Intrusion-Resilient Authentication in the Limited Communication Model

David Cash* Yan Zong Ding* Wenke Lee* Richard Lipton*

November 15, 2005

Abstract

We describe a general technique for building authentication systems that resist compromises at the client side. We derive this resistance by storing key information on hardware fast enough for valid use but too slow for an intruder (e.g., a virus) to capture much of the key before being detected and removed. We give formal models for two types of protocols: user authentication and authenticated session-key generation. The first can be used for physical authentication tokens, e.g., used for gaining access to a building. The second can be used for conducting secure remote sessions on laptops that are occasionally infected by viruses. We present and analyze protocols for each of these tasks and describe how they can be implemented. With one example setting of parameters, in the case of user authentication, we are able to guarantee security for 6 months using a device storing 384MB, and in the key generation protocol, a 128GB drive guarantees that an adversary would need 700 days to compromise the key information.

The model for intrusion resilience considered in this paper was first introduced by Dagon et al. [DLL05] and motivated by the bounded storage model for cryptography [Mau92]. Recently Dziembowski [Dzi05] independently developed this model, and studied the same problems as the ones addressed in this paper. Our user authentication protocol is essentially the same as that of [Dzi05], while our authenticated session-key generation protocol builds on that of [Dzi05].

1 Introduction

Robust systems for network security must guarantee resilience to successful compromises and intrusions. Company laptops, for example, often fall prey to Trojan horse viruses that users inadvertently "install" when they travel (without the protection of their company's firewalls). These viruses can persist until a sysadmin removes them, and then all credentials stored on the laptop must be replaced. A malicious virus could steal a user's credentials with a key logger and erase itself, compromising all future use of the credentials until they are replaced.

The threat of such compromises is currently parried by a few general techniques. An administrator can install a software suite that regularly scans the entire system and inspects new software being installed. These systems are not perfect, however, and there is always the possibility that more clever viruses will be designed to get around the software protection. A second technique is to severely limit the functionality of the system, allowing users to only run a few approved applications. For example, a popular and feature-rich Web browser is often not permitted on company computers because of the large number of vulnerabilities it contains. In this situation a user is

*College of Computing, Georgia Institute of Technology, Atlanta, GA 30332. {cdc, ding, wenke, rjl}@cc.gatech.edu.

prevented from using devices, like laptops, to their full capacity and is losing productivity to the threat of viruses.

In this paper, we construct efficient symmetric-key authentication protocols secure against successful intrusion in a model where an adversary is allowed to break into a user's machine, read all the data stored on the user's machine, perform efficient (i.e. polynomial-time) computation on the data, and yet can only transmit a limited amount of information back to the adversary's host machine. We refer to this model as the *limited communication model*. As the adversary is able to break in and access all secret information of a user, it is clear with a moment's thought that achieving security is impossible unless the size of the secret information is large relative to the communication bandwidth of the adversary. Thus, to defeat such an adversary, a long secret key is employed in the model. Bounding the communication complexity of the adversary and employing a long secret key allows one to construct symmetric-key systems where removing an adversary from a machine also removes his ability to impersonate a user.

Although its name is coined here, the limited communication model was first introduced by Dagon et al. [DLL05] in a study of protection of secret data from break-ins. Recently, independent of [DLL05] and this work, the model and some of the results of this paper were also obtained by Dziembowski [Dzi05], who studied the same authentication problems as the ones addressed in this paper.

We find practical applications which realize the limited communication model in an efficient and secure way. Specifically, we observe that special hardware can restrict access to the large key to a rate that is sufficient for valid use but too slow for an attacker to learn enough of the key to mount an attack. We will describe the hardware we employ for each protocol later, but in general we only need a storage device with a governor, that, for example, guarantees the device only answers one query for bits every few seconds. The devices we envision will take the form of a hand-held memory stick for physical authentication or a hard drive on a computer for secure remote sessions.

The limited communication model was partly motivated by, and its technical tools are based on those developed in the *bounded storage model* [Mau92] for cryptography. This model assumes that the adversary is space-bounded, and that a long stream of random bits is publicly available. In recent years very strong results have been achieved in the bounded storage model for private-key encryption [Mau92, CM97, AR99, ADR02, DR02, DM04, Lu04, Vad04] and two-party protocols [CM97, CCM98, Din01, DHRS04, MST04]. See some recent work (c.f. [Vad04, DHRS04, Din05]) for a good account of the model and a survey of the results in the model. More recently, Rabin has given the first implementation of (a close variant of) the bounded storage model by use of a network of servers serving random bits [Rab05].

1.1 Our Results

User Authentication. We first consider the problem of symmetric-key based user authentication, where a user wishes to authenticate himself to a server who shares a secret key with him. The adversary attacking the protocol can break into the user's machine or the server, read the secret information stored there, record the execution of the protocol many times, but can only send a limited amount of information back to the adversary's host machine. At some point in time, the adversary exits the user's machine[1] and attempts to impersonate the user using the information

[1]If the adversary is inside the user's machine at this time, then clearly security is impossible as with all the information on the user's machine the adversary can simply simulate the user.

collected at the adversary's host machine.

We construct a very simple user authentication protocol which is secure in our model. Essentially the same protocol was independently constructed by Dziembowski [Dzi05]. We give the first detailed analysis with efficient parameters that are calculated in Section 5. Our protocol requires little computation from the user or the server, and furthermore is information-theoretically secure (i.e. secure without computational assumptions).

We note that since in the symmetric-key setting one long secret key is needed between every pair of communicating parties, this model may be most suitable in the case where the authentication is between a user whose machine is susceptible to intrusion, and a highly secured server that cannot be broken into. In this case, one may use a pseudorandom generator G and a short random seed s to generate a pseudorandom string $G(s)$. The user stores (very long) $G(s)$, and the secure server, who authenticates many users, stores just the (short) seed s. Using a pseudorandom string instead of a truly random string only degrades the security of the protocol negligibly, provided that the adversary is computationally bounded. The server can generate small portions of the pseudorandom string (as opposed to the entire string) as needed without significant computation.

