z_A) denote the view of the adversary C_1 when interacting with $A(Q_A, w, w')$.
- C_2 refers to the adversary once the (A, C) interaction is over, i.e., C_2 will be given as inputs (τ, y_A, z_A). Since C_2 and B are executing the secure (augmented) polynomial evaluation in the stand-alone setting, we know that there exists an ideal adversary $\mathrm{C_{2,ideal}}$ such that for every τ, y_A, z_A,

  $$\{\Pi_{B,\mathrm{ideal}}, {\mathrm{C_{2,ideal}}}^{B(w,c_B)}(\tau, y_A, z_A)\} \stackrel{\mathrm{comp}}{\equiv} \{\Pi_B, C_2^{B(w,c_B)}(\tau, y_A, z_A)\},$$

  where

  $$\Pi_{B,\mathrm{ideal}} \stackrel{\mathrm{def}}{=} \mathrm{output}(B^{\mathrm{C_{2,ideal}}(\tau,y_A,z_A)}(w, c_B))$$

  and

  $$\Pi_B \stackrel{\mathrm{def}}{=} \mathrm{output}(B^{C_2(\tau,y_A,z_A)}(w, c_B)).$$

Let use this ideal adversary $\mathrm{C_{2,ideal}}$ in the above expression:

$$\begin{aligned}&\Pr\big[\mathrm{C_{hash}}(C^{A(Q_A,w,w'),B(w,w')}) = h_{w'}(f^{n+\ell}(\Pi_B)) \wedge \Pi_A \neq \Pi_B \wedge \mathrm{Sch1}\big]\\ &\quad\le \Pr\big[\mathrm{C_{hash}}(C_2^{B(w,c_B)}(\tau, y_A, z_A)) = h_{w'}(f^{n+\ell}(\Pi_B)) \wedge \mathrm{Sch1}\big]\\ &\quad\le \Pr\big[\mathrm{C_{hash}}({\mathrm{C_{2,ideal}}}^{B(w,c_B)}(\tau, y_A, z_A)) = h_{w'}(f^{n+\ell}(\Pi_{B,\mathrm{ideal}})) \wedge \mathrm{Sch1}\big] + \mathrm{neg}(n)\\ &\quad\le \Pr\big[\mathrm{C'_{hash}}(\tau, h_{w'}(f^{n+\ell}(\Pi_A)), k_1(\Pi_A)) = h_{w'}(f^{n+\ell}(Q_C(w)))\big] + \mathrm{neg}(n)\end{aligned} \tag{7}$$

where $\mathrm{C'_{hash}}$ simulates $\mathrm{C_{2,ideal}}$'s view of the ideal polynomial evaluation with B and Q_C is $\mathrm{C_{2,ideal}}$'s input (wlog we will assume that we are in the correct input case in the augmented polynomial evaluation (C, B) since by convention we define $\mathrm{C}_{\text{hash}}(\sigma) \neq h_{w'}(f^{n+\ell}(\Pi_B))$ if $\Pi_B = \perp$). Equation (7) comes from the fact that the security of the augmented polynomial evaluation (C, B) holds even for fixed inputs (w, c_B, τ, y_A, z_A) and with advice string (w', Π_A) given to the distinguisher.

We will now prove that the hash index w' is $(1-\epsilon)$-pseudorandom with respect to the inputs given to $\mathrm{C}'_{\text{hash}}$ (as well as w). This will imply that the value $h_{w'}(f^{n+\ell}(Q_C(w)))$ is $(1-\epsilon)$-indistinguishable from uniform. Thus $h_{w'}(f^{n+\ell}(Q_C(w)))$ will be predicted by $\mathrm{C}'_{\text{hash}}$ with probability at most $\epsilon + 2^{-m}$ and this will establish the key-match property for this scheduling.

To establish that the hash index w' is $(1-\epsilon)$-pseudorandom with respect to the inputs given to $\mathrm{C}'_{\text{hash}}$, we will show that $(Q_A(w))$ is $(1-\epsilon)$-pseudorandom to the adversary hence the hash $h_{w'}(f^{n+\ell}(\Pi_A))$ is a uniform string to the adversary that does not convey information about the hash index w'. This is formalized by the following lemma.

Lemma 6.2. *For every probabilistic polynomial-time adversary C' interacting with $A(Q_A)$ who halts after the augmented polynomial evaluation,* $\{w, Q_A(w), C'^{A(Q_A)}\} \stackrel{\epsilon}{\equiv} \{w, U_n, C'^{A(Q_A)}\}$.

Proof. C' receives a commitment $c_A = \mathrm{Commit}(Q_A, r_A)$ from A before executing the secure protocol for augmented polynomial evaluation. By security of the augmented polynomial evaluation, we know that there exists an ideal adversary $\mathrm{C}'_{\text{ideal}}$ such that for every Q_A, c_A, r_A, we have $C'^{A(Q_A,c_A,r_A)} \stackrel{\text{comp}}{\equiv} \mathrm{C}'_{\text{ideal}}{}^{A(Q_A,c_A,r_A)}(c_A)$. Without loss of generality, we will assume that we are in the correct input case so that $\mathrm{C}'_{\text{ideal}}$ always receives $Q_A(w_C)$ for some input $w_C = \mathrm{C}'_{\text{ideal}}(c_A)$. Hence for every w, Q_A, c_A, r_A, we have $C'^{A(Q_A,c_A,r_A)} \stackrel{\text{comp}}{\equiv} \mathrm{C}'_{\text{ideal}}(c_A, w_C, Q_A(w_C))$.

We want to show that

$$\{w, Q_A(w), \mathrm{Commit}(Q_A), w_C, Q_A(w_C)\} \stackrel{\epsilon}{\equiv} \{w, U_n, \mathrm{Commit}(Q_A), w_C, Q_A(w_C)\},$$

where $w_C = \mathrm{C}'_{\text{ideal}}(\mathrm{Commit}(Q_A))$.

- By the hiding property of the commitment scheme, we can replace the commitment to Q_A by a commitment to 0^{2n} in the distributions. This makes $w_C = \mathrm{C}'_{\text{ideal}}(\mathrm{Commit}(0^{2n}))$, which is independent of Q_A.
- Since w_C is independent of w, the probability that $w_C = w$ is at most $\epsilon = \frac{1}{|\mathcal{D}|}$.
- If $w \neq w_C$, $Q_A(w)$ is within 2^{-n} statistical distance of U_n (since $Q_A(w)$ cannot take the value $Q_A(w_c)$) and independent of $Q_A(w_C)$ by pairwise independence of (non-constant linear) polynomials. Hence we have:

$$\begin{aligned}&\{w, Q_A(w), \mathrm{Commit}(0^{2n}), w_C, Q_A(w_C) | w_C \neq w\}\\ &\stackrel{\text{comp}}{\equiv} \{w, U_n, \mathrm{Commit}(0^{2n}), w_C, Q_A(w_C) | w_C \neq w\}.\end{aligned}$$ □

By Lemma 6.2, we have:

$$\{w, \Pi_A, \tau\} \stackrel{\epsilon}{\equiv} \{w, U_n, \tau\}.$$

Note that w' is independent of all the above variables; hence we have:

$$\{w, w', \Pi_A, \tau\} \stackrel{\epsilon}{\equiv} \{w, w', U_n, \tau\}.$$

We can then apply the deterministic polytime function $(h_{w'}(f^{n+\ell}(\cdot)), k_1(\cdot))$ using the second component w' to the third component of each distribution to obtain:

$$\{w, w', \tau, h_{w'}(f^{n+\ell}(\Pi_A)), k_1(\Pi_A)\} \stackrel{\epsilon}{\equiv} \{w, w', \tau, h_{w'}(f^{n+\ell}(U_n)), k_1(U_n)\}.$$

By applying the polytime function $\mathrm{C}_{2,\text{ideal}}(\cdot)$ to the last three components of each distribution, we have:

$$\{w, w', \tau, h_{w'}(f^{n+\ell}(\Pi_A)), k_1(\Pi_A), Q_C\} \stackrel{\epsilon}{\equiv} \{w, w', \tau, h_{w'}(f^{n+\ell}(U_n)), k_1(U_n), \tilde{Q}_C\}, \tag{8}$$

where $Q_C = \mathrm{C}_{2,\text{ideal}}(\tau, y_A, z_A)$ and $\tilde{Q}_C = \mathrm{C}_{2,\text{ideal}}(\tau, h_{w'}(f^{n+\ell}(U_n)), k_1(U_n))$.

