Privacy Amplification with Asymptotically Optimal Entropy Loss*

Nishanth Chandran† Bhavana Kanukurthi‡ Rafail Ostrovsky§ Leonid Reyzin¶

August 30, 2012

Abstract

We study the problem of "privacy amplification": key agreement between two parties who both know a weak secret w, such as a password. (Such a setting is ubiquitous on the internet, where passwords are the most commonly used security device.) We assume that the key agreement protocol is taking place in the presence of an active computationally unbounded adversary Eve. The adversary may have partial knowledge about w, so we assume only that w has some entropy from Eve's point of view. Thus, the goal of the protocol is to convert this non-uniform secret w into a uniformly distributed string R that is fully secret from Eve. R may then be used as a key for running symmetric cryptographic protocols (such as encryption, authentication, etc.).

Because we make no computational assumptions, the entropy in R can come only from w. Thus such a protocol must minimize the entropy loss during its execution, so that R is as long as possible. The best previous results have entropy loss of $\Theta(\kappa^2)$, where κ is the security parameter, thus requiring the password to be very long even for small values of κ. In this work, we present the first protocol for information-theoretic key agreement that has entropy loss **linear** in the security parameter. The result is optimal up to constant factors. We achieve our improvement through a somewhat surprising application of error-correcting codes for the edit distance.

The protocol can be extended to provide also "information reconciliation," that is, to work even when the two parties have slightly different versions of w (for example, when biometrics are involved).

1 Introduction

The classical problem of privacy amplification, introduced by Bennett, Brassard, and Robert [1], considers the setting in which two parties, Alice and Bob, start out knowing a common string w that is partially

*An abstract of this paper appeared in the Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010 [6]; this paper is an expanded and corrected version of the conference paper.

†Microsoft Research, Redmond, Email: `nish@microsoft.com`. Research done in part while the author was at UCLA and was supported by NSF grants 0716835, 0716389, 0830803 and 0916574.

‡Department of Computer Science, UCLA, Email: `bhavanak@cs.bu.edu`. Research done in part while the author was at Boston University, and was supported by National Science Foundation grants CNS-0831281 and CNS-0546614, and in in part while at UCLA and was supported by NSF grants CCF-0916574; IIS-1065276; CCF-1016540; CNS-1118126; CNS-1136174; and Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0392. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

§Department of Computer Science and Mathematics, UCLA, Email: `rafail@cs.ucla.edu`. Research supported in part by NSF grants CNS-0830803; CCF-0916574; IIS-1065276; CCF-1016540; CNS-1118126; CNS-1136174; US-Israel BSF grant 2008411, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. This material is also based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0392. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

¶Department of Computer Science, Boston University, URL: `http://www.cs.bu.edu/fac/reyzin`. Research supported in part by part by National Science Foundation grants 0831281, 0546614, 1012910, and 1012798.

secret. Following [2], we make no assumption on the distribution of w beyond a lower bound on its entropy. The goal of Alice and Bob is to perform key agreement: to agree on a string R that is fully secret. Formally, R should be statistically close to uniform from the point of view of the adversary Eve (we will let $2^{-\kappa}$ be the statistical distance between the distribution of R and the uniform distribution, and call κ the *security parameter*). Such R can then be used as a key for symmetric cryptography.

We make no computational assumptions and therefore model Eve as being computationally unbounded. This requirement implies that all the entropy in R comes from w. Thus, one of the most important requirements of such a protocol is that the output length of R must be as long as possible given the entropy of w. The difference between the entropy of w and the length of R is called the *entropy loss*.

We can model the communication channel between Alice and Bob as authenticated or unsecured. Equivalently, we can model Eve as passive or active. If Eve is passive (i.e., merely listens in on the messages between Alice and Bob but cannot modify them), strong extractors [22] provide an optimal solution to this problem. Alice simply sends an extractor seed to Bob, they both extract R from w using the seed, and the entropy loss can be as low as 2κ.

However, if Eve is active (i.e., can modify the messages), the problem becomes considerably harder. In particular, one needs to worry not only about achieving R that is $2^{-\kappa}$-close to uniform, but also about ensuring that Alice and Bob output *the same* R with probability at least $1 - 2^{-\kappa}$. While the specific focus of our paper is to obtain R of maximum length, there are other parameters that one might wish to optimize. For instance, one might wish to minimize the round complexity or to minimize the entropy required in w for the protocol to work. Renner and Wolf [24], building on a series of works by Maurer, Renner and Wolf [19, 20, 29, 21], presented the first protocol that is secure against active Eve and works even when w is less than half-entropic (i.e., $h_W \leq \lambda_w/2$ where h_W and λ_w denote the entropy and the length of w). Moreover, by the use of extractors with asymptotically optimal seed length [12] and through the analysis of Kanukurthi and Reyzin [15], it can be seen that the protocol of [24] achieves entropy loss of $\Theta(\kappa^2)$ and takes $\Theta(\kappa)$ rounds of communication. (The work of [15], which builds upon [24], achieves the same asymptotic parameters but considerably improves the constants hidden inside Θ by eliminating the need for complicated extractors.) Dodis and Wichs [11] reduce the number of messages in the protocol from $\Theta(k)$ to just 2, but do not improve the entropy loss. They also present a two-message non-constructive (in other words, non-polynomial-time) protocol with an entropy loss of $\Theta(\kappa)$. (They achieve this result by introducing a primitive called non-malleable extractors and showing nonconstructively that they exist.) While this result shows the theoretical feasibility of achieving such a low entropy loss, the only efficient solution that matches this entropy loss relies on Random Oracles [4]—i.e., an "assumption" that a publicly known truly random function is available. (It should be noted that single-message protocols for the same problem exist [20, 8]; however, they have entropy loss $\lambda_w - h_W + \Theta(\kappa)$ and require thus require w to be at least half-entropic.)

Achieving an *efficient* privacy amplification protocol with entropy loss $\Theta(\kappa)$ and without resorting to the Random Oracle Model has, until now, been an open question.

Our Contribution. We construct the first efficient protocol for privacy amplification over unsecured channels whose entropy loss (and number of rounds) is linear in the security parameter κ. This security loss is optimal up to constant factors, because extractor bounds require entropy loss at least $2\kappa - O(1)$ even in the case of authenticated channels [23]. We thus demonstrate that, up to constant factors, privacy amplification over unsecured channels can be as entropy-efficient as over authenticated ones.

Extension to Information Reconciliation/Fuzzy Extractors. Consider the following generalization: Alice starts out with w, but Bob starts out with w' that is close to w in some metric space. Their goal is still the same: to agree on the same uniform string R. This problem is known as privacy amplification with information reconciliation [1] or as fuzzy extractors [10]. Constructions secure against active

Eve appeared in [25, 8, 15, 11].

This setting arises, for example, when Alice and Bob have access to a (possibly) noisy channel that can be partially eavesdropped by Eve; or when a trusted server (Alice) stores the biometric of a user (Bob), and the user subsequently uses his fresh biometric reading to authenticate himself to the server; or when Alice and Bob are mobile nodes wanting to authenticate each other based on the fact that their knowledge of a location is greater than Eve's (e.g., if they are much closer to a particular location than Eve, and thus are able to observe it at higher resolution).

Using the same approach as in [15], our protocol extends to this setting, as well.

A Related Problem. Ishai, Kushilevitz, Ostrovsky, and Sahai [14] consider the problem of constructing a two-party protocol that extracts m "clean" instances of a joint distribution (X, Y) from $\mathcal{O}(m)$ "dirty" instances. This task can be viewed as a generalization of randomness extraction (the special case is when X and Y are identical bits). However, the techniques of [14] do not directly apply to the case when we have only *one* instance of a distribution. Furthermore, the entropy loss achieved in their work is significantly greater (constant factor times m) than in our case, where we obtain entropy loss independent of the entropy or length of the secret and linear in the security parameter.

Construction Techniques. Our starting point is the protocol for interactive authentication from [15], which generalizes the authentication protocol of Renner and Wolf [24]. We focus only on the case when Alice and Bob share the same secret w; i.e., when $w = w'$ (as mentioned, the more general case can be addressed in the exact same way as in [15]). Using known techniques from the works of [24] and [15], it suffices to construct a message authentication protocol that can authentically transfer a message m from Alice to Bob using w.

The authentication protocol of [24, 15] works by authenticating bits of m one by one. For each bit of m, Bob sends Alice a random extractor seed, and, if the bit is equal to 1, Alice responds with the output of the extractor on input w using Bob's seed. Alice also sends Bob a random extractor seed of her own, to which he always responds (applying the extractor to w using Alice's seed), regardless of the bit of m. Each extractor output is κ bits long. This results in $\Theta(\kappa)$ entropy loss for every bit authenticated, and $\Theta(\kappa^2)$ entropy loss overall, because the message being authenticated needs to be $\Theta(\kappa)$ bits long (it is, actually, a MAC key in the protocol of [15]). The security proof shows that to succeed in breaking such an authentication protocol, the active adversary Eve must respond to *at least one* fresh random challenge on her own without interacting with either Alice or Bob. Because the extractor output is a nearly-uniform κ-bit string, Eve cannot succeed with probability much higher than $2^{-\kappa}$.

The high level intuition for our protocol is as follows. If we were to shorten the length of the extractor output (in the authentication protocol) to be a constant number of bits, then we only lose $\Theta(1)$ bits of entropy for every bit of m and obtain an $\Theta(\kappa)$ entropy loss overall. On the other hand, the success probability of an adversary is a constant (by the same proof as before). If we could instead now ensure that Eve must respond to *several* (namely, $\Theta(\kappa)$) fresh random challenges on her own, then we could show that the success probability is $2^{-\kappa}$. This can be done by encoding m in a special error-detecting code of distance $\Theta(\kappa)$ and ensuring that to introduce $\Theta(\kappa)$ errors required to avoid detection, Eve must come up with $\Theta(\kappa)$ extractor outputs on her own.

It turns out that the code we need is a code for the edit distance [16] (the distance that counts the number of single-character insertions and deletions required to convert one string into another), for the following reasons. We observe that, since the authentication of m is done bit-by-bit, Eve can change m by inserting individual bits, deleting them, or changing them from 0 to 1 or from 1 to 0. Deletions, insertions of 1, and changes from 0 to 1 require Eve to guess an extractor output on a fresh random seed. Thus, Eve can create edit errors in the message, but at least some types of the edit errors (namely, insertions of 1 and deletions) require her to find an extractor output on her own. Because in the context of [24, 15] the length of m and the number of 1s in it are known a priori to both Alice and Bob, insertions of 0 and changes of 1 to 0 must be accompanied by deletions and changes of 0 to 1; in fact, we show that, no matter how Eve chooses to modify the string, at least a quarter of the operations require her to find an extractor output on her own. So, if instead of authenticating the bits of the m, Alice first encoded m in an error-detecting code for 4κ edit errors, Eve would have to respond to at least κ fresh random challenges on her own. Of course, the length of the codeword must still remain linear in the length of m. The codes of Schulman and Zuckerman [26] have this property (though we need to modify them to ensure the number of 1s is the same for every codeword of a given length).

Proof Techniques. While using an error-detecting code and shortening the extractor outputs intuitively may seem to work in a straightforward manner, the proofs turn out to be quite tricky. In particular, we will need to use a proof technique that is completely different from the one used in [24] and [15], for the following reason. In the authentication protocols of [24, 15], Alice authenticates bits of the message one at a time. The proofs make use of induction on the length of the message received so far by Bob to show that if Eve was successful in changing any bit(s) of the message, then Eve must have responded to *one* fresh random challenge on her own. We, on the other hand, cannot use such an induction argument since we need to precisely characterize *how many* fresh random challenges Eve must have responded in relation to the number of bits modified. Instead, we use a new proof technique wherein we view the entire protocol transcript from the point of view of Eve as a string of literals, where each literal represent an interaction either with Alice or Bob. Using combinatorial arguments, we show that Eve cannot interleave these literals to her advantage without having to respond to many fresh random challenges.

Next, one might like to claim that if Eve were to respond with multiple extractor outputs for random, independent seeds, then her success in giving the right response for each of the seeds is also independent of her success for other seeds. Unfortunately, this intuition does not quite hold and there are subtleties in proving the theorem in this manner. We could try to use average min-entropy (introduced in [10]), which gives us a guarantee that on average the entropy in w does not get reduced by too much, and implies the desired bound on the probability that Eve is successful in a single fresh random challenge. However, if the min-entropy in a particular run is below average, then Eve has an easier task for *all* fresh random random challenge simultaneously. In an extreme case, here may be a particular bad run (which occurs with very small probability) in which all information about w is revealed. This, in turn, destroys all independence in the events of Eve's success for each challenge. To counter this, we take the approach of considering two separate cases — the case when the run does not reveal too much about w and where we can argue "sufficient independence" (which happens with high probability), and the case when the run might reveal too much about w (perhaps all of it) and we cannot argue independence (which happens with low probability). Now, if we make an assumption that w begins with $\Theta(\kappa)$ bits of entropy more than what would be needed otherwise, we can show that the probability with which independence does not hold is low enough for our theorem to be true.