Authenticated Session Key Generation. We also consider the problem of authenticated session key generation, and our results here build on the work of Dziembowski [Dzi05]. In this problem Alice and Bob share a long secret key and need to establish a session key for each communication session in the presence of an *active* adversary. In addition to reading Alice and Bob's secret after breaking in and recording the transcript of the interaction between them, the adversary is also able to block and modify messages exchanged between Alice and Bob, and inject his own messages into the network. We defer a precise description of the problem to Section 4.

Using a *random oracle*[2] [FS86, BR93], Dziembowski constructed an efficient protocol for authenticated session key generation that is secure in the limited communication model. It is however a fact that a protocol secure in the random oracle model *may not* be secure when the random oracle is replaced by *any* secure cryptographic hash functions [CGH04]. An efficient construction without a random oracle is thus desired.

We construct a variant of Dziembowski's protocol without using a random oracle. Our protocol is only slightly less efficient than the original. Our solution is based on *non-malleable coin tossing* [DDN00], which can be implemented efficiently in our model, and randomness extractors [NZ96].

Applications. We propose two applications of these protocols. The first is secure *physical authentication tokens*, which are hand-held devices used to prove one's identity. Instead of a card with a computing platform on a chip, the user needs only a small storage device that guarantees a fixed transfer rate. Our user authentication protocol involves no computation beyond fetching bits from memory, so currently known powering and timing attacks do not seem to apply. Using our analysis, we are able to show that any adversary must steal the token for a long period of time (say, months or years) before he can impersonate the user, and before that period is over, the token will hopefully be reported missing.

Our second application is for a device that allows laptops to store their key locally, but still resist virus infection. Here we equip laptops with a hard drive that is so slow that a virus infecting the laptop is able to access only a small portion of the key before it is removed by anti-virus software that the user updates (with new signatures) and runs occasionally. The hard drive can be used to

[2] A random oracle is a publicly available hash function that is assumed to be a truly random function.

negotiate authenticated session-keys using our protocol, and we are able to show that an adversary must control the laptop for years before he would be able to impersonate the valid user without the laptop.

1.2 Related Works

The issue of exposed keys has been explored in a number of models. In so-called *exposure-resilient cryptography*, and adversary can directly access most of the original bits of the secret key without effecting the security of the system, c.f. [CDH+00]. Exposure-resilient cryptography is concerned with protecting against the problem of hardware leaking the bits of a short key only once, while in this work we are using large keys that can be accessed for months in an adaptive manner.

Another technique for dealing with key exposure was developed with *forward security* [And97], where keys are updated periodically. In this model, an adversary compromises the system at some point and is unable to break the system for previous time periods that have not been compromised. The concept has turned out to be powerful, and protocols have been found for digital signatures [BM99, AR00], public-key encryption [CHK03], private-key cryptography [BY03], and many of variants of these applications.

One drawback of forward security is that key exposure necessarily invalidates all future use of that key. An approach to fixing this was taken in *key-insulated cryptography* [DKXY02], which updated keys as in forward security models. In this case, however, a user interacted with a physically secure device storing a master key for updates at the end of each period. Key-insulated schemes can then give a stronger version of security, where all past and future periods remain secure after a fixed number of compromises. They also had the interesting property that stealing the master key alone would not compromise security: an adversary would have to steal the master key *and* some temporary keys. This model was later generalized to allow for an arbitrary number of compromises of temporary keys and master keys, as long as both are not exposed during the same period [IR01].

Recently, Dagon et. al. [DLL05] described a model in which personal information was stored on a huge database and addressed the following issue. It is standard practice for Internet merchants to store credit card numbers and protect them with a weak password. This practice exposes users to the risk that once their encrypted information is stolen, it can be decrypted by a brute force attack. Since an encrypted credit card number may only be a few bytes, it may be very difficult to detect when an adversary has stolen a ciphertext.

Dagon et al. suggests a new technique to mitigate the risk of theft: distribute the information over a large database of random looking entries in a clever way that forces any adversary to steal most of the database before he has any chance at recovering any of the stored information. If the database is on the order of a few terabytes, it is easier to monitor the system and prevent anyone from stealing enough data to do any damage. Valid users, on the other hand, can access their data quickly using their passwords.

Our model differs from the previous work on key exposure in how we deal with key compromises. Instead of assuming we are able to coordinate key replacement through one of the methods above, we assume that there are physical limits on the amount of data an adversary steals and show that any adversary cannot continue to break the scheme after his access is revoked. We also address different problems from the [DLL05] work. We are concerned with active protocols for portable devices, while their application is for a database storing terabytes of information.

The limited communication model and our user authentication protocol were independently developed in [Dzi05]. There, Dziembowski also presented the first protocol for authenticated session-

key generation, which we base our variant on.

1.3 Structure of the Paper

In section 2, we present some technical concepts and notation that are needed for the description and analysis of our protocols. In section 3, we present our protocol for user authentication and state the security theorem. We defer a detailed proof to Appendix A. In section 4, we give a protocol for authenticated session-key generation and sketch a proof of its security. In section 5, we present some applications for our protocols and calculate parameters that accurately estimate the storage requirements for secure implementations.

2 Preliminaries

This section contains some notations and definitions that will be used throughout the paper. Most of these are standard.