We will now give an upper bound on the probability that $\mathrm{C}_{\text{hash}}{}'$ computes a correct validation message:

$$\begin{aligned}
&\Pr\big[\mathrm{C}'_{\text{hash}}(\tau, h_{w'}(f^{n+\ell}(\Pi_A)), k_1(\Pi_A), Q_C) = h_{w'}(f^{n+\ell}(Q_C(w)))\big] \\
&\quad \le \Pr\big[\mathrm{C}'_{\text{hash}}(\tau, h_{w'}(f^{n+\ell}(U_n)), k_1(U_n), \tilde{Q}_C) = h_{w'}(f^{n+\ell}(\tilde{Q}_C(w)))\big] + \epsilon + \mathrm{neg}(n) \qquad (9) \\
&\quad \le \Pr\big[\mathrm{C}'_{\text{hash}}(\tau, U_m, U_\ell, \tilde{Q}_C) = h_{w'}(f^{n+\ell}(\tilde{Q}_C(w)))\big] + \epsilon + \mathrm{neg}(n) \qquad (10) \\
&\quad \le \epsilon + 2^{-m} + \mathrm{neg}(n).
\end{aligned}$$

Equation (9) follows from (8), and (10) follows from the fact that $G(s) = (f^{n+\ell}(s), k_1(s))$ is a pseudorandom generator. The last inequality follows because the inputs to $\mathrm{C}'_{\text{hash}}$ are independent of w'. □

7. Key-Match Property for the Second Scheduling

Recall that Scheduling 2 is defined as "C sends the commitment c_B to B *before* A sends y_A".

The proof for this case relies on the almost pairwise independence property of the hash function $h_{w'}$, which says that for any two distinct values $x_1, x_2 \in \{0,1\}^n$, if the index w' is chosen at random and independently of x_1 and x_2, then being given the value $h_{w'}(x_1)$ does not help one guess the value $h_{w'}(x_2)$. Before y_A is sent, the hash index w' is random (since it has not been used by A or B). Thus, if we show that the values Π_A and Π_B are determined before y_A is sent, then w' is independent of $x_1 = f^{n+\ell}(\Pi_A)$ and $x_2 = f^{n+\ell}(\Pi_B)$ and the adversary cannot do much better than randomly guess $h_{w'}(x_2)$.

Π_A is certainly determined before y_A is sent (since A computes y_A based on Π_A). For Π_B, we observe that the security of augmented polynomial evaluation implies that, except with negligible probability, $\Pi_B = Q_C(w)$ for a polynomial Q_C such that $c_B = \mathrm{Commit}(Q_C)$ (unless $\Pi_B = \perp$, in which case B will certainly reject). Since c_B is sent before y_A (by the definition of Scheduling 2) and the commitment is perfectly binding, it follows that Q_C (and hence Π_B) is determined before y_A is sent.

7.1. *Mental Experiment*

Before proving the Key-Match Property for Scheduling 2, we will first consider a "Mental Experiment" where the adversary must explicitly output the polynomial Q_C. In the next section, we will reduce the Key-Match Property for Scheduling 2 to this Mental Experiment.

Protocol 7.1 (Mental Experiment).

1. **Inputs:** There are three parties A, B, C_m involved in the protocol. A and B have a joint password $(w, w') \overset{\text{R}}{\leftarrow} \mathcal{D} \times \mathcal{D}'$. In addition, A is given a random non-constant linear polynomial Q_A.
2. A sends Q_A to C_m.
3. C_m computes $Q_C = C_m(Q_A)$ and sends it to B.
4. B sends w to C.
5. A computes $Q_A(w)$ and sends $y_A = h_{w'}(f^{n+\ell}(Q_A(w)))$ to C. Note that the scheduling "Q_C is sent before y_A" is enforced.
6. C_m sends a string y_B to B.

The Mental Experiment is derived from the original protocol by giving A's inputs to the adversary C_m so that C_m can simulate the (A, C) interaction on its own. The crucial point of the Mental Experiment is that C_m sends the polynomial Q_C to B in the clear, therefore committing to it. Hence the points $Q_A(w)$ and $Q_C(w)$ are well-defined and independent of the hash index w' so that we can apply almost pairwise independence.

Proposition 7.2. *In the above Mental Experiment, for every (even computationally unbounded) adversary C_m, we have*

$$\Pr[\mathrm{C}_{\text{hash}}(C_m^{A(Q_A,w,w'),B(w,w')}) = h_{w'}(f^{n+\ell}(Q_C(w))) \wedge Q_A(w) \neq Q_C(w)] \leq \mu,$$

where C_{hash} *is a probabilistic polynomial-algorithm that takes as an input the above view* $C_m^{A(Q_A,w,w'),B(w,w')}$ *and tries to compute the hash value* $h_{w'}(f^{n+\ell}(Q_C(w)))$.

Proof. By definition of the Mental Experiment, $(Q_A(w), Q_C(w))$ can be computed from the view of the adversary C_m*before* $y_A = h_{w'}(f^{n+\ell}(Q_A(w)))$ is sent. Thus the values $(Q_A(w), Q_C(w))$ are independent of the hash index w'. Hence we obtain:

$$\begin{aligned}
&\Pr\big[\mathrm{C}_{\text{hash}}(C_m^{A(Q_A,w,w'),B(w,w')}) = h_{w'}(f^{n+\ell}(Q_C(w))) \wedge Q_A(w) \neq Q_C(w)\big] \\
&\quad \leq \Pr\big[\mathrm{C}_{\text{hash}}(C_m(Q_A, Q_C, w, h_{w'}(f^{n+\ell}(Q_A(w))))) = h_{w'}(f^{n+\ell}(Q_C(w))) \\
&\qquad \wedge Q_A(w) \neq Q_C(w)\big] \\
&\quad \leq \mu
\end{aligned}$$

where the last inequality follows from almost pairwise independence (the index of the hash function w' is random and independent from the points $f^{n+\ell}(Q_A(w))$ and $f^{n+\ell}(Q_C(w))$). $\square$

7.2. Reduction to the Mental Experiment

Proposition 7.3. *For every probabilistic polynomial-time real adversary C,*

$$\Pr[\mathrm{C}_{\mathrm{hash}}(C^{A(Q_A,w,w'),B(w,w')}) = h_{w'}(f^{n+\ell}(\Pi_B)) \wedge \Pi_A \neq \Pi_B \wedge \mathrm{Sch2}] \leq \mu + \mathrm{neg}(n).$$

Proposition 7.3 will be proved via a reduction to the Mental Experiment. We want to show that if an adversary succeeds in computing the correct hash value y_B in the original protocol, then we can build an adversary that computes the correct hash value in the Mental Experiment (and we know how to upper bound this success probability by Proposition 7.2).

In the Mental Experiment, the adversary C_m is forced to send the value Q_C in the clear before receiving $y_A = h_{w'}(f^{n+\ell}(\Pi_A))$. This is analogous to forcing the adversary C to open its commitment $c_B = \mathrm{Commit}(Q_C)$ in the original protocol. Thus, given an adversary C for the original protocol, we can build a corresponding adversary C_m in the mental experiment in the following natural way: run C until the commitment c_B must be opened, open the commitment to Q_C by exhaustive search, and then continue to run the adversary C. Note that we can afford the exhaustive search because the Mental Experiment is secure even against computationally unbounded adversaries C_m (cf. Proposition 7.2).

Actually, another difference between Scheduling 2 and the mental experiment is the possibility that B's output Π_B from the polynomial evaluation (B, C) may differ from $Q_C(w)$. However, we will argue that, by the security of augmented polynomial evaluation, Π_B equals either $Q_C(w)$ or $\perp$ (except with negligible probability). In case B's output is $\perp$, B will certainly reject and the Key-Match Property will be satisfied.

Proof. Since Commit is a perfectly binding commitment, the commitment c_B can be opened to a unique value, which we denote as Q_C. (Actually, it may be the case that c_B has no valid opening, in which case we write $Q_C = \perp$.)

Then we can break the analysis into three cases, depending on whether Π_B equals $Q_C(w)$, $\perp$, or some other value. (In case $Q_C = \perp$, we define $Q_C(w) = \perp$.)

$$\begin{aligned}
&\Pr\bigl[\mathrm{C}_{\mathrm{hash}}(C^{A(Q_A,w,w'),B(w,w')}) = h_{w'}(f^{n+\ell}(\Pi_B)) \wedge \Pi_A \neq \Pi_B \wedge \mathrm{Sch2}\bigr] \\
&\quad \leq \Pr\bigl[\mathrm{C}_{\mathrm{hash}}(C^{A(Q_A,w,w'),B(w,w')}) = h_{w'}(f^{n+\ell}(\Pi_B)) \wedge \Pi_A \neq \Pi_B \wedge \mathrm{Sch2} \\
&\qquad \wedge \Pi_B = Q_C(w)\bigr] \\
&\qquad + \Pr\bigl[\mathrm{C}_{\mathrm{hash}}(C^{A(Q_A,w,w'),B(w,w')}) = h_{w'}(f^{n+\ell}(\Pi_B)) \wedge \Pi_A \neq \Pi_B \wedge \mathrm{Sch2} \\
&\qquad \wedge \Pi_B = \perp\bigr] \\
&\qquad + \Pr\bigl[\mathrm{C}_{\mathrm{hash}}(C_{=}^{A(Q_A,w,w'),B(w,w')} h_{w'}(f^{n+\ell}(\Pi_B)) \wedge \Pi_A \neq \Pi_B \wedge \mathrm{Sch2} \\
&\qquad \wedge \Pi_B \notin \{Q_C(w), \perp\}\bigr].
\end{aligned}$$

Thus it suffices to bound each of the three probabilities on the right-hand side above. The second probability (case $\Pi_B = \perp$) is zero by convention (recall that, in the original protocol, B always rejects when $\Pi_B = \perp$). The third probability (case

$\Pi_B \notin \{Q_C(w), \perp\}$) is negligible by the security of augmented polynomial evaluation. To see this, note that we can simulate the polynomial evaluation (C, B) by an ideal polynomial evaluation, where we give the ideal adversary the entire state of A and C after the commitment c_B is sent. In the ideal setting, it always holds that $\Pi_B \in \{Q_C(w), \perp\}$. Thus, the same must hold with probability $1 - \text{neg}(n)$ in the real evaluation; otherwise, the real and ideal settings could be distinguished by a polynomial-time algorithm that has $Q_C(w)$ hardwired in as auxiliary input.