Organization of the paper. We introduce notation and define our security model in Section 2. In Section 3, we briefly describe some of the existing tools that we require for our construction. Our main construction is given in Section 4. We give the proof of our main theorem in 5.

2 Preliminaries

Notation. Let U_l denote the uniform distribution on $\{0,1\}^l$. Let X_1, X_2 be two probability distributions over some set S. Their *statistical distance* is

$$\mathbf{SD}\left(X_1, X_2\right) \stackrel{\text{def}}{=} \max_{T \subseteq S}\{\Pr[X_1 \in T] - \Pr[X_2 \in T]\} = \frac{1}{2}\sum_{s\in S}\left|\Pr_{X_1}[s] - \Pr_{X_2}[s]\right|$$

(they are said to be ε-close if $\mathbf{SD}(X_1, X_2) \le \varepsilon$). The *min-entropy* of a random variable W is $\mathbf{H}_\infty(W) = -\log(\max_w \Pr[W = w])$. For a joint distribution (W, E), define the (average) conditional min-entropy of W given E [10] as

$$\widetilde{\mathbf{H}}_\infty(W \mid E) = -\log(\mathop{\mathbf{E}}_{e \leftarrow E}(2^{-\mathbf{H}_\infty(W|E=e)}))$$

(here the expectation is taken over e for which $\Pr[E = e]$ is nonzero). A computationally unbounded adversary who receives the value of E cannot give the correct value of W (in a single attempt) with probability greater than $2^{-\widetilde{\mathbf{H}}_\infty(W|E)}$. Throughout this paper, for any string x, we use the notation λ_x to denote its length and h_x to denote its entropy (i.e, $\mathbf{H}_\infty(X)$). We make use of the following lemma [10, Lemma 2.2b], which states that the average min-entropy of a variable from the point of view of an adversary does not decrease by more than the number of bits (correlated with the variable) observed by the adversary.

Lemma 1. *If B has at most 2^λ possible values, then $\widetilde{\mathbf{H}}_\infty(A \mid B) \ge \mathbf{H}_\infty(A, B) - \lambda \ge \mathbf{H}_\infty(A) - \lambda$. and, more generally, $\widetilde{\mathbf{H}}_\infty(A \mid B, C) \ge \widetilde{\mathbf{H}}_\infty(A, B \mid C) - \lambda \ge \widetilde{\mathbf{H}}_\infty(A \mid C) - \lambda$*

We also use the following lemma [10, Lemma 2.2a], which says that min-entropy is t bits less than the average min-entropy with probability at most 2^{-t}.

Lemma 2. *For any $\delta > 0$, the conditional entropy $\mathbf{H}_\infty(A \mid B = b)$ is at least $\widetilde{\mathbf{H}}_\infty(A \mid B) - \log(\frac{1}{\delta})$ with probability at least $1 - \delta$ over the choice of b.*

Model and Security definition. Let $w \in \{0,1\}^n$ be chosen according to distribution W be the secret value held by Alice and Bob respectively. Let Protocol (A, B) be executed in the presence of an active adversary Eve. Let $\mathit{Received}_a, \mathit{Sent}_a, \mathit{Received}_b, \mathit{Sent}_b$ be the random variables describing the messages received by Alice, sent by Alice, received by Bob, and sent by Bob, respectively, when (A, B) is executed in the presence of Eve. (We will use $\mathit{received}_a, \mathit{sent}_a, \mathit{received}_b, \mathit{sent}_b$ to denote specific values of these variables.) Because Eve is computationally unbounded, we can assume she is deterministic. We assume that if either party aborts, Eve is immediately informed of it (we can think of it as being included in *Sent*). We denote the private coins of Alice and Bob by r_a and r_b respectively.

In a Privacy Amplification Protocol, the outputs of Alice and *bob* will be denoted by $\mathsf{k_A} = \mathsf{A}(w, \mathit{received}_a, r_a)$ and $\mathsf{k_B} = \mathsf{B}(w, \mathit{received}_b, r_b)$. Each of $\mathsf{k_A}$ and $\mathsf{k_B}$ can be either a key or a special symbol $\perp$, signifying that the party has detected an active attack and is aborting the protocol.

Before formally defining security, let us explain the reasoning behind the definition. We would like for the outputs of Alice and Bob to match. However, this goal is unachievable in general [19]: Eve could simply interfere with the last message of the protocol to force the recipient of that message to reject while the sender will accept (this is assuming the last message influences the recipient's accept/reject decision; if not, the same argument could be made using the last message that does). Instead, we will make a weaker requirement: with high probability, $\mathsf{k_A} = \mathsf{k_B}$ or $\mathsf{k_A} = \perp$ or $\mathsf{k_B} = \perp$ (a more strict, asymmetric definition is also possible: it could require, for example, that $\mathsf{k_A} = \mathsf{k_B}$ or $\mathsf{k_B} = \perp$, ensuring that if Alice detects a problem, then Bob does, too; it may make sense if Alice makes her accept/reject decision before Bob).

We would also like for the keys output by Alice and Bob to be (nearly) random. Let us focus on Alice's key $\mathsf{k_A}$. We could ask for distribution of $\mathsf{k_A} \mid \mathsf{k_A} \ne \perp$ to be close to uniform, but that definition seems unachievable, because Eve could disrupt the protocol unless she is confident that $\mathsf{k_A}$ is in a small set of possible values, in which case $\mathsf{k_A} \mid \mathsf{k_A} \ne \perp$ is far from uniform, simply because $\mathsf{k_A} \ne \perp$ is very rare and $\mathsf{k_A}$ is constrained in such a case.[1] The problem comes from trying to condition on rare events; indeed, if $\mathsf{k_A} \ne \perp$

[1] As was pointed out to us by Gil Segev and Yevgeniy Dodis, the definitions in [11, 15, 6, 7] suffer from this problem and the protocols in those works probably do not satisfy their own definitions for this trivial reason, even though the protocols are secure with respect to a more appropriate definition. The definiton of [19] and its variants used in subsequent works, such as [20, 29, 21], requires that, with high probability, either $\mathsf{k_A}$ has high entropy or $\mathsf{k_A} = \perp$; Maurer [19] states that an

is very rare, then we shouldn't care about the marginal distribution of $\mathsf{k_A}$ in this rare case. This problem could be solved by requiring that the product of $\Pr[\mathsf{k_A} \neq \perp]$ and distance of $\mathsf{k_A} \mid \mathsf{k_A} \neq \perp$ from uniform be small. For aesthetic reasons, we prefer to solve it differently: we follow the approach of [9, Section 6], and compare the distribution of $\mathsf{k_A}$ to the distribution that is $\perp$ with the same probability as $\Pr[\mathsf{k_A} =\perp]$, and is uniform otherwise. We note that this problem does not come up in defining security for single-message protocols, e.g., [8], because Alice always outputs $\mathsf{k_A} \neq\perp$.

Definition 1. *An interactive protocol* (A, B) *played by Alice and Bob on a communication channel fully controlled by a computationally unbounded adversary Eve, is a* $(h_W, \lambda_k, \delta, \epsilon)$-privacy amplification protocol *if it satisfies the following properties whenever* $\mathbf{H}_\infty(W) \geq h_W$*:*

1. *Correctness. If Eve is passive, then* $\Pr[\mathsf{k_A} = \mathsf{k_B}] = 1$.
2. *Robustness. For any Eve, the probability that the following experiment outputs "Eve wins" is at most* $2^{-\delta}$*: sample* w *from* W*; let* $received_a, received_b$ *be the messages received by Alice and Bob upon execution of* (A, B) *with Eve actively controlling the channel, and let* $\mathsf{k_A} = \mathsf{A}(w, received_a, r_a), \mathsf{k_B} = \mathsf{B}(w, received_b, r_b)$*. Output "Eve wins" if* $\mathsf{k_A} \neq \mathsf{k_B} \wedge \mathsf{k_A} \neq \perp \wedge \mathsf{k_B} \neq \perp$*. (Note that this is* pre-application *robustness in the terminology of [8, 9]; we do not address* post-application *robustness here.)*
3. *Extraction. Define* $\mathsf{purify}(r)$ *to be a randomized function whose input is either a binary string or* $\perp$*. If* $r = \perp$*, then* $\mathsf{purify}(r) = \perp$*; else,* $\mathsf{purify}(r)$ *is a uniformly chosen random string of lentgh* λ_k*. Note that the pair* $Sent = (Sent_a, Sent_b)$ *contains an active Eve's view of the protocol. We require that for any Eve,*

$$sd(\mathsf{k_A}, Sent), (\mathsf{purify}(\mathsf{k_A}), Sent) \leq \epsilon \quad and \quad \mathbf{SD}\left((\mathsf{k_B}, Sent), (\mathsf{purify}(\mathsf{k_B}), Sent)\right) \leq \epsilon\,.$$

The quantity $h_W - \lambda_k$ *is the known as the* entropy loss *of the protocol.*

We can allow Eve to additionally have some information on the secret w*, sampled as a random variable* E *that is correlated with* W*. We avoided introducing it above to simplify notation; to add it in, require that robustness holds even when Eve is given this additional information, and include* E *into* $Sent$ *for defining extraction. Of course, the definition is required to hold only when* $\widetilde{\mathbf{H}}_\infty(W \mid E) \geq h_W$*.*

An important building block that we will construct is an interactive authentication protocol. In an authentication protocol, both Alice and Bob have the same input w as in the privacy amplification protocol, and Alice additionally takes as input a message m that she wants to send to Bob authentically. The goal is merely to transmit this message, not to agree on a key. We let m_a denote the message that Alice is trying to send and m_b denote the message that Bob receives. Each party has an output, denoted by $\mathsf{t_A}$ and $\mathsf{t_B}$, that is "accept" or "reject"; additionally, Bob outputs m_b. We now present the formal definition. (The definition we use is just an interactive variant of one-time message authentication codes. See [15, Definition 4] for one such definition.)

Definition 2. *An interactive protocol* (A, B) *played by Alice and Bob on a communication channel fully controlled by an adversary Eve, is a* (h_W, κ)-interactive authentication protocol *if it satisfies the following properties whenever* $\mathbf{H}_\infty(W) \geq h_W$*:*

1. *Correctness. If Eve is passive, then* $\Pr[m_a = m_b] = 1$.
2. *Robustness. For any Eve, the probability that the following experiment outputs "Eve wins" is at most* $2^{-\kappa}$*: sample* w *from* W*; let* $received_a, received_b$ *be the messages received by Alice and Bob upon execution of* (A, B) *with Eve actively controlling the channel, and let* $\mathsf{A}(w, received_a, r_a, m_a) = \mathsf{t_A}, \mathsf{B}(w, received_b, r_b) = (m_b, \mathsf{t_B})$*. Output "Eve wins" if* $(m_b \neq m_a \wedge \mathsf{t_B} = \text{"}accept\text{"})$*.*

alternative definition could be provided "in terms of any reasonable constraint on the deviation of a distribution from the uniform distribution" without specifying how.

The minimum, over all possible Eves, of the quantity $h_W - \widetilde{\mathbf{H}}_\infty(W \mid Sent_a, Sent_b, \mathsf{t_A}, \mathsf{t_B})$ is known as the entropy loss *of the protocol.*

Similarly to Definition 1, in this definition we can also allow Eve to have some information on the secret w by simply giving this information as input to Eve.

We note that in this definition, asymmetry is essential: it is not enough that Alice outputs "reject" when $m_a \neq m_b$, because if Bob doesn't detect that he got the wrong message, he may mistakenly rely on it in subsequent steps (for example, if the message is an extractor seed, and Eve manages to tamper with it without being detected by Bob, then Bob may extract a key that has no secrecy, and Eve will be able to read any messages he encrypts with that key). In contrast, since Alice is the one sending m_a, she knows the correct message even if Eve tampers with the protocol and is not detected by Alice.

For the sake of simplicity, we might omit $\mathsf{t_A}, \mathsf{t_B}$ from the outputs of Alice and Bob when it is clear from context what values they take (generally, "accept" if all checks in the protocol are satisfied, and "reject" otherwise).

3 Building Blocks

We begin by presenting the building blocks needed for our main construction of a privacy amplification protocol with optimal entropy loss.