For a finite set S, by $a \in_R S$ we mean that a is chosen from the uniform distribution on S. For a random variable or probability distribution X, by $x \leftarrow X$, we mean that x is chosen according to the distribution of X. For an integer n, we use U_n to denote the uniform distribution on $\{0,1\}^n$. We use the standard notation $[n] = \{1, \ldots, n\}$. For a string $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ and a subset $S = \{i_1, \ldots, i_m\} \subset [n]$, $x_S = (x_{i_1}, \ldots, x_{i_m})$ is the substring of x at positions $\{i_1, \ldots, i_m\}$.

Unless otherwise specified, k will be a security parameter. We call a function $\mu : \mathbb{N} \to \mathbb{R}$ *negligible* if for every positive polynomial $p(\cdot)$, there exists an n_0 such that for all $k > n_0$, $\mu(k) < \frac{1}{p(k)}$. In other words, $\mu(k)$ is asymptotically smaller than k^{-c} for any constant $c > 0$. We say that a probability $p(k)$ is *overwhelming* if $p(k) = 1 - \mu(k)$ for some negligible function $\mu(k)$. We use $\mathrm{poly}(k)$ to denote an unspecified polynomial in k. We use PPT for the shorthand of probabilistic polynomial time.

Let X and Y be two discrete random variables or distributions on a (countable) set S. The *statistical distance* $\Delta(X,Y)$ between X and Y is defined as

$$\begin{aligned} \Delta(X,Y) &= \frac{1}{2}\sum_{\alpha \in S} |Pr[X=\alpha] - Pr[Y=\alpha]| \\ &= \max_{T \subset S} |\Pr[X \in T] - \Pr[Y \in T]|. \end{aligned}$$

Thus, the statistical distance between X and Y is the maximum advantage of any unbounded statistical test for distinguishing between X and Y. We say that X and Y are *ε-close* if $\Delta(X,Y) \leq \varepsilon$.

For two ensembles $X = \{X_k\}$ and $Y = \{Y_k\}$ of random variables or distributions, we say that X and Y are *statistically indistinguishable* if for all sufficiently large k, the statistical distance $\Delta(X_k, Y_k)$ is negligible in k. We say that X and Y are *computationally indistinguishable* if for every PPT distinguisher D, for all sufficiently large k, $|\Pr[D(X_k) = 1] - \Pr[D(Y_k) = 1]|$ is negligible in k. We say that X is *pseudorandom* if X is computationally indistinguishable from the uniform distribution ensemble.

An efficiently computable function G is said to be a *pseudorandom generator* (PRG for short) if (1) its output length exceeds its input length, and (2) its output distribution is pseudorandom if its input is chosen uniformly.

The *min-entropy* $\mathrm{H}_\infty(X)$ of a random variable X is defined as

$$\mathrm{H}_\infty(X) = \min_{x \in \mathrm{supp}(X)} \{-\log \Pr[\mathrm{X} = \mathrm{x}]\}$$

where $\mathrm{supp}(X)$ is the support of X. Let X be a random variable taking values in $\{0,1\}^n$, and let $k \leq n$. We say that X is a *k-source* if $\mathrm{H}_\infty(X) \geq k$, that is, for every $x \in \mathrm{supp}(X)$, $\Pr[X = x] \leq 2^{-k}$. Therefore, informally speaking, that X has high min-entropy means that the value X takes on is *hard to guess* (for an unbounded adversary). For a random variable X taking values in $\{0,1\}^n$ and $\alpha \in [0,1]$, we say that X has *entropy rate* α if X is an αn-source.

3 User Authentication

3.1 The Model

We first precisely describe the problem and the model. A user and a server share a long secret key X, and engage in a protocol in which the user authenticates himself to the server. Throughout the paper, $N = |X|$ denotes the length of the secret key. Typically N can be set to be a large polynomial in the security parameter k. A communication-bounded adversary, who is able to break into the user's machine, attacks the protocol in two phases, as follows.

Time is divided into sessions, and during each session one execution of the protocol is completed. In Phase I, the adversary attacks in t sessions for some t. For each session number $i \in [t]$, the adversary may either break in or not. We call a session during which adversary breaks in a compromised session. Let T_i denote the transcript of the interaction between the user and the server in the i-th session. Let C_i denote the information the adversary communicates to its host machine from the user's machines in the i-th session. Let R_i denote other secret information stored on the user's machine in the i-th session, such as the private randomness used by the user in this session. If the i-th session is uncompromised, then the adversary is simply an eavesdropper who records the transcript T_i, and C_i is simply the empty string. If the i-th session is compromised, then the adversary breaks in and computes a function $C_i = A(X, R_i, C_1, \ldots, C_{i-1}, T_1, \ldots, T_i)$, provided that $|C_i| \leq \beta N$ for some small fraction $\beta << 1$, and sends C_i to the adversary's host. From now on, we refer to βN as the communication bound, and β the communication rate, of the adversary.

In Phase II, after t sessions of attack in Phase I, the adversary exits the machine of the user, and based on the information $T_1, \ldots, T_t$, $C_1, \ldots, C_t$ stored on its host machine, attempts to impersonate the user. We say that the adversary is successful if at the end of Phase II, the server accepts the adversary as the user with a non-negligible probability.

We note that the adversary considered here is a passive one in the sense that he can only tap the communication channel and cannot block, modify or inject messages.

3.2 The Protocol and Its Security

We now describe our simple protocol. The server chooses a random subset $S \subset [N]$ of k indices, and asks the user to reply with the bits of the key X at these k positions. The server accepts if and only if the user's answers are correct.

If the server is secure and cannot be broken into, then in order for the server to authenticate many users, one may use a pseudorandom generator G and a short random seed s to generate the key $X = G(s)$, with X stored at the user and s stored at the server.

The protocol and its "asymmetric" variant above are presented in Figures 1 and 2 respectively.

Setup: The User and the Server share random key $X \in_R \{0,1\}^N$.

