Perfectly Secure Password Protocols in the Bounded Retrieval Model

Giovanni Di Crescenzo[1], Richard Lipton[2], and Shabsi Walfish[3]

[1] Telcordia Technologies, Piscataway, NJ, USA
giovanni@research.telcordia.com
[2] Georgia Institute of Technology, Atlanta, GA, USA
rjl@cc.gatech.edu
[3] New York University, New York, NY, USA
walfish@cs.nyu.edu

Abstract. We introduce a formal model, which we call the *Bounded Retrieval Model*, for the design and analysis of cryptographic protocols remaining secure against intruders that can retrieve a limited amount of parties' private memory. The underlying model assumption on the intruders' behavior is supported by real-life physical and logical considerations, such as the inherent superiority of a party's local data bus over a *remote* intruder's bandwidth-limited channel, or the detectability of voluminous resource access by any *local* intruder. More specifically, we assume a fixed upper bound on the amount of a party's storage retrieved by the adversary. Our model could be considered a non-trivial variation of the well-studied Bounded Storage Model, which postulates a bound on the amount of storage available to an adversary attacking a given system.

In this model we study perhaps the simplest among cryptographic tasks: user authentication via a password protocol. Specifically, we study the problem of constructing efficient password protocols that remain secure against offline dictionary attacks even when a large (but bounded) part of the storage of the server responsible for password verification is retrieved by an intruder through a remote or local connection. We show password protocols having satisfactory performance on both *efficiency* (in terms of the server's running time) and *provable security* (making the offline dictionary attack not significantly stronger than the online attack). We also study the tradeoffs between efficiency, quantitative and qualitative security in these protocols. All our schemes achieve *perfect security* (security against computationally-unbounded adversaries). Our main schemes achieve the interesting efficiency property of the server's lookup complexity being much smaller than the adversary's retrieval bound.

1 Introduction

Partially motivated by the recent press attention to intrusions from both external attackers and insiders into databases containing highly sensitive information (e.g., [27, 28]), we initiate a rigorous study of cryptographic protocols in the presence of intruders, under a novel and reasonable assumption on their power. This leads us to define a new formal model which we call the Bounded Retrieval Model since we assume a bound on the amount of a party's stored data that can be retrieved by the adversary. In practice, this bound would be due to both physical and logical considerations, as we now explain. With respect to internal attackers, this bound may result from the capabilities of a simple Intrusion Detection System (IDS), which can easily monitor any large and repeated access to the party's stored data. With respect to external attackers, this bound is further minimized as a consequence of the inherent gap between the (smaller) availability of bandwidth due to physical limits and the (larger) availability of storage memory: an attacker needing a large amount of time to retrieve large amounts of sensitive data will most likely be unable to maintain an unauthorized connection for enough time without being detected.

Our model could be considered a non-trivial variation of the well-studied Bounded Storage Model, introduced in [13] (see, e.g., [12, 14] and references therein for further studies of several cryptographic tasks, such as key-agreement, encryption, oblivious transfer, time-stamping, etc). This model postulates a fixed upper bound on the storage capacity (but no bound at all on the computational power) of the adversary attacking a cryptographic protocol. Thus, with respect to the standard model used in the security analysis of most cryptographic primitives, where the adversary is assumed to have a polynomial upper bound on both storage and computational power, this model achieves much higher security at the expense of a stronger assumption on the adversary's *storage* capability. Analogously, our model also avoids upper bounds on the computational power of the adversaries at the expense of a stronger assumption on the adversary's *retrieval* capability, which we argued before as being supported by reasonable considerations.

In this paper we use this model to analyze possibly the simplest cryptographic task: entity authentication via password verification, which we will briefly call a 'password protocol' in the rest of the paper.

Password Protocols. Despite their often noticed weaknesses, password protocols remain the most widely used method for authenticating computer users. In traditional UNIX-like password schemes, the server stores some one-way function of users' passwords in a single password database. In order to verify a login attempt, the server simply computes the same one-way function on a putative password supplied by the user attempting to login, and compares it to the stored value in the database. If the values match, the user is allowed to log in. An adversary trying to impersonate an authorized user can always try an *"online attack"* by entering different passwords in correspondence to the user's login name. However, if the user's password is chosen with enough entropy or randomness, each attempt is extremely unlikely to succeed, and modern servers are programmed to close the authentication session after just a few unsuccessful attempts. Unfortunately, the password database itself is typically small, and can be quickly and easily retrieved by any attacker capable of minimally compromising the security of the server. Although the password database does not directly contain any of the user's passwords, it opens up the possibility of an *"offline dictionary attack"* to the adversary. In such an attack, the adversary can utilize the information contained in *any single record* of the password database by attempting to apply the appropriate one-way function to every word in a dictionary in the hopes that it will match the content of that record (due to the users' tendency to supply dictionary words for their passwords). Although too large to be efficiently searched by a human, dictionaries are typically small enough so that they are efficiently searchable by a computer, thus making this offline dictionary attack quite feasible.

In this paper we explore the following simple but intriguing question: can we design a scheme so that an adversary is required to access *many records* when trying to carry out these attacks? Storage is a static, cheap resource, that is very available today (such that a server can easily provide it in huge quantities). External bandwidth, a time-dependent resource, is certainly much less available than storage. Furthermore, the bandwidth available to a remote or local attacker may be easily controlled by physical means (or even by monitoring traffic at the server's interface to the outside world). By using this gap between server's storage capacity and the adversary's ability to retrieve stored data, we show how to realize a significant server's security advantage over the adversary, thus making the off-line dictionary attack just slightly more powerful than the (practically unsuccessful) online attack.

Analysis in the Bounded Retrieval Model. Intuitively, we propose to construct password database files that are so large that either (1) they cannot be retrieved in their entirety by a local or remote intruder in any reasonable time (due to access or bandwidth limitations), or (2) any such huge retrieval operation is easily detected. Note that (2) can be obtained using very simple and efficient intrusion detection mechanisms (see, e.g., [1] for a survey and [5] for a theoretical model of intrusion detection). Specifically, a huge retrieval would be considered an anomalous event, thus triggering actions such as closing the adversary's access port or preventing access to the storage area from any insider. Furthermore, note that there are several typical scenarios where (1) can be true. The simplest is clearly that of an adversary with relatively limited bandwidth. In fact, this limitation is already present in existing networks, as even the high bandwidth connections commonly available today may require minutes to transfer modest data amounts such as 1 gigabyte. (See Appendix A for detailed numerical examples.) As another typical scenario, assume the server is distributing the password database in several locations and that the adversary is either unaware of the position of some of them, or cannot physically access some of them.

Formally, we place a bound on the quantity of information from the server's storage area that is retrieved by the adversary during an attack. Analogously to [13], security in our model can be information theoretic in nature, which is quite desirable due to the brute-force nature of off-line dictionary attacks. We consider two main classes of attacks in this model: (1) static retrievals, modeling the case where the adversary must pre-select the data he wishes to collect prior to the beginning of the actual data retrieval phase; and (2) adaptive retrievals, modeling an adversary selecting each single location to be retrieved based on the content of all previously retrieved locations. In both cases the total information retrieved by the adversary is bounded by a fixed parameter. Each of these two classes of attacks models a real world scenario. For example, static intrusions model any situation where the adversary may receive blocks of data chosen independently of their contents, such as data recovered from a damaged and discarded hard disk. Adaptive intrusions model the most general scenario, where the adversary may have arbitrary access to data blocks of its choice, such as retrieving data interactively from an insecure network file server (for example, via FTP). Although it would seem that an adversary, if possible, would always choose to perform an adaptive retrieval attack, one should keep in mind that adaptivity also requires the adversary to expend time to examine the information that is being retrieved and therefore may actually provide the adversary with less information than in the case of a static retrieval. We note that in this model access to the entire data file by the adversary is not ruled out, as it can be easily prevented using intrusion detection techniques. On the other hand, we caution the reader that the model does not include the case in which the server is totally compromised by an attacker, where the latter would actually be able to directly observe the passwords received from users during their login attempts anyway.