3.1 Extractors

Extractors [22] yield a close-to-uniform string from a random variable with high min-entropy, using a uniformly random seed i as a kind of catalyst. Strong extractors are ones in which the extracted string looks random even in the presence of the seed. We will use only strong extractors in this paper and thus sometimes omit the adjective "strong."

Definition 3. *Let $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}^l$ be a polynomial time probabilistic function that uses r bits of randomness. We say that Ext is an (n, t, l, ε)-*strong extractor *if for all random variables W over $\{0,1\}^n$ such that $\mathbf{H}_\infty(W) \geq t$, we have $\mathbf{SD}\left((\mathsf{Ext}(W; X), X), (U_l, X)\right) \leq \varepsilon$, where X is the uniform distribution over $\{0,1\}^r$.*

Universal hash functions are perhaps the simplest extractors, allowing $t = \ell - 2 + 2\log\frac{1}{\varepsilon}$ (see [27, Theorem 8.1], [13, Lemma 4.8], and references therein).

If an extractor works when the guarantee on W is for conditional min-entropy rather than min-entropy, it is called an *average-case* extractor. This notation was introduced in [10, Section 2.5]. Vadhan [28, Problem 6.8] showed that all extractors are average-case extractors with a slight loss of parameters: namely, any (t, ε)-extractor for $t \leq n-1$ is also a $(t, 3\varepsilon)$-average-case extractor. Some extractors—namely, universal hash function [5]—don't lose parameters at all in the average case [10, Section 2.5] (in fact, almost universal hash functions [27] work as well [8]).

For the purposes of this work, we would like to argue that extractor outputs are hard to predict for an adversary. We first recall the following lemma [15, Lemma 1] on the average entropy of extractor outputs.

Lemma 3. *Let Ext be a $(n, t, \ell, \varepsilon)$-strong extractor and W be a random variable over $\{0,1\}^n$ with $\mathbf{H}_\infty(W) \geq t$. Then $\widetilde{\mathbf{H}}_\infty(\mathsf{Ext}(W; X) \mid X) \geq \min\left(l, \log\frac{1}{\varepsilon}\right) - 1$. More generally, if Ext is an* average-case *$(n, t, \ell, \varepsilon)$-strong extractor and $\widetilde{\mathbf{H}}_\infty(W \mid E) \geq t$, then $\widetilde{\mathbf{H}}_\infty(\mathsf{Ext}(W, X) \mid X, E) \geq \min\left(\ell, \log\frac{1}{\varepsilon}\right) - 1$.*

The above lemma says that (assuming $\log\frac{1}{\varepsilon} > \ell$) an $\ell-$bit extractor output has an entropy of $\ell - 1$ bits on average, over all seeds; this implies that it cannot be predicted with probability higher than $2^{\ell-1}$. In this work, we will sometimes require a stronger guarantee. We will require that the probability, for a random seed, that an extractor output has entropy less than $\ell - 1$ is low. If we tried to use the bound on the average entropy of $\mathsf{Ext}(W;X) \mid X$ to directly obtain a bound on the real entropy of $\mathsf{Ext}(W;X)|X = x$ (for instance, by using the Markov's inequality), we get that $\Pr_x[\mathbf{H}_\infty(\mathsf{Ext}(W;X)) \le \ell - \delta] \le 2^{-\delta}$. This quantity is too high when δ is low. We show in the next lemma that we can obtain a tighter bound for the case when the extractor output is short.

Lemma 4. *Let* Ext *be a* $(n, t, \ell, \varepsilon)$*-strong extractor and* W *be a random variable over* $\{0,1\}^n$ *with* $\mathbf{H}_\infty(W) \ge t$*. Then* $\Pr_x[\mathbf{H}_\infty(\mathsf{Ext}(W;x)) \le \ell - 1] \le 2^\ell \varepsilon$ *(note that* x *is fixed inside the probability statement, so* $\mathbf{H}_\infty(\mathsf{Ext}(W;x)) = \mathbf{H}_\infty(\mathsf{Ext}(W;x) \mid x)$*).*

Proof. For each x let s_x denote the extractor output that maximizes $\Pr[\mathsf{Ext}(W,x) = s]$. Define the set of $\mathsf{Bad} \stackrel{\text{def}}{=} \{x : \Pr_x[\mathsf{Ext}(W,x) = s_x] \ge 2^{-\ell+1}\}$. Let $T = \{(s_x, x)\}$ where $x \in \mathsf{Bad}$. Our goal is determine the maximum probability that $x \in \mathsf{Bad}$.

The first line below follows from the definition an extractor; the rest follows from it by simple probability manipulation and the definition of Bad:

$$\begin{aligned}
\Pr[(\mathsf{Ext}(W,X),X) \in T] &\le \Pr[(U_l, X) \in T] + \varepsilon \\
\sum_{x \in \mathsf{Bad}} \Pr[\mathsf{Ext}(W,X) = s_x, X = x] &\le \sum_{x \in \mathsf{Bad}} \Pr[U_l = s_x, X = x] + \varepsilon \\
\sum_{x \in \mathsf{Bad}} \Pr[\mathsf{Ext}(W,X) = s_x \mid X = x] \Pr[X = x] &\le \sum_{x \in \mathsf{Bad}} \Pr[U_l = s_x \mid X = x] \Pr[X = x] + \varepsilon \\
\sum_{x \in \mathsf{Bad}} 2^{-\ell+1} \Pr[X = x] &\le \sum_{x \in \mathsf{Bad}} 2^{-\ell} \Pr[X = x] + \varepsilon \\
2^{-\ell+1} \Pr[x \in \mathsf{Bad}] &\le 2^{-\ell} \Pr[x \in \mathsf{Bad}] + \varepsilon \\
\Pr[x \in \mathsf{Bad}] &\le 2^\ell \varepsilon.
\end{aligned}$$

It then follows from here that,

$$\Pr_x[\mathbf{H}_\infty(\mathsf{Ext}(W;x)) \le \ell - 1] = \Pr_x[x \in \mathsf{Bad}] \le 2^\ell \varepsilon$$

□

3.2 Edit distance codes

Codes for insertion and deletion errors were first considered in the work of Levenshtein [16]. The first polynomial-time encodable and decodable codes that have constant rate and can correct a constant fraction of errors were given by Schulman and Zuckerman in [26]. For our application, we only require the code to be polynomial-time encodable and not necessarily polynomial-time decodable (this will be sufficient to get polynomial-time error detection when m sent in the clear from Alice to Bob, which is all we need). We also add a requirement that all codewords have the same Hamming weight. Let m be a message of length λ_m. For any two strings c and c' of length λ_c, let $\mathsf{EditDis}(c, c')$ denote the edit distance between c and c'; i.e., the number of single-bit insert and delete operations required to change string c to c' is $\mathsf{EditDis}(c, c')$.

Definition 4. *A function* $\mathsf{Edit}(\cdot) : \{0,1\}^{\lambda_m} \to \{0,1\}^{\lambda_c}$*, is a* (λ_m, e_A, ρ)-error-detecting code *for edit errors if it satisfies the following properties:*

- $c = \mathsf{Edit}(m)$ *can be computed in polynomial (in* λ_m*) time, given* m*, for all* $m \in \{0,1\}^{\lambda_m}$*;*
- *For any* $m, m' \in \{0,1\}^{\lambda_m}$ *with* $m \ne m'$, $\mathsf{EditDis}(c', c) > e_A \lambda_c$*, where* $c = \mathsf{Edit}(m)$ *and* $c' = \mathsf{Edit}(m')$*.*
- *For any* $m, m' \in \{0,1\}^{\lambda_m}$*, Hamming weight of (i.e., the number of ones in)* $\mathsf{Edit}(m)$ *equals the Hamming weight of* $\mathsf{Edit}(m')$*.*

$\rho = \frac{\lambda_m}{\lambda_c}$ *is called the* rate *and* e_A *is called the* relative distance *of the code.*

Schulman and Zuckerman [26] proved the following:

Theorem 1. *There exist constants* e_A, ρ*, such that for any* λ_m*, there exists a* (λ_m, e_A, ρ)*-error-detecting code for edit errors.*

Remark. Schulman and Zuckerman did not have the requirement of fixed Hammingn weight, however, such codes can be constructed easily[2] using the same techniques and keeping the rate and the relative distance constant.

3.3 Interactive authentication protocol

We use the interactive authentication protocol from [15] that Alice and Bob run in order to send bits authentically to each other. Let the security parameter be κ. Let Alice and Bob share an n-bit secret $w \in W$ with min-entropy h_W. Let the message that Alice wishes to authenticate be $m = m_1 \dots m_{\lambda_m}$. Assume that Bob knows λ_m and the number of ones in m (say $\mathsf{wt}(m)$). Let Ext be a $(\lambda_w, t, \kappa + 1, 2^{-\kappa-1})$-strong extractor with seed length q bits. That is, Ext takes seeds of input q, outputs $\kappa + 1$-bit strings that are $2^{-\kappa-1}$-close to uniform, as long as the input has sufficient entropy t. (In particular, $t \geq 3\kappa + 1$ is sufficient if Ext is a universal hash function.) The authentication protocol from [15], which is a modification of the scheme from [24], is presented below. Note that this protocol is simply an interleaving of two processes: Alice is responding to λ_m random challenges from Bob, one per message bit, with her response varying depending on the value of the bit. Bob is responding to $\lambda_m + 1$ challenges from Alice, the same way each time.

Protocol $\mathsf{Auth}(w, m)$**:**

1. Alice sends Bob a fresh random challenge $x_1 \in_r \{0,1\}^q$.
2. Bob receives x_1, sends $r_{x_1} = \mathsf{Ext}(w; x_1)$, and a fresh random challenge $y_1 \in_r \{0,1\}^q$ to Alice.
3. For $i = 1$ to λ_m :
   - Alice receives r_{x_i}, y_i. If $r_{x_i} \neq \mathsf{Ext}(w; x_i)$, she aborts. Otherwise, if $m_i = 1$, she sends to Bob $(1, r_{y_i} = \mathsf{Ext}(w; y_i))$, and if $m_i = 0$, she sends to Bob $(0, \perp)$. She also sends to Bob a fresh random challenge $x_{i+1} \in_r \{0,1\}^q$.
   - Bob receives m_i, r_{y_i}, x_{i+1}. If $m_i = 1$ and $r_{y_i} \neq \mathsf{Ext}(w; y_i)$, he aborts. If $i = \lambda_m$, he sets $m_b = m_1 \dots m_{\lambda_m}$ and aborts unless the number of 1 bits in m_b matches what he knows $\mathsf{wt}(m)$ should be. He sends $r_{x_{i+1}} = \mathsf{Ext}(w; x_{i+1})$ to Alice. If $i < \lambda_m$, he also sends a fresh random challenge $y_{i+1} \in_r \{0,1\}^q$ to Alice. If $i = \lambda_m$, he outputs $\mathsf{t_B} =$"accept" and m_b.
4. Alice receives $r_{x_{\lambda_m+1}}$. If $r_{x_{\lambda_m}+1} \neq \mathsf{Ext}(w; x_{\lambda_m+1})$, she aborts. Else, she outputs $\mathsf{t_A} =$"accept".

(In the protocol above, "Alice/Bob aborts" means she/he sets $\mathsf{t_A}/\mathsf{t_B}$ to "reject".)

We need the following definition before we can describe the intuition behind the security of this authentication scheme.

Definition 5. *We say that Eve responded to Alice(Bob) with a fresh random challenge if*

[2] Schulman and Zuckerman [26] build a two-layered code by greedily searching a logarithmic-size space for a small code for edit errors, concatenating those small codewords and then interleaving them with fixed patterns of 0s and 1s. By changing the greedy search to focus only on the subset of strings whose Hamming weight is half their length, one can get the required construction.

- *Eve received a fresh random challenge x from Alice(Bob) at some time instance t.*
- *At some time instance, $t+i$, Eve sent Alice(Bob) r such that $r = \mathsf{Ext}(w;x)$.*
- *Between the time instance t and $t+i$, Eve did not receive $\mathsf{Ext}(w;x')$ for any x' from either Alice or Bob.*

In other words, we say that Eve *responded to a fresh random challenge* sent by Alice (Bob), if Alice (Bob) sends a challenge and Eve responds to it without having received a response to an extractor challenge (on any, not necessarily the same, seed) in the meantime. (Note that if Eve sends $\perp$ in response to Bob's challenge, thereby authenticating a zero bit, it is not considered as a response to a fresh random challenge.) The intuition for the security of the above protocol is as follows. Note that Eve can insert a 0 bit (this does not require a response from Alice) and can change a 1 bit to a 0 bit. Since Bob knows λ_m as well as $\mathsf{wt}(m)$, if Eve were to insert 0 bits or change a 1 bit to a 0, then she must also either remove 0 bits or insert 1 bits. Now, removing 0 bits sent by Alice or inserting 1 bits require responding to a random challenge of Alice or Bob. By Lemma 3, we have that the r_{x_i} and r_{y_i} values have entropy κ from Eve's point of view. Since the responses (r_{x_i} and r_{y_i}) have entropy κ bits, Eve cannot respond to a fresh random challenge by Alice or Bob with probability $> 2^{-\kappa}$. Hence, the probability of Eve's success can be shown to be at most $2^{-\kappa}$.