Protocol:

1. **Server:** Choose a random subset $S \subset [N]$ of size $|S| = k$ and send S to the User.
2. **User:** Reply with the substring X_S with positions in S.
3. **Server:** Accept if and only if all the k bits sent by the User are correct.

Figure 1: User Authentication Protocol

Setup:

- A PRG $G : \{0,1\}^k \to \{0,1\}^N$ (with $k << N$) agreed upon once and for all.
- A short random seed $s \in_R \{0,1\}^k$ is chosen.
- The User stores $X = G(s)$, and the Server stores s.

Figure 2: Setup for the Asymmetric Variant

3.3 Security of the Protocol

We show that our protocol is secure. The intuition for the security of the protocol is as follows. As long as the total number of bits the adversary steals and the number of bits of the key X the user sends in the t sessions of attack in Phase I is small compared to N, the "residual" min-entropy left in X is still sufficiently high. Say that the residual min-entropy of X is δN for some $\delta < 1$, that is, conditioned on the information the adversary has by the end of Phase I, the key X has min-entropy δN. A well known and important result from the theory of pseudorandomness says that taking a random sample of X essentially preserves the entropy rate. That is, for a random subset $S \subset [N]$ of size k, the substring X_S has min-entropy $\delta' k$ for some δ' related to δ, even conditioned on the adversary's information on X. Thus, by the definition of min-entropy, the probability that the adversary correctly outputs all the bits in X_S, given his information, is at most $2^{-\delta' k}$. Note that in this analysis, the adversary is allowed to be computationally unbounded, and the only restriction is a bound on the communication complexity.

The actual proof of the security is a little more delicate, and we leave it in Appendix A. We state the main result of the section in the theorem below.

Theorem 1. For any adversary A with communication rate β who compromises $\tilde{t}$ of the t sessions in Phase I, the probability that A succeeds in impersonating the user in Phase II is at most $2^{-\Omega(\delta k/\log(1/\delta))}$, where $\delta = 1 - \tilde{t}\beta - (t+1)k/N$.

In the asymmetric variant of the protocol, where the key is pseudorandom, under the assumption that the adversary is PPT bounded and does not steal the seed of a pseudorandom generator, the adversary's probability of success increases by at most a negligible fraction.

We now address an efficiency issue in the asymmetric variant of the protocol. Since the server

authenticates many users, for each user the server only stores a seed s but not the long output of $G(s)$. The requirement of computing the PRG $G(s)$ while authenticating each user may overburden a busy server. Note that the authentication protocol requires the server to verify only a small substring of $k << N$ bits. Thus we would like to use a PRG capable of producing those bits without computing the entire output.

One way to construct such a PRG is as follows. Let $\{F_s : \{0,1\}^m \to \{0,1\}^m\}$, $m = \mathsf{poly}(k)$, be an ensemble of *pseudorandom functions* (PRFs) [GGM86]. (See [NR04, NRR02] for efficient constructions of PRFs. Practical candidates for PRFs also include block ciphers.) It is a basic fact that a generator G defined as $G(s) = (F_s(1), \ldots, F_s(\ell))$, $\ell = \mathsf{poly}(k)$, is a PRG. Choosing the parameters suitably with $k << \ell = \mathsf{poly}(k)$ and $N = m\ell$, one obtains a PRG with a "locality" property where computing k bits of the output $G(s)$ requires at most k (as opposed to all ℓ) evaluations of the basic PRF F_s.

4 Authenticated Session Key Generation

In this section we consider authenticated session key generation in the limited communication model, building on the work of Dziembowski [Dzi05].

4.1 The Model

We consider a scenario where two parties, Alice and Bob, who share a long secret key X, wish to establish a secure session key for each communication session between them in the presence of an *active* PPT adversary. In each session, in addition to recording the interaction between Alice and Bob, compromising the session by breaking into the machines of Alice and Bob, and transmitting a limited amount of information, this adversary is also able to block, modify and inject messages in the communication channel between Alice and Bob.

We use the same notations as those in Section 3. In particular, N is the length of the key X, $\beta < 1$ is the communication rate (i.e. the adversary can transmit βN bits in each session), t is the total number of sessions in Phase I, and $\tilde{t}$ is the number of compromised sessions during which the adversary breaks in. After t sessions in Phase I, the adversary exits the machines of Alice and Bob, yet is still able to block, modify and inject messages.

We say that the adversary is successful if either of the following occur:

1. During a compromised session, the adversary learns information on the session keys from some uncompromised session in the past;
2. During an uncompromised session in the future after Phase I, the adversary
   (a) learns some information on the session key accepted by Alice or Bob; or
   (b) causes Alice and Bob to accept different session keys.

We say that an authenticated session key generation protocol is secure if the success probability of the adversary is negligible. We omit a formal description of the model and definition of security here, and will include one in the full version of the paper.

4.2 A Protocol for Authenticated Session-Key Generation

We present and analyze a protocol for authenticated session key generation that is a variant of a protocol by Dziembowski [Dzi05]. Unlike the protocol of Dziembowski, our protocol does not depend on a random oracle [FS86, BR93]. A random oracle is a truly random function that is publicly available via oracle access. The random oracle model is useful because security can usually be achieved more efficiently and simply than in the standard model without the oracle. In reality such a random oracle is impossible to implement, so a cryptographic hash function is used as a heuristic replacement. However, it has recently been shown that a protocol secure in the random oracle model *may not* be secure when the random oracle is replaced by *any* secure cryptographic hash functions [CGH04]. While it is unclear whether such a replacement of a random oracle by a hash function leads to vulnerabilities in real systems, removing the random oracle without significant loss of efficiency shall always be desired. We do exactly that, at the cost of only slightly increasing the complexity of the protocol. Two main tools underlying our construction are *non-malleable coin tossing* [DDN00] and *randomness extractors* [NZ96].