We are left with bounding the first probability, which corresponds to the case that $\Pi_B = Q_C(w)$. We handle this case by a reduction to the Mental Experiment. Specifically we convert C to an adversary C_m in the Mental Experiment by using exhaustive search to open the commitment c_B:

1. **Simulating the (A, C) polynomial evaluation:** Once A sends Q_A to C_m, C_m simulates on its own the beginning of the augmented polynomial evaluation between $A(Q_A)$ and C until C outputs its commitment c_B.
2. **Opening the commitment:** Using exhaustive search, C_m computes the unique value Q_C such that $c_B = \text{Commit}(Q_C, r)$ for some r. (If there is no such r, C_m sets $Q_C = \perp$.) C_m sends Q_C to B.
3. **Simulating the (C, B) polynomial evaluation:** C_m receives w from B and $y_A = h_{w'}(f^{n+\ell}(\Pi_A))$ from A. This gives C_m enough information to simulate the rest of both the (A, C) and (C, B) interactions on its own. In particular, C_m obtains and outputs the value $y_B = \text{C}_{\text{hash}}(C^{A(Q_A, w, w'), B(w, w')})$.

Observe that:

$$\begin{aligned}
&\Pr\big[\text{C}_{\text{hash}}(C^{A(Q_A, w, w'), B(w, w')}) = h_{w'}(f^{n+\ell}(\Pi_B)) \wedge \Pi_A \neq \Pi_B \wedge \text{Sch2} \wedge \Pi_B = Q_C(w)\big] \\
&\quad \leq \Pr\big[\text{C}_{\text{hash}}(C_m^{A(Q_A, w, w'), B(w, w')}) = h_{w'}(f^{n+\ell}(Q_C(w))) \wedge Q_A(w) \neq Q_C(w)\big] \\
&\quad \leq \mu,
\end{aligned}$$

where the last inequality is by the security of the Mental Experiment (Proposition 7.2). This completes the proof of the Key-Match Property for the Second Scheduling (Proposition 7.3). □

8. Adapting the GL Techniques to Our Protocol

Now that we have established the Key-Match Property, we will adapt the proofs of [15] to our protocol for the following steps:

Simulation of the (C, B) interaction: we show that the interaction (C, B) can be simulated by an adversary C' interacting only with A, even if the interaction (A, C) is concurrent.

Simulation of the (A, C') interaction: we show that the interaction (A, C') as a stand-alone protocol can be simulated.

Combining the above steps: by combining the above simulations, we obtain a proof of security against active adversaries.

For the sake of simplicity, we will first present the simulation of the (A, C') interaction. For each step, the modifications necessary to take into account the session-key challenge of Definition 2.2 are given.

8.1. *Simulation of the (A, C') Execution*

We will show that the view of C' when interacting with A only can be simulated by a non-interactive machine C'' in the following proposition.

Proposition 8.1. *For the dictionary $\mathcal{D} \times \mathcal{D}' = \mathcal{D} \times \{0, 1\}^{d'}$, for every polytime channel C' interacting with A only, there exists a non-interactive C'' such that for every auxiliary input σ,*

$$\{w, w', k_2(\Pi_A), \text{output}(C'^{A(Q_A, w, w')}(\sigma))\} \stackrel{\epsilon}{\equiv} \{w, w', U_n, \text{output}(C''(\sigma))\}$$

where $\epsilon = \frac{1}{|\mathcal{D}|}$.

Proof. By Lemma 6.2, we know that after the augmented polynomial evaluation, $\{w, \Pi_A\}$ is $(1 - \epsilon)$-indistinguishable from $\{w, U_n\}$ with respect to C''s view, that is,

$$\{w, \Pi_A, C'^{A(Q_A)}\} \stackrel{\epsilon}{\equiv} \{w, U_n, C'^{A(Q_A)}\}.$$

Note that w' is independent of all the above variables hence we have

$$\{w, w', \Pi_A, C'^{A(Q_A)}\} \stackrel{\epsilon}{\equiv} \{w, w', U_n, C'^{A(Q_A)}\}. \tag{11}$$

We will use (11) to establish that the session-key generated by A and the validation messages sent by A are pseudorandom with respect to the adversary's view. This is formalized by the following lemma.

Lemma 8.2. *For the dictionary $\mathcal{D} \times \mathcal{D}' = \mathcal{D} \times \{0, 1\}^{d'}$, for every polytime channel C' interacting with A only, we have*:

$$\begin{aligned}&\{w, w', k_2(\Pi_A), k_1(\Pi_A), \text{MAC}_{k_1(\Pi_A)}(t_A), h_{w'}(f^{n+\ell}(\Pi_A)), C'^{A(Q_A)}\}\\ &\quad\stackrel{\epsilon}{\equiv} \{w, w', U_n^1, U_\ell, \text{MAC}_{U_\ell}(t_A), U_m, C'^{A(Q_A)}\}.\end{aligned}$$

Proof of Lemma 8.2. We first apply the polytime function $G(\cdot) = (f^{n+\ell}(\cdot), k_1(\cdot), k_2(\cdot))$ to the third component of the distributions in 11 hence:

$$\begin{aligned}&\{w, w', k_2(\Pi_A), k_1(\Pi_A), f^{n+\ell}(\Pi_A), C'^{A(Q_A)}\}\\ &\quad\stackrel{\epsilon}{\equiv} \{w, w', k_2(U_n), k_1(U_n), f^{n+\ell}(U_n), C'^{A(Q_A)}\}.\end{aligned}$$

Since G is a pseudorandom generator, this implies that:

$$\{w, w', k_2(\Pi_A), k_1(\Pi_A), f^{n+\ell}(\Pi_A), C'^{A(Q_A)}\} \stackrel{\epsilon}{\equiv} \{w, w', U_n^1, U_\ell, U_n^2, C'^{A(Q_A)}\}.$$

By uniformity of the almost pairwise-independent hash functions, we obtain:

$$\begin{aligned}&\{w, w', k_2(\Pi_A), \mathrm{MAC}_{k_1(\Pi_A)}(t_A), h_{w'}(f^{n+\ell}(\Pi_A)), C'^{A(Q_A)}\}\\ &\stackrel{\epsilon}{\equiv} \{w, w', U_n^1, \mathrm{MAC}_{U_\ell}(t_A), U_m, C'^{A(Q_A)}\},\end{aligned} \tag{12}$$

where t_A is $A(Q_A)$'s transcript of the commitment and the augmented polynomial evaluation, which can be computed from $C'^{A(Q_A)}$. □

The non-interactive adversary C'' will do the following:

1. Generate a random non-constant linear polynomial Q_A.
2. Simulate the interaction between C' and $A(Q_A)$, from which it can compute the transcript t_A.
3. Generate random strings U_ℓ and U_m.
4. Output $(C'^{A(Q_A)}, U_m, \mathrm{MAC}_{U_\ell}(t_A))$.

Since the view of the adversary C' interacting with A only consists of $C'^{A(Q_A)}$, $y_A = h_{w'}(f^{n+\ell}(\Pi_A))$, and $z_A = \mathrm{MAC}_{k_1(\Pi_A)}(t_A)$, (12) establishes that

$$\{w, w', k_2(\Pi_A), \mathrm{output}(C'^{A(Q_A,w,w')})\} \stackrel{\epsilon}{\equiv} \{w, w', U_n, \mathrm{output}(C'')\}. \qquad \square$$

Augmented Definition for Proposition 8.1 Intuitively, handling the session-key challenge should be easy because the whole point of the session-key challenge is to deal with two concurrent executions (A, C) and (C, B) but here we are only considering a single execution (A, C').