Our results. In designing password protocols in the bounded retrieval model, we pay attention to various parameters for security (e.g., the adversary's advantage over the online attack success probability, and the adversary's retrieval strategy) and efficiency (e.g., the server's lookup complexity). This allows us to appropriately set target goals for both. To that purpose, it is useful to keep in mind the following two important issues about parameters:

Adversary's advantage vs. online attack success probability. Although it is certainly desirable to have a password scheme with 0 or exponentially small adversary's advantage probability, in practice it is essentially just as desirable to have a password scheme with the adversary's advantage comparable to the online attack success probability (as the overall attacker's success probability is the sum of the two values).

Server's running time vs. adversary's retrieval bound. Although intuitively it would seem easier to design provably secure password schemes where at each user registration or verification the server reads more locations than the adversary is ever allowed, this severely restricts the efficiency of the scheme and its practical applicability.

Summarizing, the combination of efficiency and security properties we desire requires the adversary's advantage to be provably comparable to the online attack success probability, and the server's running time (as measured by the number of data blocks read) to be significantly smaller than the adversary's.

Towards this goal, our first result is a lower bound on the advantage of the adversary, which, among other things, relates the advantage to the adaptivity of the server's lookup strategy. We then start by exploring what schemes can be constructed using well-known cryptographic tools such as secret sharing schemes and all-or-nothing transforms. These result in schemes $\mathcal{P}_1$ and $\mathcal{P}_2$, having the smallest possible adversary's advantage (0 and exponentially small, respectively) in the strongest adversarial model (adaptive attacks), but requiring the server's running time to be larger than the adversary's retrieval bound.

Our main protocols, denoted as $\mathcal{P}_3$ and $\mathcal{P}_4$, achieve high efficiency in that the server's lookup complexity is much smaller than the adversary's retrieval limit. Protocol $\mathcal{P}_3$ is based on dispersers and pairwise-independent hash functions, and guarantees both security against adaptive adversaries and that the adversary's advantage is not significantly larger than the online attack success probability. Because of our previous lower bound, this protocol achieves an *optimal* bound on the adversary's advantage (up to a constant) for typical values of the adversary's retrieval bound (e.g. whenever the adversary's retrieval bound is a constant fraction of the storage). Protocol $\mathcal{P}_4$ is based on t-wise independent hash functions and strong extractors and achieves security against static retrieval attacks, ensuring exponentially-small adversary's advantage without any computational assumption. Protocols $\mathcal{P}_3$ and $\mathcal{P}_4$ can be combined, resulting in a single scheme that simultaneously enjoys both of their desirable security properties.

A more detailed account of our protocols' properties is in Figure 1. We note that none of our protocols is proved secure by assuming the existence of random oracles (we thus removed this assumption from one protocol in [7]).

Related work. The Bounded Retrieval Model is a novel variation of the Bounded Storage Model of [13], and furthermore in some of our solutions we use strong extractors, which are a common tool for protocols in the Bounded Storage Model. However, we point out that solutions and analysis for our password protocol problems have to address quite non-trivial obstacles, even given such tools. A bounded-retrieval notion similar to ours was also used implicitly in [10], in the context of smart cards with slow memory access. Whereas [10] studied the problem of token based authentication in a bounded retrieval context, we consider the problem of password based authentication.

The importance of securing the server's password file has been well-known for many years, and is discussed in detail, for instance, in [20, 8]. Various aspects of password protocols have been studied in the security literature. One important area is that of securing password protocols where the communication goes over an insecure network (e.g., see [9] for schemes based on public-key encryption and [2, 3, 18, 26] for heuristic schemes not using public keys). While this aspect is orthogonal to the server compromise security considered in our work, we stress that many of the cited results can be modularly combined with results in this paper to obtain network password protocols secure against bounded retrieval attacks. Other work on password-related protocols includes well-studied areas like password-authenticated key exchange, that are even farther from the scope of this work.

Protocol name	Adversary's advantage	Adversary's strategy	Server's complexity	Server's strategy	Storage constraints
$\mathcal{P}_1$	0	adaptive	$l > q$	non-adaptive	$n \geq 2td$
$\mathcal{P}_2$	$O(2^{-\lambda})$	adaptive	$l > q$	non-adaptive	$n \geq 2d$
$\mathcal{P}_3$	$O(m^3/(m-q)^2 l 2^d)$	adaptive	$l < q$	non-adaptive	$n \geq 2d+1$
$\mathcal{P}_4$	$(2t+4) \cdot 2^{-\lambda}$	static	$l < q$	adaptive	$n \geq O(\lambda) + 2d$
none	$2^{-\lambda}$	static	$l < q$	non-adaptive	

Fig. 1. Any two protocols in the above table are incomparable, in the sense that each one is better in some features than the other. Specifically, protocol $\mathcal{P}_1$, based on secret sharing, is of interest as it achieves 0 adversary's advantage. Protocol $\mathcal{P}_2$, based on all-or-nothing transforms, is of interest as it achieves exponentially small adversary's advantage while improving storage constraints. Protocol $\mathcal{P}_3$, based on dispersers and pairwise-independent hash functions, is of interest as it achieves security against adaptive adversaries and server's lookup complexity l smaller than the adversary's retrieval bound q. Protocol $\mathcal{P}_4$, based on strong extractors and t-wise independent hash functions, is of interest as it achieves the efficiency property of $\mathcal{P}_3$ as well as exponentially small advantage against static adversaries. At the end of the paper we also discuss a protocol that combines features from protocols $\mathcal{P}_3$ and $\mathcal{P}_4$. The last line in the table points out that achieving exponentially small advantage against adaptive or static adversaries is impossible when the server's lookup strategy is non-adaptive, due to a lower bound in Section 3. Formal definitions, including parameters and performance measures used in the table, are given in Section 2.

2 Model and Formal Definitions

We start by presenting the scenario for password protocols. We discuss the entities involved and the assumed connectivity among them, the phases, the (sub)protocols, and finally the requirements that a password protocol has to satisfy to be declared secure in the bounded retrieval model.

Entities, connectivity, resources. An arbitrary system (or network) containing a number of resources can be accessed locally or remotely through a password protocol controlled by a *server* S. The *users*, denoted as $U_1, U_2, \ldots, U_t$ for some integer t, are any entities that need resources in the system, and thus may require access to it. Although potentially all users are connected to each other as well as to the server through some communication link, for practical purposes, we are interested in password protocols where each user only interacts with the server, and not necessarily at the same time. For simplicity, we will assume that the communication link between each user and the server is private or not subject to attacks, although we note that the model in which this link is also subject to adversarial attacks is of orthogonal focus and can be separately studied (but will not be studied in this paper). The server's storage area contains a *password file*, that we denote as F, with m locations, each containing a record of n bits. We denote as $F[i]$ the content of the i-th location of F and, as $F[L]$ the set $\{F[i] : i \in L\}$.