Non-Malleable Coin Tossing. Non-malleable coin tossing, introduced by Dolev et. al. [DDN00], is a primitive that allows two parties, Alice and Bob, to agree on a (public) random string in presence of an active adversary who can again block, modify and inject messages in the network. Note that an adversary can always relay messages between Alice and Bob, or completely block the communication channel and execute two independent copies of the protocol, one with Alice and the other with Bob. Non-malleability means that these are essentially the only two things the adversary can do. That is, roughly speaking, a coin tossing protocol is non-malleable if with overwhelming probability, exactly one of the following three events occurs at the end of the protocol:

1. The adversary relays all the messages between Alice and Bob, $r_A = r_B$ where r_A and r_B are the outputs of Alice and Bob respectively, and r_A (and thus r_B) is a pseudorandom string from the adversary's view;

2. (r_A, r_B) is computationally indistinguishable from a pair (u_A, u_B) of independent and truly random strings (i.e. (r_A, r_B) is pseudorandom) from the adversary's view;

3. Alice or Bob (or both) rejects and aborts the protocol. If Alice (resp. Bob) accepts while the other party aborts the protocol, her (resp. his) output r_A (resp. r_B) is pseudorandom from the adversary's view.

We omit a formal definition of a non-malleable coin tossing here. Good formal definitions based on a simulation paradigm can be found in [DDN00, Bar02].

We now remark on the efficiency of non-malleable coin tossing protocols. In the plain model without any setup assumption, the original construction of Dolev et al [DDN00] involves many rounds of interactions. Barak [Bar02] showed that in principle, a constant-round protocol can be constructed in the plain model. However the construction of [Bar02], based on sophisticated non-blackbox techniques, is not practically efficient. We observe that in the model considered here, non-malleable coin tossing protocols can be constructed efficiently by using protocols for *non-interactive non-malleable commitment* in the *common reference string (CRS) model* [DIO98, CKOS01] in a standard way. For the sake of completeness, we describe a construction in Appendix B. We note that since in our model the two parties share a secret key, there is no need to set up an extra CRS,

and part of the secret key can be used in place of a CRS. We however do note a drawback of the latter approach, namely, the standard method and the existing simulation proof paradigm yields a provably secure protocol with a slightly non-constant round complexity. The round complexity can be reduced to a small constant at the expense of allowing the adversary's advantage to be inverse polynomial for an arbitrary fixed polynomial, rather than negligibly small. We discuss this issue further in Appendix B, and leave it an open problem to construct a practically efficient constant-round protocol for non-malleable coin tossing where the adversary's advantage is provably negligible.

Randomness Extractors. A randomness extractor, introduced by Nisan and Zuckerman [NZ96], is a procedure that converts a string from weak random source (i.e. a random variable which is not perfectly random but has sufficient min-entropy) into a (shorter) almost truly random string, using a short random seed.

Definition 4.1 ([NZ96]). A function $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is an (α, ε)-extractor if for every random variable X taking values on $\{0,1\}^n$ with min-entropy rate at least α (i.e. with $\mathrm{H}_\infty(X) \geq \alpha n$), $\Delta((U_d, \mathsf{Ext}(X, U_d)), (U_d, U_m)) < \varepsilon$.

We say that an extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is *ℓ-local* if the output of $\mathsf{Ext}(x, y)$ depends only on ℓ bits of the input $x \in \{0,1\}^n$. Local extractors have been a fundamental building block in bounded storage cryptography (c.f. [Lu04, Vad04, DHRS04]). We note that Definition 4.1 requires that the output of an extractor be almost uniform *even if the seed is exposed*.[3]

In [Vad04], Vadhan constructed asymptotically optimal local extractors, that is, $O(m)$-local (α, ε)-extractors for any constant α with seed length $d = \log n + O(\log m + \log(1/\varepsilon))$. In [DM04], Dziembowski and Maurer constructed a $O(m \log(1/\varepsilon))$-local (α, ε)-extractor with a seed length of $O(\log n \cdot \log(1/\varepsilon))$, which is slightly suboptimal asymptotically. The Dziembowski and Maurer construction is however computationally extremely efficient, involving only XOR operations, and thus in practice may be the most useful.

In addition to a non-malleable coin tossing protocol and a local extractor, we also use, as in [Dzi05], a semantically secure public-key encryption (PKE) scheme [GM84] and a message authentication code (MAC) secure against adaptive chosen message attacks [GMR88]. The definitions of a semantically secure PKE and a secure MAC are standard and are omitted here.

We now present our protocol in Figure 3.

Remark: Steps 1 and 2 in our protocol replace the first 3 steps of Dziembowski's protocol, and in particular, *removes the random oracle* employed in that protocol. Steps 3-6 are the same as the last 4 steps of Dziembowski's protocol. We note that the PKE is used to guarantee that the adversary who compromises a session does not learn information about session keys from past uncompromised sessions. Without this requirement, a non-malleable coin tossing protocol, a local extractor and a MAC would be sufficient.

4.3 Security of Our Protocol

We outline a sketch for the proof of security in this subsection. Our proof is similar to that of Dziembowski's protocol at a high level, but some care must be taken where the proofs differ.

[3] Such an extractor is usually referred to as a strong extractor in the literature on randomness extractors.

Setup:

- $X \in_R \{0,1\}^N$: The secret key shared between Alice and Bob.
- NMCT: A non-malleable coin tossing protocol between Alice and Bob.
- $\mathsf{Ext} : \{0,1\}^N \times \{0,1\}^d \to \{0,1\}^m$: A local extractor.
- $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$: A semantically secure public-key cryptosystem.
- $(\mathsf{MAC}, \mathsf{Verify})$: A MAC secure against chosen message attacks.