To analyze the entropy loss in the case of passive Eve (an active Eve can't learn too much more), we note that Bob reveals $\kappa+1$ bits of information correlated to w for each of his λ_m+1 messages to Alice, while Alice reveals $\kappa+1$ bits of information correlated to w for each of her $\mathsf{wt}(m)$ messages to Bob in which $m_i = 1$. Therefore, the entropy loss of the authentication protocol is $(\kappa+1)(\lambda_m + \mathsf{wt}(m)+1) = \Theta(\lambda_m\kappa)$ by Lemma 1. Specifically, we apply Lemma 1 where A equals W and B equals all the bits sent by Alice and Bob; the uniformly random extractor seeds sent by Alice and Bob increase $\mathbf{H}_\infty(A,B)$ and decrease λ by the same amount, and thus cancel out.

This entropy loss analysis shows that as long as $\mathbf{H}_\infty(W) \geq t + (\kappa+1)(\lambda_m + \mathsf{wt}(m)+1)$, (where t is the threshold entropy that is needed by the specific extractor used for the extraction to be secure), the entropy in w is sufficient for the extractor to work until the last round, ensuring that the protocol is secure (this analysis requires an average-case extractor; our analysis later will not have this requirement).

The entropy loss of $\Theta(\lambda_m\kappa)$ in the authentication protocol translates to an entropy loss of $\Theta(\kappa^2)$ for the privacy amplification protocol (because the privacy amplification protocol of [15] needs to authenticate a message—namely, a MAC key—of length $\Theta(\kappa)$).

4 Main Construction

Our main construction of a privacy amplification protocol is obtained by building an improved authentication protocol with low entropy loss. In particular, our main theorem is:

Theorem 2. *Let κ denote the security parameter. Let $\mathsf{Edit}(\cdot)$ be a (λ_m, e_A, ρ)-error-detecting code for constants $0 < e_A, \rho < 1$; let $\lambda_c = \lambda_m/\rho$. Set $\tau = \left\lceil \frac{4(\kappa+2)}{\lambda_c e_A} + 1 \right\rceil$. Let Ext be a $(\lambda_W, t, \tau, 2^{-2\kappa})$-strong extractor with seed length q bits, and Ext' be an average-case $(\lambda_W, t, \kappa+\tau+\lambda_c+4, 2^{-(\kappa+\tau+\lambda_c+4)})$-strong extractor with seed length q' bits, for some t.*

Then there exists an efficient (h_W, κ)-interactive authentication protocol for messages of length λ_m with entropy loss $\lambda_c(2\tau+1)+\tau+\kappa+7$. The protocol is secure as long as $h_W \geq 2\lambda_c\tau + t + \kappa + 4 + \max(\lambda_c+3, \kappa)$ and κ is sufficiently large (specifically, $\kappa \geq 4 + \tau + \log_2(2\lambda_c+1)$).

We now show why this theorem is sufficient for our result. Kanukurthi and Reyzin [15] use a message authentication protocol (in a black-box manner) to obtain a privacy amplification protocol whose entropy loss is the same, up to constant factors, as the entropy loss of the authentication protocol for sending a message m of length $\lambda_m = \theta(\kappa)$. In this work, we use the improved protocol given by the theorem above as a substitute for message authentication protocol Auth used in [15]. This, together with the fact that e_A, ρ, and τ are constants, as well as the fact that there exist $(n, t, \ell, \varepsilon)$ extractors (such as universal hash functions) for which $t = \theta(\ell + \log \frac{1}{\varepsilon})$ gives us the following corollary to Theorem 2:

Corollary 1. *For a sufficiently large*[3] *security parameter κ, there exists an efficient $(h_W, \lambda_k, 2^{-\kappa}, \epsilon)$ privacy amplification protocol with entropy loss $\mathcal{O}(\kappa + \log \frac{1}{\varepsilon})$, as long as $h_W = \Omega(\kappa + \log \frac{1}{\varepsilon})$.*

The rest of the paper will focus on proving Theorem 2. Below, we present our improved authentication protocol. We present the proof of security and entropy loss for our authentication protocol in Section 5.

Improved Authentication Protocol. We start with the authentication protocol described in Section 3.3 and decrease the length of the extractor output in each round to be the constant τ defined in the statement of Theorem 2 (instead of $\kappa + 1$). This gives us an $\Theta(\lambda_m)$ entropy loss for the authentication protocol as desired. Unfortunately, the security of this protocol no longer holds. The security proof in [15] shows that in order to get Bob to accept any message $m' \neq m$, Eve must respond to at least *one* fresh random challenge from either Alice or Bob. The probability with which Eve could respond to a fresh random challenge from either Alice or Bob is $2^{-\tau+1}$, which is not low enough if τ is a constant.

To rectify this problem, we ensure that in order to make Bob accept a different message, Eve must respond to *many* (namely, $\Theta(\kappa)$) fresh random challenges, which translates into a success probability of only $2^{-\kappa}$ as desired. To do so, we have Alice transmit the message $c = \mathsf{Edit}(m)$ (see Definition 4) and Bob verify that c is a valid codeword (or, equivalently, since we do not require that codeword validity be efficiently verifiable, Alice can send m to Bob in the clear and Bob can re-encode it to check if he gets c).

We now describe the authentication protocol precisely. Let the message that Alice wishes to authenticate be $m \in \{0,1\}^{\lambda_m}$.

Protocol $\mathsf{NewAuth}(w, m)$**:**

A. Alice sends Bob the message m. Let the message received by Bob be m'.

B. Alice and Bob execute protocol $\mathsf{Auth}(w, c)$ for $c = \mathsf{Edit}(m)$, using extractor Ext for all the responses, except that Bob does not output "accept" in the last round of Auth even if all the checks pass. Instead, the parties perform the following additional steps.

C. Let the string received by Bob be denoted by c'. Bob computes $\mathsf{Edit}(m')$. If $c' \neq \mathsf{Edit}(m')$, then Bob aborts. Otherwise, Bob performs a "liveness test" on Alice by sending her a fresh random challenge $y_{\text{live}} \in \{0,1\}^{q'}$.

D. Alice, after her last step, if she has not yet aborted, sends Bob a response $r_{y_{\text{live}}}$, computed via $r_{y_{\text{live}}} = \mathsf{Ext}'(w; y_{\text{live}})$.

E. Bob checks that the received $r_{y_{\text{live}}} = \mathsf{Ext}'(w; y_{\text{live}})$. If not, he aborts. If the check passes, he outputs $\mathsf{t_B}$="accept" and $m_b = m'$.

[3] We note that this constraint on κ, which comes from the statement of Theorem 2, is very minor. Indeed, λ_m is about 2κ, because m is a key for the message authentication code with security κ. Furthermore, $1/\rho$ is under 54, because the code of [26] has rate at least 1/27, and balancing the codeword increases its length by a factor of at most 2. So λ_c is about 108κ, and $\tau = \frac{4(\kappa+2)}{\lambda_c e_A} + 1$ is about 9 for e_A obtained from [26]. So, using the codes from [26], the constraint becomes $\kappa \geq 4 + 9 + \log_2(108\kappa + 1)$, or $\kappa \geq 25$; better error-correcting codes for edit distance will lower this constraint.

Intuition. The key to our improvement is to show that protocol Auth gives Eve an edit channel in the following sense. We will show that the success probability of Eve in changing the message c to a message c' is exponential in the edit distance between these two messages; since this distance must be greater than $e_A\lambda_c$ for Bob to accept at the end, Eve will fail if $e_A\lambda_c$ is high enough. Success probability also depends exponentially on τ, with higher τ giving lower success probability.

Indeed, consider what Eve can do. She can avoid delivering a message from Alice to Bob (this corresponds to deleting a bit from the c), but then she would have to respond to Alice's fresh random challenge contained in that message on her own to avoid detection; the probability of guessing a correct response to such a challenge is at most $2^{-\tau+1}$. She can also change a message from Alice that conveys a "0" bit into a message that conveys a "1" bit, but that would require coming up with a response to Bob's challenge, which again can happen with probability at most $2^{-\tau+1}$. If Eve attempts to do these things multiple times, the freshness of random extractor seeds (almost) guarantees independence and the joint probability of Eve succeeding in all of these events is (almost) the product of the probability of her success in each of the individual events. In addition, she can also perform actions where she does not need to reply to extractor queries on her own. Indeed, she can "insert" "0" bits by (easily) responding to Bob's challenges on her own and can also change a "1" to a "0", but since the number of 1s and the total length are fixed, she will have to pay elsewhere with deletions and changes of "0" to "1".

The final "liveness test" is needed to make sure that Bob detects if Alice aborts, since our argument so far only guarantees that *either* Alice or Bob will abort. (Note that this wasn't the problem in the protocol Auth, because if Alice aborts, Bob will abort in the next round with probability $2^{-\kappa}$, since Eve will have to respond to a fresh random challenge from Bob; and if Alice aborts in the last round, then we already have $m_a = m_b$, anyway.)

We translate this intuition into a proof in the next section.

5 Proof of the Main Theorem

In this section, we give a high-level overview of the proof of our main theorem. Namely we show that the new authentication protocol we presented in Section 4 is a secure authentication scheme with $\Theta(\lambda_m)$ entropy loss, where λ_m is the length of the message being authenticated.

We prove this in two broad steps. First we show that the authentication protocol gives Eve an edit distance channel in the following sense: Eve can modify several bits of the message that Alice authenticates to Bob. Second, we argue irrespective of what algorithm Eve uses or what modifications she does, the likely edit distance between the two messages can be bounded in terms of the security parameter, because the more bits Eve modifies, the less likely she is to be to able to answer all the required challenges successfully. As long as the code we use can detect changes up to this distance, Eve will fail.

Technical Challenges. As mentioned before, the security proofs of [24, 15] rely on showing that for Eve to modify the message that Alice sent, she would have to respond to at least one fresh random challenge on her own. The proofs use induction on the length of the message received by Bob so far and show that at any stage either Eve has responded to a random challenge on her own or the string received by Bob is essentially the same as what Alice had sent.

We, on the other hand, cannot use such an inductive proof on the length of the message for the following reason. The statement we want to make is not about whether Eve responded to a random challenge on her own or not. Instead, we would want to keep track of how many random challenges Eve responded to **and** precisely study the effects of responding to these challenges on the edit distance of the messages. Since the entire protocol is just an interleaving of challenges and responses sent back and forth, categorizing points in the protocol where Eve responded to a fresh random challenge on her own becomes a delicate task. In fact, it turns out that viewing the protocol in terms of the message received by Bob (or Alice) as was done in [24, 15] does not capture all the information needed to categorize the points where Eve had to respond to a random challenge.

Instead, we need to use a new proof technique, in which we view the entire protocol from Eve's perspective. In fact, we represent Eve's view of any run of the authentication protocol by a string E. This string will allow us to capture all the information including the order in which Eve interacted with the honest parties. This turns out be crucial in categorizing points in the protocol in which Eve had to respond to a random challenge. Once we do this, we use combinatorial arguments to relate the number of random challenges that Eve responded to on her own to the edit distance between the messages of the honest parties. Finally, we compute the probability with which Eve can respond to all the fresh random challenges so that neither Alice nor Bob rejects (the last-round liveness test ensures that if Alice rejects, then Bob does, too).

Organization. In Section 5.1 we will introduce the notation for the string representation E of the protocol from Eve's point of view. Using that notation, in Section 5.3.1 we will characterize the points in the protocol (corresponding to literals in the string E) where Eve must respond to a fresh random challenge. We call these points as costly literals. Now, if we could compute the probability with which Eve can respond to every fresh random challenge, and relate the number of costly literals in E to the edit distance between the two messages c and c', then we will be done. We do precisely this. In Section 5.3.2, we present the details of relating the edit distance between c and c' to the number of costly literals in the string E, showing that a high edit distance (required by the edit distance codes if $c \neq c'$, because Bob verifies that c' is a codeword) implies a high number of costly literals in E. Finally, we compute the probability with which Eve can respond to all fresh random challenges (which correspond to the costly literals in E) in Section 5.3.3.

Notation. Let $\mathbb{A}$ be an alphabet. Let $a \in \mathbb{A}$ be a literal from the alphabet. When we write a^*, we mean all strings of the form $\underbrace{aa\cdots a}_{i}$, where $i \geq 0$ is an integer. When $i \geq 1$, we write a^+. Let x_1 and x_2 denote any two strings. We write $x_1||x_2$ to denote the concatenation of the two strings.