Subprotocols and Phases. A password protocol can be divided into four main algorithms or subprotocols: a *setup* algorithm, a *password sampling* algorithm, a *registration* algorithm and an *identification* subprotocol.

A setup algorithm, that we denote as Set, is only run by the server. On input a security parameter λ in unary, algorithm Set returns an m-location password file F, for some $m = \text{poly}(\lambda)$ in time at most polynomial in λ.

A password sampling algorithm, that we denote as Sample, is run by users to select their passwords. We will only consider the algorithm Sample that, on input parameter 1^d, returns a uniformly chosen string from $\{0,1\}^d$. (In each of our constructions, by properly using tools such as extractors, we can modularly reduce to this case more general cases such as that of users choosing passwords from a smaller dictionary of strings with known min-entropy.) We will think of the *password length* d as a constant, this being much smaller than the security parameter λ.

The registration algorithm, denoted as Reg, is a possibly probabilistic polynomial time (in n) algorithm that takes as input a user's login name log_i, her password pw_i, and the password file F, and returns an output F for S. Here, log_i denotes a login name somehow generated by S or by U_i (we won't deal with the details on how this happens but just assume that each user has a distinct login name that is, for simplicity of notation, d-bit long), and the output F is an updated version of the password file.

During an identification subprotocol, a user U_i sends both log_i and pw_i to S, which runs a deterministic polynomial time (in λ) algorithm Ver on input log_i, pw_i, F, in addition to the various parameters and all login names, and returns *accept* (briefly, 1) or *reject* (briefly, 0), according to whether the user has been positively identified or not.

We will denote a *password protocol* as a quadruple of probabilistic algorithms $\mathcal{P} = (\text{Set}, \text{Sample}, \text{Reg}, \text{Ver})$, and we will assume, for simplicity, that an execution of $\mathcal{P}$ can be divided into three phases: first, an *initialization phase*, where the server runs the setup algorithm; then, a *registration phase*, where each among the t users $U_1, \ldots, U_t$ chooses a password using Sample and runs subprotocol Reg with server S; finally, an *identification phase*: at any time, any among $U_1, \ldots, U_t$ can run the identification subprotocol with S. We denote as $Param$ the list of parameters (represented in unary) associated with $\mathcal{P}$, that can be any subset among: the *password length* d, the *number of users* t, the *number of locations* m, the *location size* n, the *security parameter* λ, which have been defined above, and the *lookup complexity* l, and the *retrieval bound* q, which will be defined later.

Correctness requirement. A basic requirement we expect from a password protocol is that, at any time, a server positively identifies previously registered users.

Definition 1. *Let* $\mathcal{P} = (\text{Set}, \text{Sample}, \text{Reg}, \text{Ver})$ *be a password protocol with parameters* $Param = (d, t, m, n, l, q)$. *The* correctness *requirement for* $\mathcal{P}$ *is as* *follows: for each* $j \in \{1, \ldots, t\}$*, and any login-name* log_j*, it holds that*

$$\Pr\left[\begin{array}{r} F \leftarrow \textsc{Set}(1^n); \{pw_i \leftarrow \textsc{Sample}(1^d); F \leftarrow \textsc{Reg}(log_i, pw_i, F)\}_{i=1}^t : \\ \textsc{Ver}(Param, \{log_i\}_{i=1}^t, log_j, pw_j, F) = 1 \end{array}\right] = 1.$$

Bounded Retrieval security requirement. All our results consider an adversary A that is *not* time-bounded. This is not only an interesting byproduct of our results but also an especially desired requirement in our model, as we want to withstand adversaries who can run dictionary attacks. Moreover, the adversary is given knowledge of all users' login names, and is allowed to retrieve up to q entries from the server S's password file F. We consider two levels of adaptivity (that is, dependency on the content of F) that the adversary can use in choosing the q entries from F. In practical applications, the adaptivity level plays an important role, as adaptivity may slow down the retrieval rate for the adversary. Specifically, we will restrict the adversarial attack to one of the following two types:

1. *Static Retrieval:* First, the adversary must select a set of at most q locations $L = \{l_1, \ldots, l_q\}$, without observing any of the data in the password file F. Then the adversary is given the contents $F[l_1], \ldots, F[l_q]$ of the selected locations. Finally, the adversary returns a pair (log_j, pw'_j), trying to guess the password of user U_j.
2. *Adaptive Retrieval:* As before, except each of the q locations can be selected by the adversary after seeing the contents of the previously selected ones.

Formally, for $x \in \{$ static, adaptive $\}$, we say that a *bounded retrieval attack* of type x is successful if the experiment $E_x^{\mathcal{P},A}$ returns 1, where

1. if $x = static$ then $E_x^{\mathcal{P},A} = E_{static}^{\mathcal{P},\mathsf{A}}$
2. if $x = adaptive$ and $E_x^{\mathcal{P},A} = E_{adaptive}^{\mathcal{P},\mathsf{A}}$,

and, for all parameters $Param = (d, t, m, n, l, q)$ described in unary and all login names $\{log_1, \ldots, log_t\}$, the experiments are defined as follows (here, the notation $y \leftarrow Alg(x_1, x_2, \ldots)$ denotes the process of running the (possibly probabilistic) algorithm Alg on input $x_1, x_2, \ldots$ and the necessary random coins, and obtaining y as output):

$E_{static}^{\mathcal{P},\mathsf{A}}(Param, \{log_i\}_{i=1}^t)$
1. $F \leftarrow \textsc{Set}(1^n)$
2. for $i = 1, \ldots, t$,
   $pw_i \leftarrow \textsc{Sample}(1^d)$
   $F \leftarrow \textsc{Reg}(log_i, pw_i, F)$
3. $p \leftarrow (Param, \{log_i\}_{i=1}^t)$
4. $\{l_1, \ldots, l_q\} \leftarrow A(p)$
5. $(log', pw') \leftarrow A(p, \{l_i, F[l_i]\}_{i=1}^q)$
6. if $\textsc{Ver}(p, log', pw', F) = 1$ then
   return: 1
   else **return:** 0.

$E_{adaptive}^{\mathcal{P},\mathsf{A}}(Param, \{log_i\}_{i=1}^t)$
1. $F \leftarrow \textsc{Set}(1^n)$
2. for $i = 1, \ldots, t$,
   $pw_i \leftarrow \textsc{Sample}(1^d)$
   $F \leftarrow \textsc{Reg}(log_i, pw_i, F)$
3. $i \leftarrow 0$; $p \leftarrow (Param, \{log_i\}_{i=1}^t)$
4. repeat
   $i \leftarrow i + 1$
   $l_i \leftarrow A(p, \{l_j, F[l_j]\}_{j=1}^{i-1})$
   until $i = q$
5. $(log', pw') \leftarrow A(p, \{l_i, F[l_i]\}_{i=1}^q)$
6. if $\textsc{Ver}(p, log', pw', F) = 1$ then
   return: 1 else **return:** 0.

We are now ready to define the security requirement for password protocols in the bounded retrieval model.