Protocol:

1. Alice and Bob participate in NMCT to generate r_A and r_B respectively.
2. Alice and Bob compute the local extractor Ext on the key X, using r_A and r_B as seeds, and obtain $K_A = \mathrm{Ext}(X, r_A)$ and $K_B = \mathrm{Ext}(X, r_B)$ respectively.
3. Alice generates a public-key/secret-key pair $(pk, sk) \leftarrow \mathsf{Gen}(1^{\mathsf{k}})$ for Enc. She computes $tag_A \leftarrow \mathsf{MAC}_{K_A}(pk)$ and sends (pk, tag_A) to Bob.
4. Bob checks that $\mathsf{Verify}_{K_B}(pk, tag_A) = 1$. If not, Bob rejects and aborts. If $\mathsf{Verify}_{K_B}(pk, tag_A) = 1$, Bob chooses a $K' \in_R \{0,1\}^l$ uniformly, encrypts it by $C \leftarrow \mathsf{Enc}_{pk}(K')$, and tags C by $tag_B \leftarrow \mathsf{MAC}_{K_B}(C)$. Bob sends (C, tag_B) to Alice, outputs K', and accepts.
5. Alice checks that $\mathsf{Verify}_{K_A}(C, tag_B) = 1$. If not Alice rejects and aborts. If $\mathsf{Verify}_{K_A}(C, tag_B) = 1$, Alice decrypts $K' \leftarrow \mathsf{Dec}_{sk}(C)$, outputs K', and accepts.
6. The string K' is used as the session key for the current session. After using the session key K' is used, both Alice and Bob erase k' and all the private randomness used.

Figure 3: Authenticated Session Key Generation Protocol

First it is clear that in a compromised session, the adversary learns nothing about the session keys from past uncompromised sessions. This follows from the semantic security of PKE, and Step 6 of the protocol where the session key of each session is erased once it is used.

We now show that in an uncompromised session in the future, Alice and Bob cannot be fooled into accepting different session keys, and any accepted session key is pseudorandom (i.e. indistinguishable from a uniformly random string) from the view of the adversary. If Alice or Bob aborts the NMCT protocol in Step 1, then it is easy to show the security of protocol. We thus assume that neither party aborts during NMCT in Step 1.

The basic idea is as follows. After non-malleable coin tossing in Step 1 of the protocol, if both Alice and Bob accept, then either (1) $r_A = r_B$, and r_A (thus r_B) is *pseudorandom*, or (2) (r_A, r_B) is *indistinguishable from an independent and uniform pair* (u_A, u_B). In Case (1), clearly in Step 2 of the protocol $K_A = K_B$. As in the analysis in Section 3.3, after t sessions with $\tilde{t}$ compromised sessions in Phase I, the residual min-entropy of X is at least δN for some δ.[4] Thus by using a

[4]Here $\delta = 1 - \tilde{t}\beta - (t+1)k/N$ as before.

local extractor with suitably chosen parameters, from the definition of an extractor and that of pseudorandomness, it follows that $K_A = \mathsf{Ext}(X, r_A) = \mathsf{Ext}(X, r_B) = K_B$ is *pseudorandom*.[5] It is not difficult to see that in Case (1), given that $K_A = K_B$ and is pseudorandom, if the adversary relays the messages exchanged between Alice and Bob during the rest of the protocol, then by the security of PKE, it follows that Alice and Bob will accept the same session key K' which is pseudorandom and thus yields essentially no information to the adversary; if the adversary modifies a message or injects a message, then by the security of MAC, either Alice or Bob will detect and reject.

Now consider Case (2), where (r_A, r_B) is indistinguishable from an independent and uniform pair (u_A, u_B). Then similar to the argument used in Case (1), from the definition of an extractor and that of computational indistinguishability, (K_A, K_B) is indistinguishable from a uniform pair (U_A, U_B).[6] Given this, it is not difficult to see, from the security of MAC, that in Step 4 of the protocol, Bob will reject and abort the protocol. Further, since K_A is pseudorandom from the adversary's view, it follows again from the security of MAC that in Step 5 of the protocol, Alice will reject no matter what message the adversary sends at the end of the previous step. Thus in Case (2) both Alice and Bob will reject.

There is however one more subtle issue we must address, namely, the non-malleable coin tossing protocol NMCT is only guaranteed to be secure *when it is executed alone*. But here it is used as a subprotocol in our protocol, and subtlety may arise as the adversary can block in the middle, execute two concurrent copies of the protocol, one with Alice and one with Bob, and schedule them arbitrarily. We must argue that NMCT cannot broken by an adversary who uses information from later in the protocol to his advantage in the NMCT stage.

Fortunately, it is not hard to show that NMCT remains secure when used as a subprotocol in our protocol. First, suppose that the adversary concludes NMCT with Bob first, and intends to use information from later to break NMCT against Alice. In this case, the adversary must send a message of the form (pk, tag_A) to Bob, who will either reject or accept and respond with (C, tag_B). We claim that Bob rejects with overwhelming probability in this case. Since the coins r_B output by Bob at the end of NMCT are guaranteed to pseudorandom, using the argument as above, we have that the string K_B is pseudorandom in the view of the adversary. Thus generating a message (pk, tag_A) on which Bob accepts is equivalent to forging a tag in MAC, and this happens with only negligible probability by the security of MAC.

Next, suppose the adversary concludes the NMCT protocol with Alice first and attempts to break NMCT against Bob. We will show that if the adversary succeeds in breaking NMCT in this case, then he could have broken NMCT when it was executing alone. After concluding NMCT with Alice first, the adversary receives a message of the form (pk, tag_A) from Alice. Using the property of Ext, K_A is indistinguishable from uniform from the view of the adversary and Bob, thus the message (pk, tag_A), in the view of the adversary and Bob, is just a random public key and a MAC tag of pk under a key that is indistinguishable from uniform. This means that the adversary could generate the message pair himself and break NMCT without using any other help. This contradicts the security of NMCT.