5.1 String representation of the authentication protocol

We now present our proof ideas in more detail. As mentioned before, we view the entire protocol as it takes place from Eve's perspective. Alice and Bob no longer communicate with each other; they always communicate with Eve. (For example, if Alice sends a message to Bob that Eve doesn't modify, we view it as Alice sending a message to Eve and Eve forwarding it on to Bob.)

At the beginning, Alice sends Eve a challenge x_1. Subsequently, in any round i, $1 \leq i \leq \lambda_c$, Eve's interaction with Alice will consist of Eve sending a challenge y_i and response to the challenge x_i issued by Alice in the previous message (which should be equal to $\mathsf{Ext}(w; x_i)$—else, Alice will abort). Alice then sends Eve a response to y_i (namely, $(1, \mathsf{Ext}(w; y_i))$ or $(0, \perp)$), as well as the next challenge x_{i+1}. For $i = \lambda_c + 1$, Eve sends Alice the response to the challenge x_{λ_c+1} and as well as the challenge y_{live}, a seed to $\mathsf{Ext}'(\cdot)$ (where $\mathsf{Ext}'(.)$ is as defined in Theorem 2). We will call this two-message interaction that starts with a message from Eve to Alice and then from Alice to Eve as a *roundtrip* between Eve and Alice. To be more concise, we will use the literal 'a' to denote this *roundtrip* that takes place between Eve and Alice. Note that the initial sending of x_1 by Alice is not a roundtrip and is not represented by a literal.

Let us now consider Eve's interaction with Bob. In any round $1 < i \leq \lambda_c$, Eve's interaction with Bob starts with Eve sending a challenge x_i and response to the challenge y_{i-1} issued by Bob in the previous round, which should be equal to $(1, \mathsf{Ext}(w; y_{i-1}))$ or $(0, \perp)$ (else, Bob will reject). Bob then sends Eve a challenge y_i and a response to the challenge x_i, namely, $\mathsf{Ext}(w; x_i)$. For $i = 1$, the interaction is the same, except that Eve does not provide a response in her first message, because there is no y_{i-1}. For $i = \lambda_c + 1$, the interaction is the same, except that Bob provides $y_{\rm live}$, a seed to $\mathsf{Ext}'(\cdot)$, instead of a seed to $\mathsf{Ext}(\cdot)$. In the last round of interaction between Eve and Bob, she simply sends him a response $r_{y_{\rm live}}$ and gets nothing in response (but Eve still has to make sure the response is valid, because otherwise Bob will abort). We denote each such *roundtrip* between Eve and Bob by 'b'.

The notation a and b will be important for our proof. Note that we do not index a or b by the round i. This is because that information will largely be irrelevant to us. However, in any roundtrip with Alice, Alice is authenticating some bit (either 0 or 1) and Bob is receiving some bit (either 0 or 1). This allows us to have two different literals, a_0 and a_1 depending on which bit Alice is authenticating in that roundtrip. Likewise, we will also use b_0 and b_1 to denote Eve's roundtrips with Bob depending on which bit Bob is receving in that roundtrip. If we do not subscript the a or b literals, it means that the claim holds irrespective of the bit being authenticated.

Certain rounds of interaction deviate from the usual interactions. So we will distinguish them by using specially subscripting the corresponding a and b literals. In the first interaction between Eve and Bob, he receives just a challenge from Eve and no response to any challenge. (He also sends a challenge y_i and a response $r_{x_1} = \mathsf{Ext}(w, x_1)$.) So we will distinguish the first interaction between Eve and Bob from subsequent interactions by denoting it as b_{ch}. Likewise, in the last interaction between Eve and Alice, Alice replies with a long extractor response, i.e., with $r_{y_{\rm live}} = \mathsf{Ext}'(w, y_{\rm live})$. We will denote this last interaction between Eve and Alice by a_{long}. Finally, in the last round of interaction between Eve and Bob, Bob receives **only** a response from Eve and no challenge (to which Bob would have to respond). So we will denote this last interaction between Eve and Bob by b_{re}. (Note that Eve's interaction with Bob also deviates in the second-to-last round (i.e., $\lambda_c + 1$) because Bob sends Eve a seed to $\mathsf{Ext}'()$ rather to Ext; but we will not require special notation for this round.)

Using this notation, we can write out the entire protocol from Eve's point of view by simply creating a string (call it E) denoting Eve's actions. As an example, if Eve is passive and Alice is authenticating $c = c_1, c_2, \cdots, c_{\lambda_c}$ to Bob, then $\mathrm{E} = b_{ch}, a_{c_1}, b_{c_1}, a_{c_2}, b_{c_2}, \cdots, a_{c_{\lambda_c}}, b_{c_{\lambda_c}}, a_{long}, b_{re}$. Note that since Eve might be active, E will not necessarily take the structure as above. Instead E can be any interleaving of a and b literals.

As we already said, a and b literals, except b_{re}, represent roundtrips; thus, if Eve sends a message that causes Alice or Bob to abort, the string E does not contain the corresponding literal. Moreover, even though b_{re} is not a roundtrip, E will not contain b_{re} if Bob rejects. Thus, the presence of a_{long} in E indicates that Alice accepts and the presence of b_{re} in E indicates that Bob accepts.

Observe that no matter what Eve does, her entire interaction can be represented by a string E. Indeed, all Eve can do is send challenges and extractor responses to Alice and Bob, interleaving them arbitrarily. We can assume that Alice and Bob respond instantaneously (because it only makes Eve stronger if she does not have to wait), so we do not need model a situation in which Eve has two simultaneous round trips with Alice and Bob. Therefore, we can always write down the string E by looking at one round-trip at a time from Eve's point of view. We next give a high-level outline of how to use the string E to prove that Protocol $\mathsf{NewAuth}$ is secure.

5.2 Outline of the Proof

We first recall some notation. We assume that, at the start of the protocol, Alice sends Bob the message m that she wishes to authenticate. Let Bob receive m'. Alice runs protocol $\mathsf{NewAuth}$ to authenticate the codeword, $c = edit(m)$. c' denotes the codeword received by Bob. Our ultimate goal will be use the string representation of the authentication protocol to argue that Eve's probability of attacking the authentication scheme is low. In particular, we wish to show that the event $(m' \neq m \wedge \mathsf{t_B} = \text{“}accept\text{”})$ occurs with a low probability.

To show this, we need to introduce the notion of a "well-formed" string representation. We call the string E "well-formed" if it ends with a_{long}, b_{re}. We will be able to prove that the event $(m' \neq m \wedge \mathsf{t_B} = \text{“accept”} \wedge \mathrm{E}$

is well-formed) occurs with probability of at most $2^{-(\kappa+1)}$. We will also prove that the event ($\mathsf{t_B} =$"accept" $\wedge$E is not well-formed) occurs with probability at most $2^{-(\kappa+1)}$. It is easy to see that proving these two statements suffices to prove our main theorem by the union bound.[4]

5.3 Eve's success when E is well-formed

For this section we consider **only** the case when E is well-formed and analyze Eve′s probability of success in this situation. We handle the scenario where E is not well-formed in Section 5.4.2.

Let α(E) be the function that outputs the subscripts of the a literals (other than a_{long}) read out in order. Then it is easy to see that α(E) represents the message that Alice thinks she sent. Likewise if β(E) is a function that outputs the subscripts of the b literals (other than b_{ch} and b_{re}) read out in order, then β(E) is simply the message received by Bob. Observe that if Eve is passive, E will take the structure described above and $\alpha(\mathrm{E}) = \beta(\mathrm{E})$ (which is consistent with the fact that Alice's and Bob's messages are equal). Note that the literals a_{long}, b_{ch}, b_{re} do not form a part of the messages $\alpha(\mathrm{E}), \beta(\mathrm{E})$.

Recall that we would like to use the string representation E to categorize those points in the protocol where Eve would have to respond to a fresh random challenge. This brings us to the notion of *costly literals.*

Definition 6. *A literal in E is costly if in the real run of the protocol (that E represents), either Eve would have to respond to a fresh random challenge on her own in the roundtrip corresponding to the literal, or Alice or Bob would abort.*

5.3.1 Characterizing costly literals in string E

In this section, we precisely characterize the literals in E that are costly. We first prove the following lemma.

Lemma 5. *The following statements about costly literals are true:*

A. *Any a-literal (including a_{long}) is costly if it is not immediately preceded by a b-literal in E.*

B. *The last literal in the following sequence is costly: $ba_0^*b_1$. Equivalently, a b_1-literal is costly if there are no a_1-literals between it and the closest preceding b-literal. (Note that the sequence starts with b_{ch}, so the closest preceding b-literal is well defined.)*

Proof. We will consider each case of the claim separately.

A. Indeed, consider an a-literal that is not preceeded by a b-literal. It requires Eve to respond to Alice's outstanding challenge, issued in the previous a-literal (or, for the first a-literal, to x_1). By definition of fresh random challenge (Definition 5), this challenge is fresh, because Eve did not interact with either party after receiving it, because there is no a- or b-literal between the current one and the time the challenge was issued.

B. Indeed, consider such a b_1 literal. It requires Eve to respond to Bob's outstanding challenge, issued in the previous b literal. Eve receives no extractor outputs for a_0-literals. And, because E is well-formed, a_{long} cannot precede b_1, Therefore, by Definition 5, this challenge is fresh.

□

[4] It is tempting to think that the union bound is not necessary and we could therefore slightly improve our parameters. Indeed, we could avoid the union bound if we bound the following *conditional* probabilities by $2^{-\kappa}$: $\Pr[m' \neq m \wedge \mathsf{t_B} =$"accept"$\mid$ E is well-formed] and $\Pr[\mathsf{t_B} =$"accept" $\mid$ E is not well-formed]. If Eve chooses independently of anything in the protocol whether to try for a well-formed E or a not well-formed E, then we can bound these conditional probabilities in the same way as we bound the unconditional ones. However, since we allow Eve to be adaptive, she can decide whether to aim for a well-formed E depending on how the protocol proceeds: for example, she can choose to go for a not-well-formed E only in the rare cases when she can be confident of her success. Thus makes bounding the conditional probability impossible. The necessity of the union bound here is the same as in [15].

5.3.2 Bounding edit distance in terms of the number of costly literals

For this section, we can forget about the details of the protocol altogether and focus on the properties of the string E. We wish to show that a high edit distance between $\alpha(\mathrm{E})$ and $\beta(\mathrm{E})$ cannot be achieved without a lot of costly literals. In other words, we want to show that if there are few costly literals, then the edit distance is small. We will show this by constructing a method to convert $\alpha(\mathrm{E})$ into $\beta(\mathrm{E})$ such that the total number of edit operations (insertions and deletions) performed is no greater than some constant times the number of costly literals. We will do this in two steps:

A. First, we will present an algorithm that converts the string E into a new string E' with the following properties:

   - Let $c' = c'_1, \cdots, c'_{\lambda_c}$ denote $\beta(\mathrm{E})$ i.e., the string received by Bob.
   - Then $\mathrm{E}' = b_{ch}, a_{c'_1}, b_{c'_1}, \cdots, a_{c'_i}, b_{c'_i}, \cdots, a_{c'_{\lambda_c}}, b_{c'_{\lambda_c}}, a_{long}, b_{re}$ and, in particular, $\beta(\mathrm{E}') = \alpha(\mathrm{E}') = c'$.
   - The edit distance between E and E' is bounded by a constant times the number of costly literals in E.

B. Finally, we show that if E and E' are well-formed strings that do take the structure as above, then the edit distance between c and c' is equal to the edit distance between E and E'.

The next theorem and its corollary formally state and proving the relation between the edit distance and the number of costly literals in E. We will conclude this section by showing a lower bound on the number of fresh random challenges that Eve must respond to in order to succeed in modifying the message.

Let the number of costly literals in E be at most L. Then the following theorem states that the edit distance between the message authenticated by Alice and received by Bob is at most $4\mathsf{L}$.

Theorem 3. *Let E be a string consisting of a_0, b_0, a_1, b_1 literals, as well as the special literals a_{long}, b_{ch}, b_{re} that each appear only once. Assume that the number of b_0 literals is equal to the number of a_0 literals, the number of b_1 literals is equal to the number of a_1 literals, that the first b literal is b_{ch}, and that the string ends in $a_{long}b_{re}$. Assume the number of costly literals in E be at most L. Let E' be the string defined above. Then the edit distance between E and E' is at most $4\mathsf{L}$ (and, by definition of E', $\alpha(E') = \beta(E') = \beta(E)$).*

Proof. To convert E into E',

A. We first scan E and mark what we call the *edit* literals in E. These edit literals are marked so that the edit distance between E and E' is at most number of edit literals in E.