We know that

$$\{w, w', k_2(\Pi_A), C'^{A(Q_A,w,w')}(\sigma)\} \stackrel{\epsilon}{\equiv} \{w, w', U_n, C''(\sigma)\}.$$

The session-key challenge is given to C' only after the entire execution (A, C') has been completed (recall that in our protocol A always accepts). The session-key challenge can be generated from each distribution by the distinguisher. We define $C'^{A(Q_A,w,w')}(\sigma, K_\beta) \stackrel{\text{def}}{=} (C'^{A(Q_A,w,w')}(\sigma), K_\beta)$ and $C''(\sigma, K_\beta) \stackrel{\text{def}}{=} (C''(\sigma), K_\beta)$. By the above discussion we have:

$$\{w, w', k_2(\Pi_A), C'^{A(Q_A,w,w')}(\sigma, K_\beta), \beta\} \stackrel{\epsilon}{\equiv} \{w, w', U_n, C''(\sigma, K_\beta), \beta\}$$

where on the left-hand side K_β is given when A concludes and is defined as:

$$K_\beta = \begin{cases} k_2(\Pi_A) & \text{if } \beta = 1, \\ U'_n & \text{if } \beta = 0 \end{cases}$$

and on the right-hand side K_β is defined as

$$K_\beta = \begin{cases} U_n & \text{if } \beta = 1, \\ U'_n & \text{if } \beta = 0. \end{cases}$$

8.2. *Simulation of the* (C, B) *Execution*

In the following proposition, we will show that the interaction (C, B) can be simulated by an adversary C' interacting only with A, even if the interaction (A, C) is concurrent.

Proposition 8.3. *For the dictionary $\mathcal{D} \times \mathcal{D}' = \mathcal{D} \times \{0,1\}^{d'}$, for every real adversary C interacting with A and B, there exists a probabilistic polynomial-time C' interacting only with A such that for every auxiliary input $\sigma \in \{0,1\}^{\mathrm{poly}(n)}$*

$$\{w, w', k_2(\Pi_A), \mathrm{output}(C'^{A(Q_A,w,w')}(\sigma))\}$$
$$\stackrel{\epsilon+\eta}{\equiv} \{w, w', k_2(\Pi_A), \mathrm{output}(C^{A(Q_A,w,w'),B(w,w')}(\sigma))\}$$

where $\eta = 2\mu + \epsilon$.

The proof of Proposition 8.3 relies on two facts:

- It is easy to simulate B in the augmented polynomial evaluation by security of two-party computation (see Lemma 8.4).
- B's decision bit can be simulated with high probability because of the Key-Match Property (see Lemma 8.6). We need for C' to simulate B's decision bit because the view of the real adversary $C^{A(Q_A,w,w'),B(w,w')}$ includes B's decision bit.

Note that Proposition 8.3 only refers to simulating the view of the real adversary C (which includes B's decision bit) rather than the outputs of all the parties.

We first show that B can be simulated in the augmented polynomial evaluation in the following lemma.

Lemma 8.4. *Let $\tilde{C}$ be a real adversary interacting with A and a modified party $\mathrm{B}_{\not{\mathrm{d}}\mathrm{ec}}$ ($\mathrm{B}_{\not{\mathrm{d}}\mathrm{ec}}$ does the same as B except that it does not output a decision bit). There exists C' interacting only with A such that*:

$$\{w, w', k_2(\Pi_A), \mathrm{output}(C'^{A(Q_A,w,w')}(\sigma))\}$$
$$\stackrel{\mathrm{comp}}{\equiv} \{w, w', k_2(\Pi_A), \mathrm{output}(\tilde{C}^{A(Q_A,w,w'),\mathrm{B}_{\not{\mathrm{d}}\mathrm{ec}}(w)}(\sigma))\}$$

where on the left-hand side $k_2(\Pi_A)$ refers to the output of A in the execution $C'^{A(Q_A,w,w')}(\sigma)$ and on the right-hand side $k_2(\Pi_A)$ refers to the output of A in the execution $\tilde{C}^{A(Q_A,w,w'),\mathrm{B}_{\not{\mathrm{d}}\mathrm{ec}}(w)}(\sigma)$.

Proof. Note that these distributions do not refer to $\mathrm{B}_{\not{\mathrm{d}}\mathrm{ec}}$'s output from the polynomial evaluation, hence we can switch $\mathrm{B}_{\not{\mathrm{d}}\mathrm{ec}}$'s input from w to a random password $\tilde{w} \in \mathcal{D}$ via the following claim.

Claim 8.5. *For every $w, w', Q_A, \tilde{w}$ and auxiliary input $\sigma \in \{0,1\}^{\mathrm{poly}(n)}$,*

$$\{\mathrm{output}(A), \tilde{C}^{A(Q_A,w,w'),\mathrm{B}_{\not{\mathrm{d}}\mathrm{ec}}(w)}(\sigma)\} \stackrel{\mathrm{comp}}{\equiv} \{\mathrm{output}(A), \tilde{C}^{A(Q_A,w,w'),\mathrm{B}_{\not{\mathrm{d}}\mathrm{ec}}(\tilde{w})}(\sigma)\},$$

where on the left-hand side output(A) *refers to the output of* A *in the execution* $\tilde{C}^{A(Q_A,w,w'),\mathrm{B}_{\not{d}\mathrm{ec}}(w)}(\sigma)$ *and on the right-hand side* output(A) *refers to the output of* A *in the execution* $\tilde{C}^{A(Q_A,w,w'),\mathrm{B}_{\not{d}\mathrm{ec}}(\tilde{w})}(\sigma)$.

Proof. Define C' that on input (w, w', Q_A) simulates the entire (A, C) execution, including computing output(A), on its own:

$$C'^{\mathrm{B}_{\not{d}\mathrm{ec}}(w)}(w, w', Q_A, \sigma) \equiv \{\mathrm{output}(A), \tilde{C}^{A(Q_A,w,w'),\mathrm{B}_{\not{d}\mathrm{ec}}(w)}(\sigma)\},$$
$$C'^{\mathrm{B}_{\not{d}\mathrm{ec}}(\tilde{w})}(w, w', Q_A, \sigma) \equiv \{\mathrm{output}(A), \tilde{C}^{A(Q_A,w,w'),\mathrm{B}_{\not{d}\mathrm{ec}}(\tilde{w})}(\sigma)\}.$$

Since C' and $\mathrm{B}_{\not{d}\mathrm{ec}}$ are executing the secure polynomial evaluation protocol in the stand-alone setting, there exists an ideal adversary $\mathrm{C}'_{\mathrm{ideal}}$ such that for every $w, w', Q_A, \tilde{w}, \sigma$,

$$\mathrm{C}'_{\mathrm{ideal}}(w, w', Q_A, \sigma) \overset{\mathrm{comp}}{\equiv} C'^{\mathrm{B}_{\not{d}\mathrm{ec}}(w)}(w, w', Q_A, \sigma),$$
$$\mathrm{C}'_{\mathrm{ideal}}(w, w', Q_A, \sigma) \overset{\mathrm{comp}}{\equiv} C'^{\mathrm{B}_{\not{d}\mathrm{ec}}(\tilde{w})}(w, w', Q_A, \sigma).$$

By transitivity of indistinguishability, we obtain the lemma. □

By Claim 8.5, we have that for every $w, w', Q_A, \tilde{w}$,

$$\{\mathrm{output}(A), \tilde{C}^{A(Q_A,w,w'),\mathrm{B}_{\not{d}\mathrm{ec}}(w)}(\sigma)\} \overset{\mathrm{comp}}{\equiv} \{\mathrm{output}(A), \tilde{C}^{A(Q_A,w,w'),\mathrm{B}_{\not{d}\mathrm{ec}}(\tilde{w})}(\sigma)\}.$$

Applying this to w, w' and Q_A chosen uniformly at random, we obtain:

$$\begin{aligned}&\{w, w', Q_A(w), \mathrm{output}(\tilde{C}^{A(Q_A,w,w'),\mathrm{B}_{\not{d}\mathrm{ec}}(\tilde{w})}(\sigma))\}\\ &\quad\overset{\mathrm{comp}}{\equiv} \{w, w', Q_A(w), \mathrm{output}(\tilde{C}^{A(Q_A,w,w'),\mathrm{B}_{\not{d}\mathrm{ec}}(w)}(\sigma))\}.\end{aligned} \tag{13}$$

Hence for any adversary $\tilde{C}$ interacting with A and $\mathrm{B}_{\not{d}\mathrm{ec}}$, we build an adversary C' that simulates $\mathrm{B}_{\not{d}\mathrm{ec}}$ on its own as follows:

1. Generate an arbitrary element $\tilde{w}$.
2. Simulate the polynomial evaluation between C and $\mathrm{B}_{\not{d}\mathrm{ec}}(\tilde{w})$.

Thus C' only interacts with $A(Q_A, w, w')$ and we have:

$$\begin{aligned}&\{w, w', Q_A(w), \mathrm{output}(C'^{A(Q_A,w,w')}(\sigma))\}\\ &\quad\overset{\mathrm{comp}}{\equiv} 1\{w, w', Q_A(w), \mathrm{output}(\tilde{C}^{A(Q_A,w,w'),\mathrm{B}_{\not{d}\mathrm{ec}}(\tilde{w})}(\sigma))\}.\end{aligned} \tag{14}$$

Combining (13) and (14) and recalling that $k_2(\Pi_A) = k_2(Q_A(w))$, the lemma follows. □

Augmented Definition for Lemma 8.4 In the case of the augmented definition, the proof of Lemma 8.4 still holds because Claim 8.5 will hold for every session-key challenge given by A. Hence we have

$$\begin{aligned}&\{w, w', k_2(\Pi_A), \mathrm{output}(C'^{A(Q_A,w,w')}(\sigma, K_\beta)), \beta\}\\ &\quad\overset{\mathrm{comp}}{\equiv} \{w, w', k_2(\Pi_A), \mathrm{output}(\tilde{C}^{A(Q_A,w,w'),\mathrm{B}_{\not{d}\mathrm{ec}}(w)}(\sigma, K_\beta)), \beta\}\end{aligned}$$

where K_β is given when A concludes and is defined as:

$$K_\beta = \begin{cases} k_2(\Pi_A) & \text{if } \beta = 1, \\ U_n' & \text{if } \beta = 0. \end{cases}$$

To establish Proposition 8.3, it remains to show that B's decision bit can be simulated with high probability.