[5] If a $r_A = r_B$ were a truly random seed, then the output of the extractor is statistically close to uniform. Since $r_A = r_B$ is pseudorandom, a standard reducibility argument shows that the output $K_A = K_B$ is pseudorandom. We note that here the strong property of an extractor requiring that the output remains close to uniform *even if the seed is exposed*, is used in an essential way.

[6] Here again the strong property of an extractor requiring that the output remains close to uniform *even if the seed is exposed*, is used in an essential way.

This concludes a proof sketch. A formal proof of security is based on a rather complicated simulation argument. We leave a precise statement of the theorem and a full proof to the full version of the paper.

5 Efficient Realizations in Applications

In this section we describe some example applications for which our protocols are useful. We calculate the constants in our asymptotic statements above to determine some example settings for parameters.

5.1 Physical Authentication Tokens

Our first application is for what we term *physical authentication tokens*. These are small devices that are used to authenticate a user in any situation where the subsequent sessions cannot be hijacked. These tokens can be used to unlock secured doors or perform logins when a user is physically at the machine, but they are not suitable for remote authentication because a malicious adversary can always take over a session after the initial authentication. The scenario we are concerned with is where an adversary nearby can eavesdrop, perform analysis, and attempt to learn the stored key.

Our protocol for user authentication can be implemented on physical authentication tokens by storing the long string X on the token and storing the short seed s at the server. We can realize the bounded communication requirement of the model by designing the device to respond to queries for bits of X only once every few seconds. This will be quick enough for a valid user to authenticate without waiting, but an adversary attempting to learn much of X will be severely restricted in how much of X he can access.

A token implementing our protocol has two desirable properties that are not found in current devices providing user authentication. First, the token is just a simple storage device and does not perform any calculations beyond fetching bits. Most known side channel attacks do not apply here because these timing and powering attacks usually exploit the irregular nature of advanced computations in cryptographic protocols. The only meaningful attack against our devices would be one that stole more bits of X.

Second, aside from the security of the pseudorandom generator, our security is information theoretic, meaning that we make no complexity assumptions to reach our security guarantee and that the security of an implementation of our protocol will not degrade as faster attacking algorithms are designed.

We can use a detailed analysis of the calculations in Appendix A to determine the constants hidden in the asymptotic notation and compute the actual parameters for our tokens. The relevant constants can be computing using the lemmata of [Vad04] and [DHRS04].

One possible setting of gives the following:

1. $|X| = 384 \cdot 2^{23}$, meaning the tokens store 384MB of pseudorandom data.
2. $k = 512$, so each query is for 512 bits.
3. The device answers one query every 10 seconds.
4. $\beta\tilde{t} = 1/4$, which means the adversary gets to query the device once every 10 seconds for 6 months.

5. $t = 10,000$, so the device can be used for that many authentications.

6. The chance that an adversary successfully impersonates a user is at most 2^{-20}.

Of course, the adversary's attack duration can be changed by increasing the storage of the device or increasing the amount of time it will wait between queries.

5.2 Secure Sessions on Laptops

We also propose an application for our authenticated session-key generation protocol. Here we can realize the stated model by equipping a client computer (e.g., a desktop or a laptop) with a slow hard drive to store the random data X and meter it out at a rate sufficient for valid users, but nearly useless for malicious adversaries. Then the client can participate in our protocol using the drive to access the pseudorandom bits of X.

By using this configuration we gain some resistance to virus infection. We are concerned with the scenario where a virus infects a computer and steals and transmits the stored secret key to a remote computer controlled by an adversary. Then the adversary tries to impersonate the user of the infected computer. Using our protocol and rate provided by the drive, we determine how long a virus would need to control the machine before it could gain any significant information about the generated session keys. This ensures us that even if a virus goes undetected, there is no lasting effect as long as it is not present for more than the allowed time, which will be on the order of years. Typically, anti virus companies are able to produce a signature of a new virus within a few days of the first few instances of virus surfacing in the wild. We assume that the user updates the virus signature database and runs an antivirus program on her computer occasionally, e.g., at least once a year.

We can accurately compute the amount of time an adversary will need to successfully attack the system by setting the parameters for each of the underlying primitives to their current industry standards, and then analyzing the error from the extractor.

The analysis of the extractor given in [DM04] can easily be used to calculate our parameters for authenticated session key generation. We must use the more general theorem given there because we are not using the extractor for key expansion, and our extractor output will usually be much shorter than the extractor seed computed by NMCT. Still, the calculations are not complicated and we get the following example parameters:

1. $|X| = 2^{40}$, so the drive must store 128 GB of data.

2. Set the MAC key length to 256 bits.

3. The drive responds with 16K of information once every 10 seconds.

4. The virus gets to query the drive for bits for 700 days.

5. Then the probability that an adversary can distinguish the MAC key from a truly uniform key is less than 2^{-30}.

These parameters provide extremely high security. If the size of the drive is a restricting factor, it is possible to work with a smaller drive, at a reasonable cost in either the security parameter or the amount of time the adversary gets to attack the system.

6 Conclusions and Future Work

We have described a new technique to harden portable devices against compromises that expose secret key information, and defined a formal model for analyzing protocols implementing our technique. We are also able to show that this method is practical and presented some of its realistic applications.

Our user authenticated protocol is very efficient and can be readily implemented. The efficiency of our authenticated session key generation protocol relies on the efficiency of non-malleable coin tossing, and we leave it an open problem to construct a more efficient protocol for non-malleable coin tossing than the one described here.