B. Now, we need to show that if the number of costly literals is L, then the number of edit literals is at most $4\mathsf{L}$. It will be easier to prove the above statement by categorizing edit literals into (disjoint sets of) *good edit literals* and *bad edit literals*. Our proof will be in three steps:

   (a) Using notation $\#bad$ to denote the number of bad edit literals and $\#edit$ to denote the number of edit literals, we first show that $\#edit \leq 2 \times \#bad$ (Lemma 6).

   (b) Next, letting $\#costly$ to denote the number of costly literals, we will show that $\#bad \leq 2 \times \#costly = 2\mathsf{L}$ (Lemma 7).

   (c) Finally, combining the above two lemmas, we get $\#edit \leq 4\times\ \#costly = 4\mathsf{L}$, which is the required statement.

We now proceed to give the details. We begin by explain what we do with the the string once the edit literals are marked. Next, we show how to mark literals as good edit literals and bad edit literals.

Definition 7. Edit Literals: *When a literal in E is marked as an edit literal, it corresponds to the following:*

- *If an a_p-literal (where $p \in \{0,1\}$) is marked as an edit literal, then to convert E into E′, the edit a_p-literal is deleted from E.*
- *When a b_p-literal (where $p \in \{0,1\}$) is marked as an edit literal, then when going from E to E′, an a_p literal is inserted just before the b_p literal.*

Clearly E can be converted to E′ by at most as many insertion and deletion operations as the number of edit literals, and hence the edit distance between E and E′ is at most number of edit literals. As we describe next, to mark the edit literals in E, we first split E into disjoint substrings and mark the edit literals in each substring.

Marking Bad/Good Edit Literals in E. To do this, let E be written as the concatenation of k strings; i.e., $\mathrm{E} = \mathrm{E}_1||\mathrm{E}_2||\cdots||\mathrm{E}_k$. Each E_i consists of a contiguous sequence of one or more a literals followed by one or more b literals (except for E_1, which may not have any a literals). For example,

$$\mathrm{E} = \underbrace{a_1, a_1, a_1, b_{ch}, b_1}_{\mathrm{E}_1}, \underbrace{a_0, a_1, b_0}_{\mathrm{E}_2}, \underbrace{a_1, \ldots}_{\mathrm{E}_3}, \quad \ldots, \quad \underbrace{\ldots, b_0}_{\mathrm{E}_{k-1}}, \underbrace{a_1, a_{long}, b_{re}}_{\mathrm{E}_k}$$

or

$$\mathrm{E} = \underbrace{b_{ch}}_{\mathrm{E}_1}, \underbrace{a_1, a_1, a_1, b_0, b_1}_{\mathrm{E}_2}, \underbrace{a_0, \ldots}_{\mathrm{E}_3}, \quad \ldots, \quad \underbrace{\ldots, b_0}_{\mathrm{E}_{k-1}}, \underbrace{a_1, a_0, a_{long}, b_{re}}_{\mathrm{E}_k}.$$

Note that E is well-formed, so E_k always ends with $a_{long}b_{re}$.

We now describe the algorithm to mark literals in each substring E_i as (good/bad) edit literals. Note that we never mark the literals a_{long}, b_{ch}, b_{re} as edit literals.

Algorithm MarkEdits

(1) E_1 has the form $a^*b_{ch}b^*$. Mark all the literals in it except b_{ch} as bad edit literals.

(2) For $i > 1$, if E_i is not the last substring, then it has the form $a^*a_pb_qb^*$ (where $p \in \{0,1\}, q \in \{0,1\}$). Call the last a literal and the first b literal in E_i "pivots" and proceed as follows:

  (a) if $p = q$, mark every literal in E_i other than the pivots as a bad edit literal.

  (b) if $p = 1, q = 0$, mark every literal in E_i other than the a-pivot as a bad edit literal. In addition, mark the a-pivot as a good edit literal.

  (c) if $p = 0, q = 1$: If E_i is of the form $a^*a_1a^*a_0b_1b^*$, make the a literal subscripted by 1 as the pivot (instead of the original a-pivot) and mark all literals other than the pivots as bad edit literals. (There may be many different a_1 literals. Which specific one we choose as the pivot is immaterial.)

  (d) if $p = 0, q = 1$: If E_i is not of the form $a^*a_1a^*a_0b_1b^*$ (i.e., E_i is of the form $a_0^*a_0b_1b^*$), mark every literal in the E_i other than the a-pivot as a bad edit literal. In addition, mark the a-pivot as a good edit literal.

(3) The last E_k has the form $a^*a_{long}b_{re}$. Mark all a literals as bad edit literals, except a_{long}.

By inspection, after the edit literals are marked and the string is derived by deletion and insertion according to the markings, the result is E′. It remains to analyze the number of edit literals.

Lemma 6. *$\#edit \le 2 \times \#bad$.*

Proof. Let $\#good$ denote the number of good edit literals. Since the set of good and bad edit literals are disjoint, it holds that $\#edit = \#good + \#bad$. So it suffices to show that $\#good \leq \#bad$. It is easy to see that the only times we mark a literal as a good edit literal are in cases 2b and 2d. In both cases, the good edit literal is the a-pivot; and corresponding to it there is a unique b literal (namely the b-pivot) that is marked as bad. □

Lemma 7. *$\#bad \leq 2 \times \#costly$.*

Proof. To prove this lemma, we will categorize our bad edit literals into bad_{b_0}, bad_{b_1}, and bad_a disjoint sets of literals. To show that $\#bad \leq 2 \times \#costly$, we need to show that $bad_{b_0} + bad_{b_1} + bad_a \leq 2 \times \#costly$. We will prove this in two steps. In Lemma 8 we will show that $\#bad_a + \#bad_{b_1} = \#costly$. Then, in Lemma 9, we will show that $\#bad_{b_0} \leq \#bad_a + \#bad_{b_1}$. □

Lemma 8. *$\#bad_a + \#bad_{b_1} = \#costly$.*

Proof. We will prove the lemma by proving that it holds for every substring E_i defined as above.

First consider $i = 1$. The substring E_1 is of the form $a^* b_{ch} b^*$. In this case, all the a literals are marked as bad edit literals, and, indeed, they are all costly by Lemma 5, because none of them is preceded by a b literal. Similarly, all b_1-literals are marked as bad edit literals, and, indeed they are all costly by Lemma 5, because there are no a literals between them and b-literals that precede them.

If $1 < i < k$, then E_i of the form $a^* a_p b_q b^*$ (where $p, q \in \{0, 1\}$), then Algorithm MarkEdits marks all the a-literals except the a-pivot as a bad edit literal. And, indeed, all but one of the a-literals is costly by Lemma 5, because only the first is immediately preceded by a b literal. Hence, the number of bad and costly a-literals is the same. If the first b-literal in E_i is b_1, then it is marked as bad by Algorithm MarkEdits whenever there is no a_1 literal in E_i, which is exactly when it is costly by Lemma 5. All the other b_1 literals are always marked as bad and, indeed are costly, because there are no a literals immediately preceding them.

Finally, E_k of the form $a^* a_{long} b_{re}$, then all a-literals but the first one are costly by Lemma 5, and all but the last one are marked as bad, so again the number of costly literals is equal to the number of bad ones. □

Lemma 9. *$\#bad_{b_0} \leq \#bad_a + \#bad_{b_1}$.*

Proof. Recall (from right before the statement of Theorem 3) that the number of 0s in $\alpha(\mathrm{E})$ is equal to the number of 0s in $\beta(\mathrm{E})$. Therefore, the number of b_0 literals is the same as the number of a_0 literals. By inspecting the cases in Algorithm MarkEdits, we see that b_0 literal is marked as bad if and only if it is not preceded immediately by an a_0 literal. Call an a_0 literal *displaced* if it is not immediately succeeded by a b_0 literal. Because the number of a_0 literals is equal to the number of b_0 literals, the number of displaced a_0 literals is equal to the number of bad b_0 literals. We will now analyze the number of displaced a_0 literals.

Consider a displaced a_0 literal. Because E is well-formed, it must be immediately succeeded by an a literal, a b_{ch} literal, or a b_1 literal. In the first two cases, it will be marked as bad. In the third case, it will be marked as bad (by Step 2c of Algorithm MarkEdits), or the following b_1 literal will be marked as bad (by step 2d of Algorithm MarkEdits).

Thus, every displaced a_0 literal is either marked as bad, or corresponds to a bad b_1 literal. Hence, the number of displaced a_0 literals is at most $\#bad_a + \#bad_{b_1}$. □

We combine Lemma 8 and Lemma 9 to get Lemma 7. From Lemma 7 and Lemma 6 we obtain Theorem 3. □

Theorem 3 states if E has at most L costly literals, then the edit distance between E and E′ is at most $4\mathsf{L}$. Ultimately, we need a bound on the edit distance between $\alpha(\mathrm{E})$ and $\beta(E)$. We show it in the following Corollary.

Corollary 2. *Let E be a string consisting of a_0, b_0, a_1, b_1 literals, as well as the special literals a_{long}, b_{ch}, b_{re} that each appear only once. Assume that the number of b_0 literals is equal to the number of a_0 literals, the number of b_1 literals is equal to the number of a_1 literals, that the first b literal is b_{ch}, and that the string ends in $a_{long}b_{re}$. Assume that the number of costly literals in E is at most L. Then the edit distance between $\alpha(E)$ and $\beta(E)$ is at most $4\mathsf{L}$.*

Proof. The procedure described in Theorem 3 transforms E into E$'$ by using insert and delete operations only on the a literals, leaving b literals untouched. Therefore, edit-distance$(\alpha(\mathrm{E}), \alpha(\mathrm{E}')) =$ edit-distance$(\mathrm{E}, \mathrm{E}') \leq 4L$. And $\alpha(\mathrm{E}') = \beta(\mathrm{E}') = \beta(\mathrm{E})$, so edit-distance$(\alpha(\mathrm{E}), \beta(\mathrm{E})) =$ edit-distance$(\mathrm{E}, \mathrm{E}')$. And, by Theorem 3, edit-distance$(\mathrm{E}, \mathrm{E}') \leq 4L$. ☐

Corollary 3. *Let Alice and Bob execute protocol* $\mathsf{NewAuth}$ *in the presence of an active adversary Eve. Assume that the string representation of the authentication protocol, E, is well-formed. (This automatically implies that Alice and Bob accept the protocol transcript.) If the message m_b output by Bob is not the same as the message m_a sent by Alice, then Eve must have responded to at least $e_A\lambda_c/4$ fresh random challenges on her own.*

Proof. Since Bob accepts, must be that c' is a valid codeword, which means that Bob received the same length string with the same number of zeros and ones as Alice sent (because the edit distance code Edit contains only fixed-length, fixed-Hamming-weight strings). Therefore, E satisfies the conditions of Corollary 2. Assume that Eve responded to fewer than $e_A\lambda_c/4$ fresh random challenges. Then, by Definiton 6, E has fewer than $e_A\lambda_c/4$ costly literals, and thus by Corollary 2, the edit distance $\alpha(\mathrm{E})$ and $\beta(\mathrm{E})$ is less than $e_A\lambda_c$. But $\alpha(\mathrm{E})$ and $\beta(\mathrm{E})$ are distinct valid codewords, which contradicts the properties of the edit distance code from Definition 4. ☐

5.3.3 Relating the number of fresh random challenge responses to the probability of Eve's success

So far we have shown that the edit distance between the message authenticated by Alice and received by Bob gives us a lower bound on the number of fresh random challenges (or extractor seeds) that Eve needs to respond to (with extractor outputs) on her own. In this section, our goal will be to get an upper bound on the probability with which she can succeed in responding to all of those random challenges. Note that she needs to respond to all of them correctly so that neither Alice nor Bob abort the protocol, because if Alice aborts, then the last-round liveness test will ensure that Bob aborts, too, as we will show in Section 5.4.2.

Observe that in each round of Protocol $\mathsf{NewAuth}$, Eve receives a fresh random challenge (i.e., an extractor seed) from one of the honest parties (say Alice). Eve can see the seed and then decide whether she wants to respond with a guess for the extractor output on her own or whether she wants to talk to Bob first (she can send Bob the same seed or modify it). She has this freedom even if she knows in advance the modifications she wants to make to the message being authenticated. Indeed, consider, for example, a portion of the protocol where Alice is authenticating a sequence of 0s to Bob. Suppose that Eve would like to delete one of these 0s. In this case, it doesn't matter to her which 0 she is deleting and, therefore, which challenge she choses to respond to. She could first see the challenge seed and respond to it on her own only if the entropy of the extractor conditioned on that **specific seed** is lower than the average entropy of an extractor output. This leads us to a subtle concern that we need to address: even though Lemma 3 guarantees that the entropy of an extractor output is high *on average*, Eve could improve her chances by adaptively choosing which extractor seed she wishes to respond to. We handle this issue by resorting to Lemma 4 which bounds the probability with which an extractor output evaluated on a randomly chosen seed has entropy less than $\tau - 1$. We use this lemma to first bound the probability that Eve will receive even one extractor seed (in the entire run of Protocol $\mathsf{NewAuth}$) for which the output entropy is less than $\tau - 1$.