Lemma 8.6. *Let C be a real adversary interacting with A and B. There exists $\tilde{C}$ interacting with A and $\mathrm{B}_{\not{d}\mathrm{ec}}$ such that*

$$\{w, w', k_2(\Pi_A), \mathrm{output}(\tilde{C}^{A(Q_A,w,w'),\mathrm{B}_{\not{d}\mathrm{ec}}(w)}(\sigma))\}$$
$$\stackrel{\epsilon+\eta}{\equiv} \{w, w', k_2(\Pi_A), \mathrm{output}(C^{A(Q_A,w,w'),B(w,w')}(\sigma))\}.$$

Proof. The proof of Lemma 8.6 relies on the fact that the decision bit of B can be predicted by C with high probability because of the Key-Match Property and the following claim. Claim 8.7 below states that if $\Pi_A = \Pi_B$ and the adversary C was not reliable (hence the transcripts t_A and t_B differ), then the adversary cannot compute a MAC such that B will accept. Hence the decision bit of B can be predicted by C with high probability: if C is reliable, then B will accept; otherwise, B will reject.

Claim 8.7. *For every C interacting with A and $\mathrm{B}_{\not{d}\mathrm{ec}}$, the probability that $t_B \neq t_A$ and C computes $\mathrm{MAC}_{k_1(\Pi_A)}(t_B)$ is at most $\epsilon + \mathrm{neg}(n)$.*

Proof. First, we will remove B by modifying C into C' from Lemma 8.4, that simulates B in the polynomial evaluation phase. We know from Lemma 8.2 that:

$$\{w, w', k_2(\Pi_A), k_1(\Pi_A), \mathrm{MAC}_{k_1(\Pi_A)}(t_A), h_{w'}(f^{n+\ell}(\Pi_A)), C'^{A(Q_A)}\}$$
$$\stackrel{\epsilon}{\equiv} \{w, w', U_n^1, U_\ell, \mathrm{MAC}_{U_\ell}(t_A), U_m, C'^{A(Q_A)}\}.$$

We will bound the probability that the adversary computes the correct MAC for $t \neq t_A$:

$$\Pr\big[\mathrm{C_{mac}}(C'^{A(Q_A)}, h_{w'}(f^{n+\ell}(\Pi_A)), \mathrm{MAC}_{k_1(\Pi_A)}(t_A)) = \mathrm{MAC}_{k_1(\Pi_A)}(t)\big]$$
$$\leq \Pr\big[\mathrm{C_{mac}}(C'^{A(Q_A)}, U_m, \mathrm{MAC}_{U_\ell}(t_A)) = \mathrm{MAC}_{U_\ell}(t)\big] + \epsilon + \mathrm{neg}(n)$$
$$\leq \epsilon + \mathrm{neg}(n)$$

where the last inequality comes from the one-time MAC with pseudorandomness property. □

Using Claim 8.7, we obtain the following adversary $\tilde{C}$: $\tilde{C}$ interacts with A and $\mathrm{B}_{\not{d}\mathrm{ec}}$ by passing their messages to C. Since $\tilde{C}$ has the transcript of the interactions (A, C) and (C, B), $\tilde{C}$ can tell whether C was reliable or not. If C was reliable, $\tilde{C}$ predicts that $\mathrm{dec}_B = \texttt{accept}$ (since B always accepts if C is reliable), otherwise, it predicts $\mathrm{dec}_B =$

`reject`. We know that $\Pr[\tilde{C}$ predicts incorrectly$] = \Pr[\mathrm{dec}_B = \texttt{accept} \wedge \mathit{reliable}_C = \texttt{false}]$. In order to prove Lemma 8.6, it remains to show that for any C,

$$\Pr[\mathrm{dec}_B = \texttt{accept} \wedge \mathit{reliable}_C = \texttt{false}] < \epsilon + \eta + \mathrm{neg}(n).$$

We will break the probability that B accepts when C was not reliable into the following cases:

$$\begin{aligned}
&\Pr[\mathrm{dec}_B = \texttt{accept} \wedge \mathit{reliable}_C = \texttt{false}] \\
&\quad = \Pr[\mathrm{dec}_B = \texttt{accept} \wedge \mathit{reliable}_C = \texttt{false} \wedge \Pi_A \neq \Pi_B] \\
&\qquad + \Pr[\mathrm{dec}_B = \texttt{accept} \wedge \mathit{reliable}_C = \texttt{false} \wedge \Pi_A = \Pi_B] \\
&\quad \leq (\eta + \mathrm{neg}(n)) + (\epsilon + \mathrm{neg}(n)) \\
&\quad \leq \epsilon + \eta + \mathrm{neg}(n).
\end{aligned}$$

The first step just considers whether the keys Π_A and Π_B are equal. The second step follows from the Key-Match Property (we know that if $\Pi_A \neq \Pi_B$, then with high probability, B will reject) and Claim 8.7 (we know that if C was not reliable i.e., $t \neq t_A$, then with high probability C will not compute the correct MAC for t and B will reject). $\square$

Augmented Definition for Lemma 8.6

- C **is not reliable and** B **concludes first:** $\tilde{C}$ will set B's simulated decision bit to be $\mathrm{dec}_B = \texttt{reject}$ and its simulated session-key challenge to be $\perp$. Note that if B concludes first, then with high probability B would indeed reject (which follows from the fact that if C is not reliable, then with high probability B will reject as shown above).
- A **concludes first:** Lemma 8.6 must be slightly modified. One can show using Lemma 8.2 that the probability that $t_B \neq t_A$ and C computes $\mathrm{MAC}_{k_1(\Pi_A)}(t_B)$ is at most $\epsilon + \mathrm{neg}(n)$ even if $k_2(\Pi_A) = k_A$ is given.

From the above two arguments, we have:

$$\begin{aligned}
&\{w, w', k_2(\Pi_A), \mathrm{output}(\tilde{C}^{A(Q_A, w, w'), \mathrm{B}_{\mathrm{dec}}(w)}(\sigma, K_\beta)), \beta\} \\
&\quad \stackrel{\epsilon+\eta}{\equiv} \{w, w', k_2(\Pi_A), \mathrm{output}(C^{A(Q_A, w, w'), B(w, w')}(\sigma, K_\beta)), \beta\},
\end{aligned}$$

where on left-hand side K_β is given when A concludes and is defined as:

$$K_\beta = \begin{cases} k_2(\Pi_A) & \text{if } \beta = 1, \\ U'_n & \text{if } \beta = 0 \end{cases}$$

and on the right-hand side the session-key challenge K_β is given once the first party (either A or B) concludes with output L_1 and is defined as:

$$K_\beta = \begin{cases} L_1 & \text{if } \beta = 1 \text{ or } L_1 = \perp, \\ U'_n & \text{if } \beta = 0 \text{ and } L_1 \neq \perp. \end{cases}$$

8.3. *Security Theorem*

Combining Propositions 8.3 and 8.1, we will now prove our basic security theorem against active adversaries. Using the simulations guaranteed by Propositions 8.3 and 8.1 we are guaranteed the existence of a non-interactive adversary C'' whose view is indistinguishable from that of a real adversary C interacting with A and B.

We will need to modify this non-interactive adversary C'' into an ideal adversary (as in Definitions 2.1 and 2.2) as well as include the inputs and outputs of the honest parties A and B in the ideal and real distributions.

Theorem 8.8 (Theorem 4.2, restated). *For the dictionary $\mathcal{D} \times \mathcal{D}' = \mathcal{D} \times \{0,1\}^{d'}$, for every probabilistic polynomial-time real adversary C, there exists a polynomial-time ideal model adversary $\hat{C}$ for Definition 2.2 such that for any $\sigma \in \{0,1\}^{\mathrm{poly}(n)}$*

$$\{\mathrm{IDEAL}_{\hat{C}}(\mathcal{D},\sigma)\} \stackrel{3\epsilon+2\eta}{\equiv} \{\mathrm{REAL}_C(\mathcal{D},\sigma)\}$$

where $\eta = 2\mu + \epsilon$.

Theorem 4.2 follows from the above by noting that the two distributions are $(1 - O(\gamma))$-indistinguishable for $\gamma = \max\{\epsilon, \mu\}$, where $\epsilon = 1/|\mathcal{D}|$ and $\mu = O(n/|\mathcal{D}'|^{1/3} \log|\mathcal{D}'|) = O(n/|\mathcal{D}'|^{1/3})$.