Definition 2. *Let* $\mathcal{P} = (\textsc{Set}, \textsc{Sample}, \textsc{Reg}, \textsc{Ver})$ *be a password protocol with parameters* (d, t, m, n, l, q)*. For* $x \in \{$ *static, adaptive* $\}$*, we say that* $\mathcal{P}$ *is* ϵ-secure against a bounded retrieval attack of type x *if for any algorithm* A*, all login names* $\{log_1, \ldots, log_t\}$*, and any* $j = 1, \ldots, t$*, it holds that*

$$\Pr\left[\, b \leftarrow E_x^{\mathcal{P},A}(Param, \{log_i\}_{i=1}^t) \,:\, b = 1 \wedge log' = log_j \,\right] \leq \frac{1}{2^d} + \epsilon.$$

Remarks. In the above definitions we only have addressed the most basic and practically relevant variant of a number of definitions that one could come up with. For instance, one could strengthen the security requirement by defining an adversary to be successful even if it obtains any nonzero information about the joint values of all passwords, rather than just being able to successfully login, as defined above. (This requirement seems stronger than what's desired in practice.)

Performance Metrics. In addition to the above different adversarial models, when designing password protocols secure under bounded retrieval attacks, we also consider various performance metrics, which we will now discuss in detail. In the rest of the paper we will present a lower bound on the *adversary's advantage*, denoted as ϵ, and protocols that exhibit tradeoffs between all these metrics, in the effort of balancing their security and efficiency.

Time, lookup strategy, storage complexity. An obviously important metric is the *time complexity* of algorithms $\textsc{Set}, \textsc{Reg}$ and $\textsc{Ver}$; in particular, we will pay attention to the (possibly parallel) time complexity of $\textsc{Ver}$, as it is run more frequently in applications. Additionally, we will pay special attention to the *lookup strategy* of algorithm $\textsc{Ver}$, and specifically, to whether it is *adaptive* or *non-adaptive*; that is, based on location content or not. Also related to time complexity is the *storage complexity*; that is, the amount of storage used by the server during the initialization phase. Although storage is today an easily available resource, we will ensure that even a large increase in the storage complexity does not make the time complexity impractical.

Lookup complexity. Additionally, we will pay special attention to the *lookup complexity* of algorithms $\textsc{Reg}$ and $\textsc{Ver}$, which we denote as l, and defined as the maximum number of locations from F that is read or written by either algorithm $\textsc{Reg}$ during its execution on an input log_i, pw_i, F or algorithm $\textsc{Ver}$, when run on an input log, pw, F, in addition to all parameters and login-names. We will assume, without loss of generality, that this number is the same for all inputs to $\textsc{Reg}$ and $\textsc{Ver}$. (We note that all algorithms $\textsc{Reg}, \textsc{Ver}$ can be simply modified so that this holds).

Adversary's breaking advantage. Our model is of information-theoretic nature, as we will consider security against adversaries that are not time-bounded. Therefore, we will be interested in constructions that achieve *adversary's advantage* ϵ either $= 0$ or exponentially small (in the security parameter λ). Additionally, given that an on-line attack is always available in practice to an adversary, we will be interested in constructions that achieve $\epsilon = O(2^{-d})$, where d is the length of a password. (Note that 2^{-d} may not be exponentially small in the security parameter.)

Lookup complexity vs. Retrieval Bound. Given lookup complexity l and retrieval bound q for the adversary, it is of interest to achieve constructions that have the smallest possible value for l and the highest possible for q, in combination with satisfactory performance on the above metrics.

3 A lower bound on the adversary's advantage

We present a lower bound on the security of password protocols having lookup complexity smaller than the adversary's retrieval bound. This will be used to prove the protocol in Section 5 optimal up to a multiplicative constant.

Some definitions. Let $\mathcal{P} = (\textsc{Set}, \textsc{Sample}, \textsc{Reg}, \textsc{Ver})$ be a password protocol and let l denote the lookup complexity of the verification subprotocol $\textsc{Ver}$. We now define t distributions LocD_j, for $j = 1, \ldots, t$, where each LocD_j is the distribution of the locations in F accessed by the algorithm $\textsc{Ver}$ on fixed input $(Param, \{log_i\}_{i=1}^{t}, log_j, pw_j, F)$ generated as in experiment $E_{static}^{\mathcal{P},A}$. Formally, we first define algorithm $\mathrm{L}\textsc{Ver}$ as the algorithm that, given an input $(Param, \{log_i\}_{i=1}^{t}, log_j, pw_j, F)$, returns the set L of locations from F accessed during an execution of algorithm $\textsc{Ver}$ on the same input. Then we can define, for $j = 1, \ldots, t$, the distribution LocD_j as

$$\{\text{run steps } 1,2 \text{ of } E_{static}^{\mathcal{P},A}\,;L \leftarrow \mathrm{L}\textsc{Ver}(Param, \{log_i\}_{i=1}^{t}, log_j, pw_j, F) \;:\; L\};$$

that is, the distribution of locations read by the server during a login by U_j. Note that both $\textsc{Ver}$ and $\mathrm{L}\textsc{Ver}$ are deterministic algorithms, and therefore the actual probability space for distribution LocD_j is given by the randomness contained in the public file F obtained during the execution of experiment $E_{static}^{\mathcal{P},A}$; and, specifically, by how the locations accessed by $\textsc{Ver}$ change, if at all, as an effect of such randomness. For instance, in the case of a non-adaptive lookup strategy, by definition, L can be a single value and therefore the distribution LocD_j trivializes to having a single value in its support. Recall that for a distribution D over support X, the *collision probability* $cp(D)$ is defined as $\sum_{x\in X}(\Pr[\,x' \leftarrow D : x' = x\,])^2$; where we note that if a distribution has a single value in its support, then its collision probability is 1.

Lower Bound Statement and Discussion. Informally, the following lower bound formalizes the intuition that if the servers' lookup complexity is smaller than the adversary's retrieval bound then the larger the amount of adaptivity in the server's lookup strategy, the harder is the adversary's job in finding a password. More formally:

Theorem 1. *Let $\mathcal{P} = (\textsc{Set}, \textsc{Sample}, \textsc{Reg}, \textsc{Ver})$ be a password protocol with parameters (d, n, t, l, m, q), and assume that $\mathcal{P}$ is ϵ-secure against a bounded re-*

trieval attack of static type. If $l \le q$ then it holds that

$$\epsilon \ \ge \max_{j\in\{1,\dots,t\}} \left(\left\lfloor \frac{q}{l} \right\rfloor \cdot \frac{1}{2^d} \cdot cp(\mathrm{LocD}_j) \right),$$

where $cp(\mathrm{LocD}_j)$ is the collision probability of distribution LocD_j defined above.

We note that in the case of non-adaptive lookup strategy from Ver, distribution LocD_j returns a single value, its collision probability is equal to 1, and the bound in the above theorem becomes $\epsilon \ge \lfloor q/l \rfloor \cdot 2^{-d}$, under the hypothesis $l \le q$. As in practice, work in the order of 2^d may be efficiently performed, we derive that non-adaptive strategies for Ver can only result in password protocols ϵ-secure for values of ϵ that are *not* smaller than the on-line attack success probability (e.g., $\epsilon = \Omega(2^{-d})$).