Once this event doesn't happen (and, therefore, the entire protocol doesn't contain a single bad seed), we know that the extractor outputs have high entropy. We now give a high level overview of how to bound the probability with which Eve succeeds in responding to **multiple** random challenges assuming that there are no bad seeds. Since the extractor output is τ bits long, Eve succeeds in responding to a single random challenge with probability at most $2^{-(\tau-1)}$. Then we would expect that the probability that Eve responds to μ fresh random challenges (chosen independently) would be at most $2^{-\mu(\tau-1)}$. Unfortunately, for reasons explained after the proof of the next lemma, that's not quite the case, but we can get something close: we show that the probability that Eve responds to μ fresh random challenges on her own is $2^{-\mu(\tau-1)} + 2^{-\kappa-3}$, if the average min-entropy of w at the last extraction is at least $t + 2\kappa$ (where, recall, t is the entropy needed for the extractor to work).

We now proceed with the formal argument. Consider a run of Protocol $\mathsf{NewAuth}$, excluding the last message $r_{y_{\text{live}}}$ from Alice to Bob. We know that Eve receives one extractor seed (challenge) in each round of interaction with an honest party. Among all the extractor seeds she receives, we denote the random challenges to which she chooses to respond on her own (rather than by first talking to the other party) by $z_1, \cdots, z_\mu$. Let $S_j = 1$ if Eve's response to z_j is correct, and 0 otherwise. Note that if $S_j = 0$, then Alice or Bob (whoever sent z_j) will abort. Once both parties abort, there are no more random challenges, so μ may be smaller or larger depending on how successful Eve is. For ease of notation, let $S_j =\perp$ for $j > \mu$. Let Tr denote all the messages Eve receives from Alice and Bob, up to, but not including, $r_{y_{\text{live}}}$ (if Alice or Bob aborts before $r_{y_{\text{live}}}$ is sent, then Tr includes information about when the abort happens). We now prove the following lemma.

Lemma 10. *Assume that $\widetilde{\mathbf{H}}_\infty(W \mid \mathsf{Tr}) \geq 2\kappa + t$. Then, for any μ, $\Pr[S_1 = S_2 = \cdots = S_\mu = 1] \leq 2^{-\mu(\tau-1)} + 2^{-\kappa-3}$ (where the probability is taken over the choice of w and randomness of the challenges.)*

Proof. Let Tr_i be the first i messages in Tr (if both parties abort before sending i messages, then Tr_i is undefined). Note that $\widetilde{\mathbf{H}}_\infty(W \mid \mathsf{Tr}_i) \geq 2\kappa + t$. Let G_i be the event that Tr_i is such that $\mathbf{H}_\infty(W \mid \mathsf{Tr}_i) \geq t$. Then the complement $\bar{G_i}$ is the event that the entropy loss due to the protocol communication is greater than the expected value by at least 2κ bits. By Lemma 2, $\bar{G_i}$ happens with probability at most $2^{-2\kappa}$.

Let H_i be the event that the random challenge z issued in the i^{th} message is such that $\mathbf{H}_\infty(\mathsf{Ext}(W; z) \mid \mathsf{Tr}_i) \geq \tau - 1$. If G_i holds, then W has sufficient entropy conditioned on Tr_i and z is uniform (because G_i does not bias z, it only potentially biases challenges before z, since responses to them are in Tr_i). Hence we can apply Lemma 4 with $\ell = \tau$ and $\varepsilon = 2^{-2\kappa}$ to get that $\Pr[\bar{H_i} \mid G_i] \leq 2^{\tau-2\kappa}$. Therefore, $\Pr[\bar{H_i} \cup \bar{G_i}] = \Pr[\bar{H_i} \cap G_i] + \Pr[\bar{G_i}] = \Pr[\bar{H_i} \mid G_i]\Pr[G_i] + \Pr[\bar{G_i}] \leq 2^{\tau-2\kappa} + 2^{-2\kappa}$.

For ease of notation, for i greater than the number of messages in Tr, define H_i and G_i to be always true. Let G be $\bigcap_i (G_i \cap H_i)$. That is, G is the event that Tr is such that $\mathbf{H}_\infty(W \mid \mathsf{Tr}) \geq t$, and, for every random challenge in Tr, the extractor response conditioned on the transcript-so-far has entropy at least $\tau - 1$. Thus, for each j, $\Pr[S_j = 1 \mid G] \leq 2^{-\tau-1}$, because Eve has to guess a value with entropy at least $\tau - 1$.

Because Alice and Bob issue at most $2\lambda_c+1$ challenges (not including y_{live}), the probability that G does not hold is $\Pr[\bar{G}] \leq (2\lambda_c + 1)(2^{\tau-2\kappa} + 2^{-2\kappa})$ by the union bound. By the constraint on κ in the statement of Theorem 2, we know that $(2\lambda_c + 1)2^\tau \leq 2^{\kappa-4}$, ensuring that $\Pr[\bar{G}] \leq (2\lambda_c + 1)2^\tau(2^{-2\kappa} + 2^{-2\kappa}) \leq 2^{\kappa-4} \cdot 2^{-2\kappa+1} = 2^{-\kappa-3}$.

Therefore,

$$\begin{aligned}
\Pr\left[S_1 = 1 \cap S_2 = 1 \cap \cdots \cap S_\mu\right] &\leq \Pr\left[S_1 = 1 \cap S_2 = 1 \cap \cdots \cap S_\mu = 1 \mid G\right]\Pr[G] + \Pr[\bar{G}] \\
&\leq \Pr[S_1 = 1 \mid G] \times \Pr[S_2 = 1 \mid S_1 = 1 \cap G] \times \cdots \\
&\quad \cdots \times \Pr[S_\mu = 1 \mid S_1 = \cdots = S_{\mu-1} = 1 \cap G] + 2^{-\kappa-3} \\
&\leq 2^{-\mu(\tau-1)} + 2^{-\kappa-3}.
\end{aligned}$$

The last inequality holds because the transcript, by definition, includes information on whether Alice and Bob abort, and therefore, information on whether each S_j is 0, 1, or $\perp$. Therefore, adding the values of

$S_1, \dots, S_{j-1}$ to the condition does not change the fact that the answer to the challenge z_j has entropy at least $\tau - 1$, because we are still conditioning on nothing more than a transcript from G.

$\square$

Need for Extra Buffer of 2κ Entropic Bits. In the proof above, it may seem unclear why we need to impose a restriction on the starting entropy being 2κ bits greater than needed by simple average-entropy arguments. On an intuitive level, it might seem that since there is guarantee on the average entropy of w, and because the challenges are all chosen independently of each other, we should be able to say that (on average) probability of success in responding to each of the challenges should just be independent. It seems intuitive that we should just be able to multiply the probability that each $S_j = 1$ directly instead of having to condition on sufficient entropy.

Unfortunately, this intuition seems wrong. Average entropy gives us a guarantee that, on average, the transcript doesn't reveal too much information about w. But we are conditioning not on the average case, but on Eve's success. So consider the case where the first challenge is particularly lucky (for Eve), so that that it reveals all information about w if $S_1 = 1$. Now, while the probability of this happening is very small, it still destroys independence properties because, conditioned on this event, $S_2 = S_3 = \cdots = S_\mu = 1$ with probability 1.

One might be tempted to think that one can bypass such issues by simply adding together the average entropy of each response that Eve needs to give, and considering the average entropy of the entire set of responses. But this approach also doesn't seem to work. For instance, consider the seemingly intuitive statement: "if $\mathbf{H}_\infty(A) \geq m_1$ and $\widetilde{\mathbf{H}}_\infty(B \mid A) \geq m_2$, then $\mathbf{H}_\infty(A, B) \geq m_1 + m_2$." This statement is false (to see a counterexample, let (A, B) equal $(0, 0)$ and $(0, 1)$ with probability $1/8$ each and $(1, 1)$ with probability $7/8$). Thus, even though we can bound the entropy of Eve's first response and the average entropy of Eve's second response given the first, we cannot simply use those bounds to obtain the entropy of the joint distribution of the two responses.

It is for this reason that we need to take the approach of considering three cases: the case of a transcript that reveals too much, the case of a transcript that has weak seeds, and the case when neither of these happens. The first two cases occur with low probability, and in third case we have something close to independence of extractor responses.

5.4 Putting it all together

In this section we combine the above results to give the proof of our main result (Theorem 2).

Correctness The correctness of the protocol follows from the correctness of $\mathsf{Auth}(w, c)$: if both parties follow the rules and Eve doesn't interfere, then neither party will reject and the correct m is sent across in the very first message from Alice to Bob.

Entropy Loss To calculate the entropy loss, note that Alice sends at most $\lambda_c - 1$ extractor responses (because at least one bit of c must be 0, because all codewords have the same Hamming weight) of length τ plus the last extractor response of length $\kappa+\tau+\lambda_c+4$; and Bob sends λ_c+1 extractor responses of length τ. Alice and Bob also send extractor seeds, which are uniformly random; let λ_{r_a} and λ_{r_b} be the total bit-length of all these seeds for Alice and Bob, respectively. For ease of analysis, assume that Alice and Bob send all the random bits regardless of whether they abort the protocol (for instance, they send all their remaining random bits when they abort)—clearly, this does not increase the entropy loss. Then, regardless of Eve, the pair $(sent_a, \mathsf{t_A})$ can be written down as a string of length at most $(\lambda_c - 1)\tau + \kappa + \tau + \lambda_c + \lambda_{r_a} + 4$ bits (it suffices to write down everything Alice sends; if it is full-length, then $\mathsf{t_A} = 1$, else $\mathsf{t_A} = 0$). Because the number of bit strings of length at most α is less than $2^{\alpha+1}$ (notice that using "at most" instead of "exactly" allows us to handle the case of active Eve, who gets information from observing where in the protocol abort happened), the number of possible such pairs $(sent_A, \mathsf{t_A})$ is at most $2^{(\lambda_c-1)\tau+\kappa+\tau+\lambda_c+\lambda_{r_a}+5}$. Similarly, the pair $(sent_b, \mathsf{t_B})$ can be written down as a string of length at most $(\lambda_c+1)\tau+1$ (here, $\mathsf{t_B}$ cannot be inferred from the string length, hence the $+1$), and thus has at most $2^{(\lambda_c+1)\tau+2}$ values. Hence, $(sent_a, sent_b, \mathsf{t_A}, \mathsf{t_B})$ has at most $2^{2\lambda_c\tau+\kappa+\tau+\lambda_c+\lambda_{r_a}+\lambda_{r_b}+7}$ values. Also, because this variable contains $\lambda_{r_a}+\lambda_{r_b}$ independent uniform bits, $\mathbf{H}_\infty(W, (sent_A, sent_B, \mathsf{t_A}, \mathsf{t_B})) \geq \mathbf{H}_\infty(W) + \lambda_{r_a} + \lambda_{r_b}$. We can now apply Lemma 1 to get that the entropy loss is at most $2\lambda_c\tau + \kappa + \tau + \lambda_c + 7$.

Robustness We now move on to robustness. There are two ways for Eve to succeed in the security game, and we consider them in order to show that each one can happen with probability at most $2^{-\kappa-1}$, giving us robustness $2^{-\kappa}$ by the union bound (see Section 5.2).

5.4.1 Bounding probability of successful message change when E is well-formed

By Corollary 3, when E is well-formed, Eve must respond to at least $\mu = \frac{e_A\lambda_c}{4}$ fresh random challenges in order to make sure that neither Alice nor Bob aborts. By definition of "well-formed" in Section 5.2, she must do so before receiving $r_{y_{\text{live}}}$.

In order to apply Lemma 10, it suffices to ensure $\widetilde{\mathbf{H}}_\infty(W \mid \mathsf{Tr}) \geq 2\kappa + t$, where Tr is the transcript until just before $r_{y_{\text{live}}}$ is sent. We can apply the same reasoning as the entropy loss analysis above, except omitting the length $\kappa + \tau + \lambda_c + 4$ of $r_{y_{\text{live}}}$ and replacing it with a single bit indicating whether Alice accepts in the last round. Thus, $\widetilde{\mathbf{H}}_\infty(W \mid \mathsf{Tr}) \geq \mathbf{H}_\infty(W) - 2\tau\lambda_c - 4$. The setting of minimum $\mathbf{H}_\infty(W)$ in Theorem 2 ensures that this value is at least $2\kappa + t$.