Proof. From the previous two sections, we know that there exists a non-interactive C'' such that

$$\{w, w', U_n, C''(\sigma)\} \stackrel{2\epsilon+\eta}{\equiv} \{w, w', k_2(\Pi_A), \mathrm{output}(C^{A,B}(\sigma))\}.$$

The ideal model adversary $\hat{C}$ does the following:

- $\hat{C}$ decides that A will conclude first and accept in the ideal model.
- C invokes C'' that is non-interactive.
- According to the view output by C'', $\hat{C}$ will decide whether B accepts or not in the ideal execution.
- $\hat{C}$ outputs the output of C''.

$$\Rightarrow \{w, w', U_n, \hat{C}(\sigma)\} \stackrel{2\epsilon+\eta}{\equiv} \{w, w', k_2(\Pi_A), \mathrm{output}(C^{A,B}(\sigma))\}. \tag{15}$$

We now need to include B's output in the above distributions. Let D be a distinguisher for $\mathrm{IDEAL}_{\hat{C}}$ and REAL_C. We will consider the different cases, whether B accepts or not.

If B rejects

$$\Pr[D(\mathrm{IDEAL}_{\hat{C}}) = 1 \wedge \mathrm{dec}_B = \mathtt{reject}]$$
$$= \Pr[D(w, w', U_n, \perp, \hat{C}) = 1 \wedge \mathrm{dec}_B = \mathtt{reject}],$$
$$\Pr[D(\mathrm{REAL}_C) = 1 \wedge \mathrm{dec}_B = \mathtt{reject}]$$
$$= \Pr[D(w, w', k_2(\Pi_A), \perp, C^{A,B}) = 1 \wedge \mathrm{dec}_B = \mathtt{reject}].$$

In the ideal model, we are guaranteed that when B rejects, $\hat{C}$ will send $b=0$ to the trusted party, causing B to output $\perp$. In the real protocol, when B rejects, it always outputs $\perp$. But B's decision bit is contained in the view $\hat{C}$ (as simulated by C'') and in the view $C^{A,B}$ so by (15) the difference between $\Pr[D(\mathrm{IDEAL}_{\hat{C}})=1 \wedge \mathrm{dec}_B = \texttt{reject}]$ and $\Pr[D(\mathrm{REAL}_C)=1 \wedge \mathrm{dec}_B = \texttt{reject}]$ is at most $2\epsilon+\eta+\mathrm{neg}(n)$.

If B accepts

- Suppose C was reliable: in the real model, B always accepts and outputs $k_2(\Pi_A)$; in the ideal model, B outputs U_n. C is acting like a passive adversary, so we know that $\mathrm{IDEAL}_{\hat{C}} \overset{\text{comp}}{\equiv} \mathrm{REAL}_C$.
- Suppose C was not reliable, but B accepts. From the proof of Theorem 8.3, we know that $\Pr[\mathrm{dec}_B = \texttt{accept} \wedge \mathit{reliable}_C = \texttt{false}] \le \epsilon+\eta+\mathrm{neg}(n)$, whether in the real model or in the one simulated by $\hat{C}$.

$$\begin{aligned}&\big|\Pr[D(\mathrm{IDEAL}_{\hat{C}})=1 \wedge \mathrm{dec}_B = \texttt{accept} \wedge \mathit{reliable}_C = \texttt{false}] \\ &\qquad - \Pr[D(\mathrm{REAL}_C)=1 \wedge \mathrm{dec}_B = \texttt{accept} \wedge \mathit{reliable}_C = \texttt{false}]\big| \\ &\quad \le \epsilon+\eta+\mathrm{neg}(n).\end{aligned}$$

Combining all the above cases, we have that the ideal distribution and the real distribution are distinguishable with probability at most $3\epsilon+2\eta$. □

Augmented Definition for Theorem 8.8 From the previous sections, we know that

$$\{w, w', U_n, C''(\sigma, K_\beta), \beta\} \overset{2\epsilon+\eta}{\equiv} \{w, w', k_2(\Pi_A), C^{A,B}(\sigma, K_\beta), \beta\}$$

where on the left-hand side K_β is defined as

$$K_\beta = \begin{cases} U_n & \text{if } \beta = 1, \\ U'_n & \text{if } \beta = 0 \end{cases}$$

and on the right-hand side the session-key challenge K_β is given once the first party (either A or B) concludes with output L_1:

$$K_\beta = \begin{cases} L_1 & \text{if } \beta = 1 \text{ or } L_1 = \perp, \\ U_n & \text{if } \beta = 0 \text{ and } L_1 \ne \perp. \end{cases}$$

The ideal adversary $\hat{C}$ does the following:

- $\hat{C}$ decides that A will conclude first and accept. The trusted party chooses $\beta \overset{\text{R}}{\leftarrow} \{0,1\}$ and gives $\hat{C}$ the string K_β where

$$K_\beta = \begin{cases} U_n & \text{if } \beta = 1, \\ U'_n & \text{if } \beta = 0. \end{cases}$$

- C invokes $C''(\sigma, K_\beta)$ that is non-interactive.

- According to the view output by C'', $\hat{C}$ will decide whether B accepts or not in the ideal execution.
- $\hat{C}$ outputs the output of $C''(\sigma, K_\beta)$.

9. Additional Security Theorems

We will now show the shared dictionary of the form $\mathcal{D} \times \{0,1\}^d$ required in Theorem 4.2 can be realized from several other types of dictionaries $\mathcal{D}''$, achieving security bounds of the form $(\mathrm{poly}(n)/|\mathcal{D}''|)^{\Omega(1)}$ in all cases.

Single Random Password We can split a single random password from a dictionary $\mathcal{D}'' = \{0,1\}^{d''}$ into two parts, one of length d and one of length $d' = d'' - d$. Optimizing, we set $d = (d'' - 3\log n)/4$, and obtain a security bound of

$$\gamma = \max\left\{\frac{1}{2^d}, O\left(\frac{n}{2^{d'/3}}\right)\right\} = O\left(\frac{n^3}{|\mathcal{D}''|}\right)^{1/4}.$$

Arbitrary Password with Common Random String We can convert a password from an arbitrary dictionary $\mathcal{D}''' \subseteq \{0,1\}^n$ into a single random password (as in the previous construction) in the common random string model, using randomness extractors, which we define now.

A random variable X is a *k-source* if for all x, $\Pr[X = x] \leq 2^{-k}$. (In other words, X has *min-entropy* at least k.) Note that the uniform distribution on $\mathcal{D}'''$ is a k-source for $k = \log|\mathcal{D}'''|$.

Definition 9.1 [23]. A function $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^m$ is a (*strong*) (k,α)-*extractor* if for every k-source X on $\{0,1\}^n$, the random variable $(U_\ell, \mathrm{Ext}(X, U_\ell))$ is α-close to (U_ℓ, U_m).

That is, using a random *seed* of length ℓ, the function Ext extracts m almost-uniform bits from the k-source X. We call Ext *explicit* if it is computable in polynomial time (in n and ℓ).

We will use the following construction of "low min-entropy" extractors.

Lemma 9.2 [26]. *For every n, $k \leq n$, and $\alpha > 0$, there exists an explicit (k,α)-extractor* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^m$ *with $\ell = O(\log n + m) + 2\log(1/\alpha)$ and $m = k - 2\log(1/\alpha) - O(1)$.*

To use extractors with our protocol, we view the common random string as the seed for the extractor, and apply the extractor to convert the password from the arbitrary dictionary $\mathcal{D}''' \subseteq \{0,1\}^n$ into $d'' = m$ almost-uniform bits, which we use in place of the "single random password" in the previous construction. We pay an additive loss of α (the error of the extractor) in the security bound, and also lose because the extractor cannot extract *all* of the min-entropy in the source (i.e. d'' will be smaller than $\log|\mathcal{D}'''|$). Optimizing with the extractor of Lemma 9.2, we set $k = \log|\mathcal{D}'''|$ and

$\alpha = (n^3/|\mathcal{D}'''|)^{1/6}$, and obtain $d'' = m = k - 2\log(1/\alpha) - O(1)$, i.e. $|\mathcal{D}''| = 2^{d''} = \Omega(\alpha^2 \cdot 2^k) = \Omega(n \cdot |\mathcal{D}'''|^{2/3})$. Then we have:

$$\gamma = O\left(\frac{n^3}{|\mathcal{D}''|}\right)^{1/4} = O\left(\frac{n^3}{n \cdot |\mathcal{D}'''|^{2/3}}\right)^{1/4} = O\left(\frac{n^3}{|\mathcal{D}'''|}\right)^{1/6},$$

for a final security bound of

$$\gamma + \alpha = O\left(\frac{n^3}{|\mathcal{D}'''|}\right)^{1/6}.$$

The length of our common random string is $\ell = O(\log n + k) = O(\log n + \log|\mathcal{D}'''|)$. Note that this is only logarithmic in the security parameter n, whereas the protocols of [13,18] require common reference strings of length polynomially related to n. On the other hand, using our protocol requires knowing (or assuming) a lower bound on the size of the dictionary (and this lower bound is what determines the security). The protocols of [13,15,18] do not require such a lower bound.