The formal proof of Theorem 1 follows by showing an adversary that can run some modified version of the server's algorithm and always finds a password that would be accepted by the server with probability equal to the lower bound on ϵ in the statement of the theorem. Specifically, the adversary creates a new password file F' identically and independently distributed from the real one; then it starts an off-line dictionary attack by trying several passwords from the dictionary, as follows. For each password, it runs the server's registration and verification algorithms on input F' to determine the set of locations read or written by the server; then, it queries the same set of locations from the real file F, and runs the verification algorithm to see if that password would be accepted by the server. Details of the proof appear in the full version of the paper.

4 Strongly-secure Constructions with Large Lookup Complexity

The purpose of this section is to present two very basic constructions of password protocols secure against bounded retrieval attacks, and show that they achieve very strong security at the expense of requiring an inefficient lookup strategy from the server. Specifically, these constructions achieve essentially the best possible security properties: the adversary's advantage can be 0 in one construction and exponentially small in the other one. Furthermore, these values are achieved against an adaptive adversary. The server's lookup strategy in these constructions is also non-adaptive. On the other hand, in both constructions the server's lookup complexity is larger than the adversary's retrieval bound. In fact, in one of the two constructions the server has to access the entire password file in order to verify a user's identity. (Constructions in the next sections will lower the server's lookup complexity and at the same time obtain desirable security properties.) Formally, we obtain the following:

Theorem 2. *For $i = 1, 2$ there exist protocols $\mathcal{P}_i = (\mathrm{Set}_i, \mathrm{Sample}_i, \mathrm{Reg}_i, \mathrm{Ver}_i)$ with parameters $(n, t, d; m_i, q_i, l_i)$, that are ϵ_i-secure against a bounded retrieval attack of adaptive type, and such that*

1. $\epsilon_1 = 0$, $m_1 \geq l_1 \geq q_1 + 1$ *and* $n \geq 2td$.
2. $\epsilon_2 = O(2^{-\lambda})$, $m_2 = l_2 \geq q_2 + \min(\lambda, o(q_2))$, *and* $n \geq 2d$.

Note that in both constructions $l_i \geq q_i$. For practical applications, the fact that the server's lookup complexity is large constrains the size of the password file so that it is not very large (or otherwise the identification phase would not be efficient). As a consequence, the adversary's retrieval bound cannot be large either, which restricts the applicability of these schemes to settings where the adversary has a small retrieval rate (e.g., if the adversary has a slow connection). The two schemes satisfying Theorem 2 are based on secret sharing schemes for threshold access structures, as in [22] (using polynomial interpolation), and on adaptively-secure all-or-nothing transforms, as in [6] (using adaptively-secure exposure-resilient functions). Very informally, in the first scheme, the entire password file contains the shares of a threshold scheme, where the secret is the concatenation of all login names and passwords, and the threshold is set as strictly larger than the adversary's retrieval bound. Analogously, in the second scheme, the password file can be seen as an all-or-nothing transform of the concatenation of all login names and passwords. We provide a formal description of these schemes in the full version of this paper.

5 A Secure Construction with Small Lookup Complexity

The constructions in Section 4 showed how to achieve strong security (in terms of both the adversary's advantage and the attack type) and non-adaptive server lookup at the expense of a large lookup complexity. In this section we start exploring what security we can achieve if we target constructions with low lookup complexity, while still maintaining non-adaptive lookup. The lower bound of Section 3 implies that the best security that can be obtained under this setting is comparable to the security against on-line attack. In the rest of the section we give a construction that achieves this security level and is therefore essentially optimal (up to lower-order multiplicative factors) for this setting. We present a password protocol secure against bounded retrieval attacks, which we also call SCS, since the server's storage algorithm in this protocol is based on three basic actions: Select, Combine and Store. Specifically, on an input consisting of a login and a password, the server carefully selects several locations from the password files, combines their content according to some function, and stores the result of this function as a tag that can be associated with this password. We instantiate the 'select' action of the SCS scheme by using dispersers, and the 'combine' action using a pairwise-independent hash function. Our construction has server's lookup complexity lower than the adversary's retrieval bound, and, moreover, the following properties: adversary's advantage comparable with the security against on-line attack; non-adaptive server's lookup strategy; constant parallel time complexity; and security against adaptive adversaries. Formally, we obtain the following:

Theorem 3. *There exists a password protocol* $\mathcal{P}_3 = (\textsc{Set}, \textsc{Sample}, \textsc{Reg}, \textsc{Ver})$ *with parameters* $Param = (n, t, d, m, q, l)$*, that is* ϵ*-secure against a bounded retrieval attack of adaptive type, and such that, for any* t, d, m, q*, it holds that* $\epsilon = \frac{m^3}{l\cdot(m-q)^2} \cdot \frac{1}{2^d}$*, for* $n \geq 2d + 1$ *and* $l = 2^b$*, where*
$b = \log^2(d) \cdot \mathrm{poly}(\log\log d) + (\log d) \cdot (\log(m/(m-q)))$.

Note that the value of ϵ in the theorem matches (up to a constant) the bound from Theorem 1 in the typical case $q = cm$, for $0 < c < 1$. We also note that the constant factor c here can be made arbitrarily close to 1. We now prove Theorem 3.

A first tool: t**-wise independent hash families.** Informally, t-wise independence requires that for any fixed set of t elements, a uniformly selected function from the hash family will map those elements to t *uniformly distributed* and *independent* outputs. A formal definition of t-wise independent hash functions follows.

Definition 3 (t**-wise Independent Hash Function).** *A family* $\mathcal{H}$ *of functions* $h_w : \{0,1\}^a \to \{0,1\}^b$ *is* t-wise independent *if, for any distinct elements* $x_1, \ldots, x_t \in \{0,1\}^a$*, and any* $r_1, \ldots, r_t \in \{0,1\}^b$*, we have that*

$$\Pr_w[h_w(x_1) = r_1, \ldots, h_w(x_t) = r_t] = (2^{-b})^t$$

A commonly used t-wise independent hash function is defined, when $c = a = b$, by simply evaluating a $t-1$ degree polynomial over $GF(2^c)$. Specifically, define the following family $\mathcal{H}$, where $x, w_1, \ldots, w_t$ are viewed as elements of $GF(2^c)$, the field over which the computation is to be performed:

$$\mathcal{H} = \{h_{w_1,\ldots,w_t} \mid h_{w_1,\ldots,w_t}(x) = \sum_{j=1}^{t} w_j x^{j-1}\}$$

In our constructions we will use this construction of t-wise independent hash families both in the case $a > b$ (in this section, when $t = 2$) and in the case $a < b$ (in the next section, for larger values of t) where, in both cases, we set $c = \max(a, b)$ and we use trivial padding or truncation operations to satisfy length consistencies. We note that a function from this family can be indexed by exactly t strings of c bits each.

A second tool: Extractors and dispersers. Extractors and dispersers were first introduced in [17] and [24], respectively, and have received a significant amount of attention in several areas of computer science, mostly in the derandomization literature, but also in other areas including combinatorics, network theory and security. Both extractors and dispersers are often defined as bipartite graphs, while in this paper it will be easier to use their functional definition, which we now recall.

The *statistical distance* between two distributions D_1, D_2 over the same space S is defined as $sd(D_1, D_2) = \frac{1}{2}\,\Sigma_{x\in S}\,|\Pr[\,x \leftarrow D_1\,] - \Pr[\,x \leftarrow D_2\,]\,|$. We say that distributions D_1, D_2 are δ*-close* if it holds that $sd(D_1, D_2) \leq \delta$. We say that a distribution D is δ-*close to uniform* if it holds that $sd(D,U) \leq \delta$, where U denotes the uniform distribution over the same space S. The *min-entropy* of a distribution D over space S is defined as $H_\infty(D) = \min_x\{-\log_2(\Pr[\,x \leftarrow D\,])\}$.