Applying Lemma 10, we see that Eve will succeed with probability at most $2^{-\mu(\tau-1)} + 2^{-\kappa-3}$, where $\mu \geq \frac{e_A\lambda_c}{4}$. Because we have $\tau = \frac{4(\kappa+2)}{\lambda_c e_A} + 1$, we get the probability of Eve's success with a well-formed E is at most $2^{-\kappa-1}$.

5.4.2 Bounding Eve's success probability when E is not well-formed

To complete the proof, all that is left is to show that the probability that Eve is successful and E is not well-formed is at most $2^{-\kappa-1}$. As we indicated in Section 5.2, we will show that when E is not well-formed, getting Bob to accept will itself be an unlikely event for Eve. Recall that E is not well-formed if it doesn't end with $a_{long}b_{re}$. As we did in Section 5.3.2, we let E be written as the concatenation of k strings; i.e., $\mathrm{E} = \mathrm{E}_1||\mathrm{E}_2||\cdots||\mathrm{E}_k$ where each E_i is of the form a^*b^*. If Bob accepts, then b_{re} is the last b-literal in E. Consider the substring E_j which contains the literal b_{re}. (Clearly j can only be k or $k-1$; if $j = k-1$, E_k contains only the a literals). We consider two distinct scenarios and show that the probability that Eve will succeed in either of them is at most $2^{-\kappa-2}$, giving us a total probability of success when E is not well-formed of at most $2^{-\kappa-1}$ (even though the scenarios are mutually exclusive, we need to use the union bound here for the reason explained in Section 5.2).

Case 1: Let E_j be of the form $a^+b^+b_{re}$ or $a^+a_{long}b^+b_{re}$. In other words, there is at least one b literal that separates the preceding a literals from b_{re}. In this scenario, Eve needs to come up with the response contained in b_{re} completely on her own, because she does not talk to Alice after Bob's challenge is issued. She succeeds in this with a probability of at most 2^{-k-2} by Lemma 3, as long as the entropy of w is at least t even conditioned on Eve's view. (This follows because if entropy is at least t, the output of Ext' is of length at least $\kappa + 3$ and is at least $2^{-\kappa-3}$-close to uniform.) And the entropy conditioned on Eve's view is at least t by Lemma 1, because the entropy loss of the protocol, analyzed above, is at most $(2\tau+1)\lambda_c + \kappa + \tau + 7$, and h_W is set to exceed it by at least t in the statement of Theorem 2.

Case 2: Let E_j be of the form a^+b_{re} (where a^+ is a substring of a_0 and a_1 literals; it cannot end with a_{long} as the string would be well-formed otherwise). This scenario arises if Eve can get Bob to think that the protocol is already on its last round and have him send the liveness test challenge y_{live}. Even though Alice has not yet reached the last round, Eve can then issue multiple challenges y_i to Alice for rounds $i <= \lambda_c$ in order to get responses r_{y_i} that are correlated with the response she needs to guess, $r_{y_{\text{live}}}$. In other words, by the time Eve responds to Bob, his challenge y_{live} is no longer fresh.

Fortunately for us, each time Eve issues a challenge y_i to Alice, she also has to provide a response r_{x_i} to Alice's previous challenge, and she can't get Bob's help with that, because Bob already thinks he is on the last round. Thus, in order to gain τ bits of information about $r_{y_{\text{live}}}$ via y_i, Eve has to pay with a guess for r_{x_i}, which is τ bits long (it's important to note one exception: the first guess r_{x_i} is not on a fresh random challenge x_i, because there was a response from Bob—namely, $r_{x_{\lambda_c+1}}$—after the challenge x_i was issued). If this guess is ever incorrect, Alice will abort, and at that point Eve will have to produce y_{live} on her own, using all the information she has gathered from Alice's responses.

To make this reasoning more precise, we will consider average minentropy of $r_{y_{\text{live}}}$ conditioned on the challenge y_{live}. We will assume, for ease of reasoning, that Alice's challenges x_i are all preselected at the beginning of the protocol, and merely revealed in the correct round. Let Tr be the entire view of Eve until the moment Bob sends y_{live}, and also include in Tr all the x_i challenges from Alice and correct responses $\mathsf{Ext}(w; x_i)$, even those not yet seen by Eve. Define event G the same way it is defined in Lemma 10. Condition on G. Then, because G holds, W has sufficient minentropy for Ext' to work in the last round. Thus, by Lemma 3, the average minentropy of $r_{y_{\text{live}}}$ conditioned on a uniformly random y_{live} and a fixed Tr is at least $\kappa + 3 + \tau + \lambda_c$.

Assume that at the time when Bob issued this challenge, Alice had sent j messages. Then the game Eve is playing is the following: she has to guess the variable $r_{y_{\text{live}}}$ of average min-entropy at least $\kappa+3+\tau+\lambda_c$ by Lemma 3. Eve may be able to reduce this average minentropy by at most τ by receiving r_j of length τ (see Lemma 1). She may further be able to decrease it by τ each time she guesses an unrelated variable of minentropy $\tau - 1$ (namely, the value r_{x_i} for $j < i = \lambda_c$), and receives r_{y_i} of length τ. Note that information on the correctness of her guess r_{x_i} doesn't reduce the average min-entropy of $r_{y_{\text{live}}}$, because the correct guess was already included in Tr when we defined G. If, at any point, this guess for r_{x_i} is incorrect, then Eve doesn't get to play the game any further, and has to guess $r_{y_{\lambda_c+1}}$. Her probability of success, conditioned on G, is thus $2^{-(\kappa+3+\tau+\lambda_c)+\tau}(2^{-(\tau-1)}2^{\tau} + 2^{-2(\tau-1)}2^{2\tau} + \cdots + 2^{-(\lambda_c-1)(\tau-1)}2^{(\lambda_c-1)\tau}) \le 2^{-\kappa-3}$. Furthermore, $\Pr[\bar{G}] \le 2^{-\kappa-3}$ (see proof of Lemma 10 and note that the condition required for that Lemma, namely that $\widetilde{\mathbf{H}}_\infty(w \mid \mathsf{Tr}) \ge 2\kappa + t$, holds by the same reasoning as in Section 5.4.1, since Tr contains the same information). We thus get that the overall probability of success in case 2 is bounded by $2^{-\kappa-2}$.

6 Conclusions and Subsequent Work

We have presented a protocol that allows two parties sharing a low entropy secret to extract a shared key of optimal length—if the shared secret has entropy m, then the length of the extracted key is $m - \Theta(\kappa)$, where κ is the security parameter. We obtain our result through a somewhat unexpected application of edit distance codes. While our protocol has optimal entropy loss, it has a round complexity of $\Theta(\kappa)$. On the other hand, Dodis and Wichs [11] showed nonconstructively, through the use of nonmalleable extractors, that there exists a protocol with both optimal entropy loss and optimal round complexity (2 rounds, which is shown to be necessary by [11]). Until recently, the problem of finding a polynomial-time protocol with optimal round complexity and optimal entropy loss was open. Li [18] made progress on the open problem by showing two-round protocol for w whose entropy rate is an arbitrary constant; his work was using explicit constructions of nonmalleable extractors and novel protocol techniques shown in [9, 7, 17] (which achieve optimal entropy loss when the entropy rate of w is at least 1/2). In other related work, the work of [3] studies the reusability of authentication schemes by using weak long-term keys to derive one-time session keys.

7 Acknowledgments

We thank Alexandr Andoni, Yevgeniy Dodis, and Madhu Sudan for helpful discussions. We also thank the anonymous reviewers of STOC 2010 for their detailed comments. We thank Gil Segev for pointing out a subtle flaw in the definition of privacy amplification which appeared in an earlier version of this paper.

References

[1] C. Bennett, G. Brassard, and J.-M. Robert. Privacy amplification by public discussion. *SIAM Journal on Computing*, 17(2):210–229, 1988.

[2] C. H. Bennett, G. Brassard, C. Crépeau, and U. M. Maurer. Generalized privacy amplification. *IEEE Transactions on Information Theory*, 41(6):1915–1923, 1995.

[3] N. J. Bouman and S. Fehr. Secure authentication from a weak key, without leaking information. In *EUROCRYPT*, pages 246–265, 2011.

[4] X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith. Secure remote authentication using biometric data. In R. Cramer, editor, *Advances in Cryptology—EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 147–163. Springer-Verlag, 2005.

[5] J. Carter and M. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18:143–154, 1979.

[6] N. Chandran, B. Kanukurthi, R. Ostrovsky, and L. Reyzin. Privacy amplification with asymptotically optimal entropy loss. In L. J. Schulman, editor, *STOC*, pages 785–794. ACM, 2010.

[7] G. Cohen, R. Raz, and G. Segev. Non-malleable extractors with short seeds and applications to privacy amplification. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:96, 2011.

[8] Y. Dodis, B. Kanukurthi, J. Katz, L. Reyzin, and A. Smith. Robust fuzzy extractors and authenticated key agreement from close secrets. 2012. To appear.

[9] Y. Dodis, X. Li, T. D. Wooley, and D. Zuckerman. Privacy amplification and non-malleable extractors via character sums. In R. Ostrovsky, editor, *FOCS*, pages 668–677. IEEE, 2011.

[10] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139, 2008. arXiv:cs/0602007.

[11] Y. Dodis and D. Wichs. Non-malleable extractors and symmetric key cryptography from weak secrets. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, pages 601–610, Bethesda, Maryland, 31 May–2 June 2009.

[12] V. Guruswami, C. Umans, and S. P. Vadhan. Unbalanced expanders and randomness extractors from Parvaresh-Vardy codes. In *IEEE Conference on Computational Complexity*, pages 96–108, 2007.

[13] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. Construction of pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[14] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Extracting correlations. In *50th Annual Symposium on Foundations of Computer Science*, Atlanta, Georgia, Oct. 2009. IEEE.

[15] B. Kanukurthi and L. Reyzin. Key agreement from close secrets over unsecured channels. In A. Joux, editor, *Advances in Cryptology—EUROCRYPT 2009*, volume 5479 of *LNCS*. Springer, 2009.

[16] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707–710, Feb. 1966.

[17] X. Li. Design extractors, non-malleable condensers and privacy amplification. In *STOC*, pages 837–854, 2012.

[18] X. Li. Non-malleable extractors, two-source extractors and privacy amplification. Cryptology ePrint Archive, Report 2012/188, 2012. `http://eprint.iacr.org/`.

[19] U. Maurer. Information-theoretically secure secret-key agreement by NOT authenticated public discussion. In W. Fumy, editor, *Advances in Cryptology—EUROCRYPT 97*, volume 1233 of *LNCS*, pages 209–225. Springer-Verlag, 1997.

[20] U. Maurer and S. Wolf. Privacy amplification secure against active adversaries. In B. S. Kaliski, Jr., editor, *Advances in Cryptology—CRYPTO '97*, volume 1294 of *LNCS*, pages 307–321. Springer-Verlag, 1997.

[21] U. Maurer and S. Wolf. Secret-key agreement over unauthenticated public channels — Part III: Privacy amplification. *IEEE Trans. Info. Theory*, 49(4):839–851, 2003.

[22] N. Nisan and D. Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–53, 1996.

[23] J. Radhakrishnan and A. Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM Journal on Computing*, 13(1):2–24, 2000.

[24] R. Renner and S. Wolf. Unconditional authenticity and privacy from an arbitrarily weak secret. In D. Boneh, editor, *Advances in Cryptology—CRYPTO 2003*, volume 2729 of *LNCS*, pages 78–95. Springer-Verlag, 2003.

[25] R. Renner and S. Wolf. The exact price for unconditionally secure asymmetric cryptography. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology—EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 109–125. Springer-Verlag, 2004.

[26] L. J. Schulman and D. Zuckerman. Asymptotically good codes correcting insertions, deletions, and transpositions. *IEEE Transactions on Information Theory*, 45(7):2552–2557, 1999.

[27] D. R. Stinson. Universal hash families and the leftover hash lemma, and applications to cryptography and computing. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 42:3–31, 2002. Available at `http://www.cacr.math.uwaterloo.ca/~dstinson/publist.html`.

[28] S. Vadhan. *Pseudorandomness*. Foundations and Trends in Theoretical Computer Science. Now Publishers, 2012. To appear.

[29] S. Wolf. Strong security against active attacks in information-theoretic secret-key agreement. In K. Ohta and D. Pei, editors, *Advances in Cryptology—ASIACRYPT '98*, volume 1514 of *LNCS*, pages 405–419. Springer-Verlag, 1998.