Two Independent Passwords If the parties share *two* independent passwords w_1, w_2 coming from arbitrary dictionaries $\mathcal{D}_1, \mathcal{D}_2 \subseteq \{0,1\}^r$, then they can apply a (seedless) extractor for 2 independent weak random sources [9] to convert these into a single random password. Even better is to use the following variant of 2-source extractors:

Definition 9.3 [11]. A function $\mathrm{Ble} : \{0,1\}^r \times \{0,1\}^r \to \{0,1\}^m$ is a (*strong*) (k_1, k_2, α)-*blender* if for every k_1-source X_1 and independent k_2-source X_2 on $\{0,1\}^r$, the random variable $(X_1, \mathrm{Ble}(X_1, X_2))$ is α-close to (X_1, U_m).

Thus, if the parties share two independent passwords w_1, w_2 coming from arbitrary dictionaries, a strong blender can be used to convert w_2 into an almost-uniform string $w' = \mathrm{Ble}(w_1, w_2)$ that is essentially independent of the other password, and thus (w_1, w') can be used in our original construction. Nonconstructively, strong (k_1, k_2, α)-blenders are known to exist with $m = k_2 - 2\log(1/\alpha) - O(1)$, provided that $k_1 > \log r + 2\log(1/\alpha) + O(1)$. If there were explicit constructions matching these parameters, we would obtain a protocol with security bound of

$$\gamma = O\left(\max\left\{\left(\frac{n}{|\mathcal{D}_1|}\right)^{1/2}, \left(\frac{n^3}{|\mathcal{D}_2|}\right)^{1/4}\right\}\right).$$

Unfortunately, explicit constructions of blenders (or 2-source extractors) are only known in cases when either k_1 or k_2 are at least $r/2$. (See [12] and the references therein for the current state-of-the-art.) Thus we would not obtain a protocol that could work for arbitrary dictionaries $\mathcal{D}_1, \mathcal{D}_2 \subseteq \{0,1\}^n$ of size $\mathrm{poly}(n)$. However, these constructions do allow us to obtain a protocol for arbitrary dictionaries $\mathcal{D}_1, \mathcal{D}_2 \subseteq \{0,1\}^r$ of size, say, $2^{.51r}$, for $r \le n$ and even $r = O(\log n)$.

Acknowledgements

We thank Oded Goldreich and Yehuda Lindell for their encouragement and an inspiring discussion which led to a major simplification of our protocol. We are also grateful to Mihir Bellare for pointing out the extension of our protocol to arbitrary dictionaries in the common random string model. We thank the referees for their many helpful comments, one of which led to a substantial simplification and improvement to the analysis in Sect. 7.

Appendix A. Secure Two-Party Computation

This presentation is taken from [14]. We will describe secure two-party computation for the special case of single-output functionalities, i.e., functionalities where only one party obtains an output. Indeed, we will only use tools from secure two-party computation when dealing with the augmented polynomial evaluation functionality, which is a single-output functionality. Furthermore, for simplicity, we will restrict our description to the case where none of the parties aborts and at least one of the two parties is honest.

Let $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$ be a deterministic single-output functionality, i.e., f is of the form $f(x,y) = (f_1(x,y), \lambda)$.

We first define the ideal model:

Inputs Each party obtains an input denoted x and y respectively.

Sending inputs to the trusted party An honest party will always send its input x or y to the trusted party. A malicious party will send some input x' or y', which may depend on its initial input and auxiliary input.

Answer of the trusted party Upon obtaining (x,y), the trusted party will reply with $f_1(x,y)$ to the first party.

Output An honest party will always output the message obtained from the trusted party. A malicious party may output a polytime computable function of its initial input, its auxiliary input and the message obtained from the trusted party.

Let (B_1, B_2) be a pair of probabilistic polynomial-time strategies in the ideal model, such that at least one of the two parties is honest. The joint distribution of f under (B_1, B_2) in the ideal model, on input pair (x,y) and auxiliary input z, denoted by $\mathrm{IDEAL}_{f,B_1(z),B_2(z)}$, is:

- In the case where B_1 is honest, $\mathrm{IDEAL}_{f,B_1(z),B_2(z)}(x,y) = (f_1(x, B_2(y,z)), B_2(y,z,\lambda))$.
- In the case where B_2 is honest, $\mathrm{IDEAL}_{f,B_1(z),B_2(z)}(x,y) = (B_1(x,z, f_1(B_1(x,z),y)), \lambda)$.

We now describe the real model. Let Π be a two-party protocol for computing f. Let (A_1, A_2) be a pair of probabilistic polynomial-time representing strategies in the real model, such that at least one of two parties is honest (i.e., follows the strategy specified by Π). The joint execution of Π under (A_1, A_2) in the real model, on input pair (x,y) and auxiliary input z, denoted by $\mathrm{REAL}_{\Pi,A_1(z),A_2(z)}$ is defined as the output pair resulting from the interaction between $A_1(x,z)$ and $A_2(x,z)$.

Definition A.1. Let $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$ be a deterministic single-output functionality and Π be a two-party protocol for computing f. Protocol Π *securely computes* f if for every probabilistic polynomial-time pair (A_1, A_2) (such that at least one party follows the strategy specified by Π), there exists a probabilistic polynomial-time pair (B_1, B_2) (such that the corresponding party is honest in the ideal model) such that:

$$\{\mathrm{IDEAL}_{f,B_1(z),B_2(z)}(x,y)\}_{x,y,z} \stackrel{\text{comp}}{\equiv} \{\mathrm{REAL}_{\Pi,A_1(z),A_2(z)}(x,y)\}_{x,y,z}.$$

Assuming the existence of enhanced trapdoor permutations, it is known how to obtain a secure protocol for any two-party computation ([28], see [14]).

Appendix B. Almost Pairwise Independence

Here we recall a standard construction of almost pairwise-independent hash functions, modified to ensure that all of the hash functions are regular (which is typically not required in the definition of almost pairwise independence).

Lemma B.1. *For a given dictionary* $\mathcal{D}' = \{0,1\}^{d'} \subseteq \{0,1\}^n$, *there exists a family of almost pairwise-independent hash functions* $\mathcal{H} = \{h_{w'} : \{0,1\}^n \to \{0,1\}^m\}$ *for* $\mu = O(\frac{n}{d' 2^{d'/3}}) = O(\frac{n}{|\mathcal{D}'|^{1/3} \log |\mathcal{D}'|})$.

Proof sketch. Set $m = \lfloor d'/3 \rfloor$ and let $\mathbb{F}$ be the finite field $\mathrm{GF}(2^m)$. Let $k = \lceil n/m \rceil$. An element $p = (p_0, p_1, \ldots, p_{k-1}) \in \mathbb{F}^k$ can be seen as the coefficients of a polynomial of degree at most $(k-1)$ over the finite field $\mathbb{F}$.

Let the index w' be a triple $(\alpha, \beta, \gamma) \in \mathbb{F} \times \mathbb{F} \times \mathbb{F}$. We define the hash function $h_{w'} = h_{\alpha,\beta,\gamma}$ as follows:

$$h_{\alpha,\beta,\gamma} = \begin{cases} \alpha \cdot p(\beta) + \gamma, & \alpha \neq 0, \\ p(\beta) + \gamma, & \alpha = 0. \end{cases}$$

That is, we evaluate the polynomial p at the point $\beta \in \mathbb{F}$, and then apply the linear function $x \mapsto \alpha \cdot x + \gamma$ (unless $\alpha = 0$, in which case we use $x \mapsto x + \gamma$). Note that it requires $3m \le d'$ bits to specify $w' = (\alpha, \beta, \gamma)$ and the hash functions have input length $k \cdot m \ge n$.

We will now verify that $\mathcal{H} = \{h_{\alpha,\beta,\gamma} : \mathbb{F}^k \to \mathbb{F}\}$ is a family of almost pairwise-independent hash functions. For the uniformity condition, note that for every $p \in F^k$, when we choose (α, β, γ) uniformly at random from $\mathbb{F} \times \mathbb{F} \times \mathbb{F}$, it holds that $h_{\alpha,\beta,\gamma}(p)$ is uniform over $\mathbb{F}$; indeed this holds even when α, β are fixed and γ is chosen uniformly at random. For a fixed index $(\alpha, \beta, \gamma) \in \mathbb{F} \times \mathbb{F} \times \mathbb{F}$ and a fixed element $y \in \mathbb{F}$, $\Pr_{p \in \mathbb{F}^k}[h_{\alpha,\beta,\gamma}(p) = y] = 1/|\mathbb{F}|$, hence the function $h_{\alpha,\beta,\gamma}$ is regular.