A function Ext: $\{0,1\}^a \times \{0,1\}^b \to \{0,1\}^c$ is called a (k,δ)-*extractor* if for any distribution D on $\{0,1\}^a$ with min-entropy at least k, the distribution $N(D)$ is δ-close to uniform, where $N(D) = \{x \leftarrow D; e \leftarrow \{0,1\}^b; y \leftarrow \mathrm{Ext}(x,e) : y\}$.

A function Disp: $\{0,1\}^a \times \{0,1\}^b \to \{0,1\}^c$ is called a (k,δ)-*disperser* if for any $A \subseteq \{0,1\}^a$ such that $|A| \geq 2^k$, it holds that $|N(A)| \geq (1-\delta)2^c$, where $N(A) = \{z \,|\, z = \mathrm{Disp}(x,y),\, x \in A,\, y \in \{0,1\}^b\}$.

We refer to [16, 23] for surveys of applications, constructions and related results for extractors and dispersers. (We use the formal definition of dispersers that appears in [16]; other papers such as [23] use a slightly different definition.)

Construction of protocol $\mathcal{P}_3$. The protocol $\mathcal{P}_3 = (\textsc{Sample}, \textsc{Set}, \textsc{Reg}, \textsc{Ver})$ uses a polynomial-time computable function $\textsc{Select} : \{0,1\}^d \times \{0,1\}^d \to [m]^l$, that we later instantiate using extractors, and a family $\mathcal{H}$ of pairwise-independent hash functions $h_w : \{0,1\}^{nl+2d} \to \{0,1\}^n$ (selection of which parameterizes the Reg and Ver algorithms). We first describe algorithms Set, Reg, and Ver, and then one instantiation of the function Select.

Algorithm Set. Formally, algorithm Set, on input parameters d, t, n, l, m, q in unary, returns an m-location password file F, which can be parsed as $F = X \circ T$ with $|X| = m_x$ and $|T| = t$ (and thus $m = m_x + t$). X is initialized as an array of values $X[1], \ldots, X[m_x]$ uniformly chosen from $\{0,1\}^n$, and T is initialized as an empty array of t locations.

Algorithm Reg. The registration algorithm maps a login and a password to a subset of locations in the set of locations containing random elements, combines their content by computing a tag as their sum, and stores the tag. Formally, on input log_i, pw_i, F, algorithm Reg runs the following steps:

1. compute $(loc_1, \ldots, loc_l) = \textsc{Select}(log_i, pw_i)$;
2. compute $tag_i = h_w(log_i|pw_i|X[loc_1]|\cdots|X[loc_l])$,
3. store tag_i into T by setting $T[i] = tag_i$.

Algorithm Ver. The verification algorithm recomputes the tag corresponding to the input login and password and checks that it is equal to the tag stored during the registration phase. Formally, on input $Param, \{log_i\}_{i=1}^t, log', pw', F$, where $F = X|T$, algorithm Ver runs the following steps:

1. compute $(loc'_1, \ldots, loc'_l) = \textsc{Select}(log', pw')$;
2. let $j \in \{1, \ldots, t\}$ be such that $log_j = log'$;
3. if there exists no such j then return: 0 and halt;
4. verify that $T[j] = h_w(log'|pw'|X[loc'_1]|\cdots|X[loc'_l])$;
5. if so, return: 1; else return: 0.

Instantiation of function Select. For our construction we only need to apply dispersers, but since (k,δ)-extractors are also (k,δ)-dispersers (this can be seen by setting D equal to the uniform distribution over subset A), and given that extractors have been much more studied in the literature, we will apply (a certain kind of) extractors. In particular, we are interested in extractors that firstly maximize the parameter c, denoting the extractor output, so that it is as close as possible to the sum of the min-entropy of the source and the number of real random bits used. Secondly, it is of interest to minimize the value of parameter b for that to happen. This choice criterion is based on that of minimizing the adversary's advantage first, and then, further minimizing the server's sequential running time. We note that in this scheme the parallel running time is constant with respect to the lookup complexity l, regardless of which extractor we choose. A recent survey [23] summarizes most known results about extractors, and we can plug in some of the results in Table 1, pp. 11 of [23] to obtain a function SELECT with satisfactory performance. Bearing in mind the aforementioned criterion, we will use the following fact (obtained from Corollary 6.15 of [21]):

Fact 4. [21] *For any* $0 \leq \alpha < a$ *and* $\delta > \exp(-\alpha/(\log^* \alpha)^{O(\log^* \alpha)})$, *there exists an explicit* (k, δ)*-extractor Ext*: $\{0,1\}^a \times \{0,1\}^b \rightarrow \{0,1\}^c$ *such that* $k = a - \alpha$, $b = O(\log^2(a) \cdot \text{poly}(\log\log a) + (\log a) \cdot (\log(1/\delta)))$ *and* $c = k + b - 2\log(1/\delta) - O(1)$.

Informally, we can instantiate SELECT as the function returning all outputs of the above extractor, when given the password as a first input and all possible l values as a second input. (In the graph-based formulation, these would be all neighbors of the node associated with the password). More formally, for any parameters n, t, d, m, q, where $m = m_x + t$, we can instantiate SELECT as follows. For $log \in \{0,1\}^d$ and $pw \in \{0,1\}^d$, we define $\text{SELECT}(log, pw) = (loc_1, \ldots, loc_l)$, where $loc_j = \text{Ext}((log|pw), j)$, for $j = 1, \ldots, l$; algorithm Ext: $\{0,1\}^a \times \{0,1\}^b \rightarrow \{0,1\}^c$ is the (k, δ)-extractor guaranteed from Fact 4, where $\alpha = d$, $a = 2d$; $b = O(\log^2(2d) \cdot \text{poly}(\log\log 2d) + (\log 2d) \cdot (\log(m/(m-q))))$; $c = \log m$; $l = 2^b$; $\delta = 1 - q/m$; and $k = \log m - l - 2\log(m/(m-q)) - O(1)$.

Proving the security of the SCS protocol. Proving the security property of $\mathcal{P}_3$ makes crucial use of the properties of dispersers and of pairwise-independent hash functions. The main intuition is that if the adversary queries q locations from F, possibly using an adaptive querying strategy, even if he tries to run an off-line password attack, he will be able to test only a very small number of passwords. More specifically, we observe the following facts, using the properties of pairwise-independent hash functions: (1) the probability that the server accepts a false password, is very small. Then we observe that the content of the locations queried by the adversary define t partitions of the set of passwords into two sets: the set of passwords that are mapped to locations queried by the adversary and its complement. Furthermore, (2) the size of the first set is small, (i.e., $O(m^3/l(m-q)^2)$), and (3) any password in the second set does not give a significant advantage to the adversary in being successful; where (2) uses the properties of the disperser from Fact 4 and (3) uses the definition of pairwise-independent hash functions. The security property of $\mathcal{P}_3$ follows by combining the three mentioned facts.

6 Strong Security with Small, Adaptive Lookup Complexity

In the previous section we showed that it is possible to construct password protocols secure against bounded retrieval attacks by adaptive adversaries, and simultaneously have low lookup complexity. The adversary's advantage in the previous construction is not significantly larger than the on-line attack success probability, and essentially meets the lower bound in Section 3. In this section we investigate the possibility of achieving even smaller adversary's advantage (say, exponentially small) while maintaining an efficient lookup complexity. Since the server's lookup strategy will be adaptive, the lower bound in Section 3 does not apply. The scheme remains incomparable to the scheme in previous section though, as it is only secure against static adversaries. We call our new scheme HE, for Hashing and Extraction, according to the strategy used by the server's registration algorithm. Formally, we obtain the following:

Theorem 5. There exists a protocol $\mathcal{P}_4 = (\textsc{Set}_4, \textsc{Sample}_4, \textsc{Reg}_4, \textsc{Ver}_4)$ with parameters $Param = (n, t, d, m, q, l)$, that is ϵ-secure against a bounded retrieval attack of static type, and such that, for any t, d, q, m, it holds that
$\epsilon = (2t+3) \cdot 2^{-\lambda}$, $m > q + t \geq [\ l = t + O(d+\lambda)\ ]$ and $n = O(\lambda) + 2d$.

We stress that in $\mathcal{P}_4$ the server uses an adaptive lookup strategy and therefore the exponentially small upper bound on ϵ does not contradict the lower bound of Theorem 1. We now sketch the proof of Theorem 5.

Tools used by our HE protocol. The construction uses two tools: t-wise independent hash functions (see Definition 3), where t is the number of users, and locally computable and strong extractors.

Locally-computable and strong extractors. We recall two additional properties that extractors (defined in Section 5) may satisfy. Intuitively, the definition of strong extractors requires that the extractor's output remains statistically close to random even when conditioned on the value of the random seed; furthermore, the definition of locally computable extractors requires that the extractor reads only a small subset of the bits contained in the (large) input distribution that the entropy is to be extracted from (this is for efficiency reasons only). The formal definitions follow.

A function Ext: $\{0,1\}^a \times \{0,1\}^b \rightarrow \{0,1\}^c$ is called a *strong* (k, δ)-*extractor* if for any distribution D on $\{0,1\}^a$ with min-entropy at least k, the distribution $U(b) \times N(D, U(b))$ is δ-close to distribution $U(b) \times U(c)$, where $N(D, U(b))$ is defined as $\{x \leftarrow D; e \leftarrow \{0,1\}^b; y \leftarrow \mathrm{Ext}(x, e) : y\}$, and, for any z, $U(z)$ denotes the uniform distribution over $\{0,1\}^z$.

An extractor Ext: $\{0,1\}^a \times \{0,1\}^b \rightarrow \{0,1\}^c$ is ℓ-*locally-computable* if for any $R \in \{0,1\}^b$, the value of R uniquely determines the bit locations in $x \in \{0,1\}^a$ used while computing $\mathrm{Ext}(x, R)$ and the number of such locations is at most ℓ.

We will use the following strong and locally-computable extractor, guaranteed from Theorem 8.5 in [25]:

Fact 6 ([25]). *Let ρ, σ be arbitrary constants > 0. For every $a \in \mathbb{N}$, $\delta > \exp(-a/2^{O(\log^* a)})$, $c \leq (1-\sigma)a\rho$, there is an explicit ℓ-locally computable and strong (k, δ) extractor* $\mathrm{Ext} : \{0,1\}^a \times \{0,1\}^b \to \{0,1\}^c$ *such that:*

1. $k = a\rho$
2. $b = \log a + O(\log c + \log(1/\delta))$
3. $\ell = (1+\sigma)kc/n + \log(1/\delta))$

Construction of protocol $\mathcal{P}_4$. We assume, for simplicity, that the algorithm SAMPLE just uniformly and independently selects a password from $\{0,1\}^d$.

Algorithm SET. Let the data block size $n = 2d + \lambda$, where λ is the security parameter. SET initializes a t-location array W with a uniformly chosen t-wise independent hash function $h_w : \{0,1\}^d \to \{0,1\}^n$. SET then initializes X as an array of m_x locations containing uniformly and independently chosen values in $\{0,1\}^n$. Additionally, SET initializes an empty array T with t empty locations of n bits each, and then sets $F = T \circ W \circ X$. Observe that the total number of data blocks in F is $m = 2t + m_x$.

Algorithm REG. The registration algorithm first hashes the login name and the password to a random value using the t-wise independent hash function specified by the W component of F, to produce a seed value R. (Note that this step makes the server's lookup strategy adaptive, as the computation of R depends on the contents of W, and subsequent lookup operations in X will depend on R.) The extractor is applied to the X component of F using the previously computed seed R in order to produce a (nearly) uniform random output. The resulting output R' may then be used as the tag associated with this password, and is stored in the T component of F. Formally, on input i, log_i, pw_i, F, where $F = T \circ W \circ X$, algorithm REG does the following:

1. set $w = (W[1], \ldots, W[t])$ and $R = h_w(log_i \,|\, pw_i)$.
2. set $R' = \mathrm{Ext}(X, R)$;
3. store $tag_i = R'$ in location $T[i]$.

Algorithm VER. The VER algorithm is essentially identical to the REG algorithm, only after computing tag_i, rather than storing it in the $T[i]$ location in F, the value is compared with the previously stored value in $T[i]$, and the result of the comparison is output. The total number of lookups performed by VER is $l = t + \ell + 1$. Formally, VER, on input $Param, \{log_i\}_{i=1}^t, log', pw', F = T \circ W \circ X$, does the following:

1. set $w = (W[1], \ldots, W[t])$ and $R = h_w(log' \,|\, pw')$.
2. let $j \in \{1, \ldots, t\}$ be such that $log_j = log'$;
3. if there exists no such j then return: 0 and halt;
4. set $R' = \mathrm{Ext}(X, R)$;
5. if $T[j] = R'$ then return: 1 else return: 0

Proof that $\mathcal{P}_4$ satisfies Theorem 5. Proving the security property of $\mathcal{P}_4$ makes crucial use of the properties of strong extractors and of t-wise independent hash functions, as follows. We use the properties of strong extractors to show that the first tag is statistically indistinguishable from a uniformly distributed tag, even conditioned on the value of all other tags and on the value of the hash function used to generate the seed for the extractor. In proving that the conditioning on the value of all other tags does not affect the statistical indistinguishability, we use the indistinguishability of the extractor's output from random, even conditioned over the result of a bounded-output function over the extractor's input. In proving that the conditioning on the seed does not affect the statistical indistinguishability, we use the extractor's 'strong' property. Then we replace the first tag with a random tag and repeat the analogous argument over the second tag, etc. (Note that independence of the R values used to compute each tag is guaranteed for up to t users by the t-wise independent hash function.) Finally, we compute an upper bound on the adversary's advantage when all tags are random by computing an upper bound on collisions on the t-wise independent hash function and on the extractor used. A formal proof is available in the full version of the paper.

An extension: Combining protocols $\mathcal{P}_3$ and $\mathcal{P}_4$. Recall that protocol $\mathcal{P}_3$ is secure against adaptive adversaries but allows the adversary to achieve a non-negligible advantage (which is optimal in the setting of adaptive intrusions). Furthermore, protocol $\mathcal{P}_4$ only allows the adversary to achieve at most negligible advantage, but is only secure against static adversaries. We would like to achieve the "best of both worlds" with a single scheme that limits the adversary to a negligible advantage in case of static intrusions, but remains secure even under an adaptive attack.