For the almost pairwise-independence, we will show that

$$\Pr_{\alpha,\beta,\gamma}[h_{\alpha,\beta,\gamma}(q) = y_2 | h_{\alpha,\beta,\gamma}(p) = y_1] \le \frac{k+1}{|\mathbb{F}|} = O\left(\frac{n/d'}{2^{d'/3}}\right)$$

for all $p \neq q \in \mathbb{F}^k$, and $y_1, y_2 \in \mathbb{F}$. To bound this, we first note that:

$$\Pr_{\alpha,\beta,\gamma}[h_{\alpha,\beta,\gamma}(q) = y_2 | h_{\alpha,\beta,\gamma}(p) = y_1] = \frac{\Pr_{\alpha,\beta,\gamma}[h_{\alpha,\beta,\gamma}(p) = y_1 \wedge h_{\alpha,\beta,\gamma}(q) = y_2]}{\Pr_{\alpha,\beta,\gamma}[h_{\alpha,\beta,\gamma}(p) = y_1]}$$
$$= |\mathbb{F}| \cdot \Pr_{\alpha,\beta,\gamma}[h_{\alpha,\beta,\gamma}(p) = y_1 \wedge h_{\alpha,\beta,\gamma}(q) = y_2].$$

Thus it suffices to show that

$$\Pr_{\alpha,\beta,\gamma}[h_{\alpha,\beta,\gamma}(p) = y_1 \wedge h_{\alpha,\beta,\gamma}(q) = y_2] \leq \frac{k+1}{|\mathbb{F}|^2}.$$

Let p, q, y_1, y_2 be fixed.

- Suppose $p(\beta) = q(\beta)$ (i.e. β is a root of the polynomial $(p - q)$, which happens with probability at most $(k-1)/|\mathbb{F}|$). Then $h_{\alpha,\beta,\gamma}(p) = h_{\alpha,\beta,\gamma}(q)$ for every α, γ, and this value is distributed uniformly at random in $\mathbb{F}$ (over the choice of γ). Thus, the probability that $h_{\alpha,\beta,\gamma}(p) = y_1$ and $h_{\alpha,\beta,\gamma}(q) = y_2$ is at most $1/|\mathbb{F}|$ (given that $p(\beta) = q(\beta)$).
- Suppose $p(\beta) \neq q(\beta)$. Then, over the choice α, γ, the values $\alpha \cdot p(\beta) + \gamma$ and $\alpha \cdot q(\beta) + \gamma$ are uniform and independent in $\mathbb{F}$. Thus, the probability that $\alpha \cdot p(\beta) + \gamma = y_1$ and $\alpha \cdot q(\beta) + \gamma = y_2$ equals $1/|\mathbb{F}|^2$. However, we need to bound this probability for $h_{\alpha,\beta,\gamma}$, which differs from these in case $\alpha = 0$. But the probability (over α and γ) that $\alpha = 0$ and $h_{0,\beta,\gamma}(p) = p(\beta) + \gamma = y_1$ is $1/|\mathbb{F}|^2$.

In total, we have

$$\Pr_{\alpha,\beta,\gamma}[h_{\alpha,\beta,\gamma}(p) = y_1 \wedge h_{\alpha,\beta,\gamma}(q) = y_2] \leq \frac{k-1}{|\mathbb{F}|} \cdot \frac{1}{|\mathbb{F}|} + \frac{2}{|\mathbb{F}|^2} = \frac{k+1}{|\mathbb{F}|^2},$$

as desired. □

References

[1] B. Barak, Constant-Round Coin-Tossing with a Man in the Middle or Realizing the Shared Random String Model, in *IEEE Symposium on Foundations of Computer Science* (2002), pp. 345–355

[2] M. Bellare, D. Pointcheval, P. Rogaway, Authenticated Key Exchange Secure against Dictionary Attacks, in *Advances in Cryptology—Eurocrypt 2000 Proceedings*. Lecture Notes in Computer Science, vol. 1807 (Springer, Berlin, 2000), pp. 139–155

[3] M. Bellare, P. Rogaway, Entity Authentication and Key Distribution, in *Advances in Cryptology—Crypto 93 Proceedings*. Lecture Notes in Computer Science, vol. 773 (Springer, Berlin, 1994), pp. 232–249

[4] S. Bellovin, M. Merritt, Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks, in *ACM/IEEE Symposium on Research in Security and Privacy* (1992), pp. 72–84

[5] S. Bellovin, M. Merritt, Augmented Encrypted Key Exchange: A Password-Based Protocol Secure against Dictionary Attacks and Password File Compromise, in *ACM Conference on Computer and Communications Security* (1993), pp. 244–250

[6] M. Boyarsky, Public-Key Cryptography and Password Protocols: The Multi-User Case, in *ACM Conference on Computer and Communications Security* (1999), pp. 63–72

[7] V. Boyko, P. MacKenzie, S. Patel, Provably Secure Password-Authenticated Key Exchange Using Diffie–Hellman, in *Advances in Cryptology—Eurocrypt 2000 Proceedings*. Lecture Notes in Computer Science, vol. 1807 (Springer, Berlin, 2000), pp. 156–171

[8] R. Canetti, Universally Composable Security: A New Paradigm for Cryptographic Protocols, in *IEEE Symposium on Foundations of Computer Science* (2001), pp. 136–145

[9] B. Chor, O. Goldreich, Unbiased Bits from Sources of Weak Randomness and Probabilistic Communication Complexity, *SIAM J. Comput.* **17**(2), 230–261 (1988)

[10] W. Diffie, M. Hellman, New Directions in Cryptography, *IEEE Trans. Inf. Theory* **22**(6), 644–654 (1976)

[11] Y. Dodis, R. Oliveira, On Extracting Private Randomness over a Public Channel. Approximation, Randomization, and Combinatorial Optimization, in *Proc. of APPROX 2003 and RANDOM 2003*. Lecture Notes in Computer Science, vol. 2764 (Springer, Berlin, 2003), pp. 252–263

[12] Y. Dodis, A. Elbaz, R. Raz, R. Oliveira, Improved Randomness Extraction from Two Independent Sources. Approximation, Randomization, and Combinatorial Optimization, in *Proc. of APPROX 2004 and RANDOM 2004*. Lecture Notes in Computer Science, vol. 3122 (Springer, Berlin, 2004)

[13] R. Gennaro, Y. Lindell, A Framework for Password-Based Authenticated Key Exchange, in *Advances in Cryptology—Eurocrypt 2003 Proceedings*. Lecture Notes in Computer Science, vol. 2656 (Springer, Berlin, 2003), pp. 524–543

[14] O. Goldreich, *Foundations of Cryptography,* vol. 2 (Cambridge University Press, Cambridge, 2004)

[15] O. Goldreich, Y. Lindell, Session-Key Generation Using Human Passwords Only, in *Advances in Cryptology—Crypto 2001 Proceedings*. Lecture Notes in Computer Science, vol. 2139 (Springer, Berlin, 2001), pp. 408–432. Full version to appear in Journal of Cryptology

[16] S. Goldwasser, S. Micali, Probabilistic Encryption, *J. Comput. Syst. Sci.* **28**(2), 270–299 (1984)

[17] S. Halevi, H. Krawczyk, Public-Key Cryptography and Password Protocols, in *ACM Conference on Computer and Communications Security* (1998), pp. 122–131

[18] J. Katz, R. Ostrovsky, M. Yung, Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords, in *Advances in Cryptology—Eurocrypt 2001 Proceedings*. Lecture Notes in Computer Science, vol. 2045 (Springer, Berlin, 2001), pp. 475–494

[19] P. MacKenzie, S. Patel, R. Swaminathan, Password Authenticated Key Exchange Based on RSA, in *ASIACRYPT* (2000), pp. 599–613

[20] S. Micali, C. Rackoff, B. Sloan, The Notion of Security for Probabilistic Cryptosystems, *SIAM J. Comput.* **17**, 412–426 (1988)

[21] M. Naor, B. Pinkas, Oblivious Transfer and Polynomial Evaluation, in *ACM Symposium on Theory of Computing* (1999), pp. 245–254

[22] M.-H. Nguyen, S. Vadhan, Simpler Session-Key Generation from Short Random Passwords, in *Proceedings of the First Theory of Cryptography Conference (TCC '04)*. Lecture Notes in Computer Science, vol. 2951 (Springer, Berlin, 2004), pp. 428–445

[23] N. Nisan, D. Zuckerman, Randomness is Linear in Space, *J. Comput. Syst. Sci.* **52**(1), 43–52 (1996)

[24] R. Richardson, J. Kilian, On the Concurrent Composition of Zero-Knowledge Proofs, in *Advances in Cryptology—Eurocrypt 99 Proceedings*. Lecture Notes in Computer Science, vol. 1592 (Springer, Berlin, 1999), pp. 415–431

[25] V. Shoup, On Formal Models for Secure Key Exchange, Cryptology ePrint Archive Report 1999/012 (1999)

[26] A. Srinivasan, D. Zuckerman, Computing with Very Weak Random Sources, *SIAM J. Comput.* **28**(4), 1453–1459 (1999)

[27] M. Steiner, G. Tsudik, M. Waidner, Refinement and Extension of Encrypted Key Exchange, *Oper. Syst. Rev.* **29**(3), 22–30 (1995)

[28] A. Yao, How to Generate and Exchange Secrets, in *IEEE Symposium on Foundations of Computer Science* (1986), pp. 162–167