Fortunately, such a solution is indeed possible. We simply modify the construction of $\mathcal{P}_3$, replacing the input pw_i with the tag_i computed as in protocol $\mathcal{P}_4$. That is, the the final tag_i values computed using $\mathcal{P}_3$ will now be based on "password" inputs taken from the tag_i values computed via $\mathcal{P}_4$ using the user's actual password. It can be shown that the resulting scheme achieves security comparable to that of $\mathcal{P}_4$ under static intrusion attacks, and comparable to that of $\mathcal{P}_3$ under adaptive intrusion attacks.

Acknowledgment. The first author thanks Rajesh Talpade for interesting discussions on intrusion detection. This material is based upon work supported by the Air Force Research Laboratory - Rome Labs under Contract No. FA8750-04-C-0249.

References

1. S. Axelsson. Research in Intrusion-Detection systems: A Survey, in *Technical Report 98–17*, Dept. of Comp. Eng., Chalmers, Univ. of Technology, Goteborg, Sweden, 1998, http://citeseer.ist.psu.edu/axelsson98research.html.
2. S. Bellovin and M. Merrit. Encrypted Key Exchange, in *Proc. of the 1992 Internet Society Network and Distributed System Security Symposium*.
3. S. Bellovin and M. Merrit. Augmented Encrypted Key Exchange, in *Proc. of the 1st ACM Conference on Computer and Communication Security*, pp. 224-250

4. G. R. Blakley. Safeguarding cryptographic keys. In *Proc. of the National Computer Conference*, v.48, pp. 242–268, 1979.
5. G. Di Crescenzo, A. Ghosh, and R. Talpade. Towards a Theory of Intrusion Detection. In *Proc. of European Symposium on Research in computer Security* (ESORICS 2005), vol. 3679 of LNCS, pp. 267-286, Springer-Verlag.
6. Y. Dodis, A. Sahai, A. Smith. On Perfect and Adaptive Security in Exposure-Resilient Cryptography. In *Proc. of EUROCRYPT 2001*, vol. 2045 of LNCS, pp. 301-324. Springer-Verlag.
7. Password Protocols provably secure in the Bounded Retrieval Model, first public version of this work, unpublished draft, April 2005.
8. D.C. Feldmeier and P.R. Karn. UNIX Password Security - Ten Years Later, in *Proceedings of Crypto'89*, LNCS, no. 435, Springer-Verlag, pp. 44-63
9. S. Halevi and H. Krawczyk. Public-key Cryptography and Password Protocols. In *Proc. of the 5th annual ACM conference on Computer and Communications Security*, pp. 122–131, 1998
10. John Kelsey and Bruce Schneier. Authenticating Secure Tokens Using Slow Memory Access. *USENIX Workshop on Smart Card Technology*, USENIX Press, pp. 101–106, 1999.
11. Chi-Jen Lu. Encryption against storage-bounded adversaries from on-line strong extractors. *Journal of Cryptology*, vol. 17, no. 1, pp. 27–42, 2004.
12. Stefan Dziembowski and Ueli Maurer. Optimal Randomizer Efficiency in the Bounded-Storage Model. In *Journal of Cryptology*, vol. 17, no. 1, pp. 5-26.
13. Ueli Maurer. Conditionally-Perfect Secrecy and a Provably-Secure Randomized Cipher. *Journal of Cryptology*, vol. 5, no. 1, pp. 53–66, 1992.
14. Tal Moran, Ronen Shaltiel, Amnon Ta-Shma. Non-interactive Timestamping in the Bounded Storage Model. In *Proc. of CRYPT0 2004*, vol. 3152 of LNCS, pp. 460–476. Springer-Verlag.
15. R. Morris and K. Thompson. Password Security: A Case History, in *Communications of the ACM*, Vol. 22, no. 11, 1979, pp. 594-597.
16. N. Nisan and A. Ta-Shma. Extracting Randomness: A Survey and New Constructions, in *Journal of Computer and System Sciences*, February 1999, vol. 58, no. 1, pp. 148-173(26)
17. N. Nisan and D. Zuckerman. More Deterministic Simulation in Logspace, in *Proc. of ACM STOC 93.*
18. S. Patel. Number theoretic attacks on secure password schemes, in *Proc. of the 1997 IEEE Symposium on Security and Privacy.*
19. Benny Pinkas and Tomas Sander. Securing Passwords Against Dictionary Attacks. *ACM CCS-9: Computer and Communications Security*, Nov., 2002.
20. N. Provos and D. Mazieres, A Future-Adaptable Password Scheme, In *Proceedings of the Annual USENIX Technical Conference*, 1999.
21. O. Reingold, S. Vadhan, and A. Wigderson. Entropy Waves, The Zig-Zag Graph Product, and New Constant-Degree Expanders and Extractors. in *Electronic Colloquium on Computational Complexity TR01-018*; other versions in *Proceedings of FOCS 2000* and *Annals of Mathematics*, vol. 155, pp. 157-187, 2002.
22. A. Shamir. How to Share a Secret. *Communications of the ACM*, Volume 22 , Issue 11 (November 1979)
23. R. Shaltiel. Recent developments in Explicit Constructions of Extractors. *Bulletin of the EATCS*, 77:67–95, 2002.
24. M. Sipser. Expanders, Randomness and Time vs. Space, in *Journal of Computer and System Sciences*, vol. 36, 1988.

25. S. P. Vadhan. On constructing locally computable extractors and cryptosystems in the bounded storage model. *Journal of Cryptology*, vol. 17, no. 1, pp. 43–77, 2004.
26. T. Wu, *The secure remote password protocol*, in Proc. of the 1998 Internet Society Network and Distributed System Security Symposium
27. http://money.cnn.com/2003/02/18/technology/creditcards/
28. http://www.detnews.com/2005/technology/0506/18/tech-219662.htm

A Numerical Examples

Some typical parameters for instantiating protocol $\mathcal{P}_4$ might be as follows. Set $O(\lambda) = 176$ for a dictionary of size $\approx 2^d$, where, say $d = 40$ (this yields a dictionary of approximately 1 trillion words). We have that $n = 2d + O(\lambda) = 80 + 176 = 256 = 2^8$ (assuming a small constant under the O notation). This requires that data be read from storage in chunks not less than 48 bytes in size.

For a system with $t \approx 2^{12} = 4096$ maximum users, we can achieve the following parameters. Letting $m = 2t + \hat{m} = 2^{13} + 2^{35} \approx 2^{35}$, we obtain a total storage requirement of $mn = 2^8 2^{35} = 2^{43}$ bits, or approximately 1 TB (terabyte). It should be noted that 1 terabyte of storage can currently be purchased at very reasonable cost (under \$1000). Given storage of this size, we can safely set $\beta = 0.99$, allowing the adversary to retrieve up to 99 percent of the storage, which is about 990 MB (megabytes) of data. If we limit the server to an outgoing bandwidth of 8192 bits/sec = 1024 bytes/sec, it will take the adversary over 30 years to download that much data. With an outgoing bandwidth of 1024 bytes/sec, the server can process approximately 32 logins/sec. The lookup complexity will be $l = t + O(d + \lambda) \approx 2^{12} + C(40 + 176) \approx 2^{13}$, which is about 8000 blocks of 256-bits each, per login (a total of less than half a megabyte of data).