The next step in cultivating our model is finding protocols for other cryptographic tasks. Authenticated session-key generation, as we have stated it, is an interesting "catch all" for symmetric-key cryptography, as many protocols can be derived using it. Practically speaking, however, these derived protocols are only a proof of concept because they require interaction between the user and server. It would be interesting to find non-interactive protocols for other tasks, and extend the model to asymmetric-key cryptography. It would also be interesting if these protocols could use very small storage devices, on the order of a few megabytes.

References

[ADR02] Yonatan Aumann, Yan Zong Ding, and Michael O. Rabin. Everlasting security in the bounded storage model. *IEEE Transactions on Information Theory*, 48(6):1668–1680, June 2002.

[And97] Ross Anderson. Two remarks on public key cryptology. Invited Lecture. In *4th ACM Conference on Computer and Communications Security*, 1997.

[AR99] Yonatan Aumann and Michael O. Rabin. Information theoretically secure communication in the limited storage space model. In *Advances in Cryptology - CRYPTO '99*, pages 65–79. Springer-Verlag, 1999.

[AR00] Michel Abdalla and Leonid Reyzin. A new forward-secure digital signature scheme. In Tatsuaki Okamoto, editor, *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 116–129. Springer, 2000.

[Bar02] Boaz Barak. Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In *FOCS*, pages 345–355. IEEE Computer Society, 2002.

[BM99] Mihir Bellare and Sara K. Miner. A forward-secure digital signature scheme. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 431–448. Springer, 1999.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BY03] Mihir Bellare and Bennet S. Yee. Forward-security in private-key cryptography. In Marc Joye, editor, *CT-RSA*, volume 2612 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2003.

[CCM98] Christian Cachin, Claude Crépeau, and Julien Marcil. Oblivious transfer with a memory-bounded receiver. In *39th Annual IEEE Symposium on Foundations of Computer Science*, pages 493–502, November 1998.

[CDH+00] Ran Canetti, Yevgeniy Dodis, Shai Halevi, Eyal Kushilevitz, and Amit Sahai. Exposure-resilient functions and all-or-nothing transforms. In *EUROCRYPT*, pages 453–469, 2000.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. On the random-oracle methodology as applied to length-restricted signature schemes. In Moni Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 40–57. Springer, 2004.

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 255–271. Springer, 2003.

[CKOS01] Giovanni Di Crescenzo, Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Efficient and non-interactive non-malleable commitment. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 40–59. Springer, 2001.

[CM97] Christian Cachin and Ueli Maurer. Unconditional security against memory bounded adversaries. In *Advances in Cryptology - CRYPTO '97*, pages 292–306. Springer-Verlag, 1997.

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.

[DHRS04] Yan Zong Ding, Danny Harnik, Alon Rosen, and Ronen Shaltiel. Constant-round oblivious transfer in the bounded storage model. In *1st Theory of Cryptography Conference – TCC '04*, pages 446–472, 2004.

[Din01] Yan Zong Ding. Oblivious transfer in the bounded storage model. In *Advances in Cryptology – CRYPTO '01*, pages 155–170. Springer-Verlag, August 2001.

[Din05] Yan Zong Ding. Error correction in the bounded storage model. In Joe Kilian, editor, *TCC*, volume 3378 of *Lecture Notes in Computer Science*, pages 578–599. Springer, 2005.

[DIO98] Giovanni Di Crescenzo, Yuval Ishai, and Rafail Ostrovsky. Non-interactive and non-malleable commitment. In *STOC*, pages 141–150, 1998.

[DKXY02] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Key-insulated public key cryptosystems. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 65–82. Springer, 2002.

[DLL05] David Dagon, Wenke Lee, and Richard J. Lipton. Protecting secret data from insider attacks. In Andrew S. Patrick and Moti Yung, editors, *Financial Cryptography*, volume 3570 of *Lecture Notes in Computer Science*, pages 16–30. Springer, 2005.

[DM04] Stefan Dziembowski and Ueli Maurer. Optimal randomizer efficiency in the bounded-storage model. *Journal of Cryptology*, 17(1):5–26, 2004.

[DR02] Yan Zong Ding and Michael O. Rabin. Hyper-encryption and everlasting security (extended abstract). In *19th Annual Symposium on Theoretical Aspects of Computer Science*, pages 1–26. Springer-Verlag, March 2002.

[Dzi05] Stefan Dziembowski. Intrusion-resilience via the bounded-storage model. Cryptology ePrint Archive, Report 2005/179, 2005.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):656–715, 1984.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, June 1988.

[IR01] Gene Itkis and Leonid Reyzin. Forward-secure signatures with optimal signing and verifying. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 332–354. Springer, 2001.

[KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *Advances in Cryptology - CRYPTO '04, Lecture Notes in Computer Science*, pages 335–354. Springer-Verlag, 2004.

[Lu04] Chi-Jen Lu. Encryption against space-bounded adversaries from on-line strong extractors. *Journal of Cryptology*, 17(1):27–42, 2004.

[Mau92] Ueli Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 5(1):53–66, 1992.

[MST04] Tal Moran, Ronen Shaltiel, and Amnon Ta-Shma. Non-interactive timestamping in the bounded storage model. In *Advances in Cryptology - CRYPTO '04*, pages 460–476. Springer-Verlag, 2004.

[NR04] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2):231–262, 2004.

[NRR02] Moni Naor, Omer Reingold, and Alon Rosen. Pseudorandom functions and factoring. *SIAM J. Comput.*, 31(5):1383–1404, 2002.

[NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996.

[Rab05] Michael O. Rabin. Provably unbreakable hyper-encryption in the limited access model. In *IEEE Information Theory Workshop on Theory and Practice in Information-Theoretic Security*, 2005.

[Vad04] Salil P. Vadhan. Constructing locally computable extractors and cryptosystems in the bounded storage model. *Journal of Cryptology*, 17(1):43–77, 2004.

Appendix A: Proof of Theorem 1

In this appendix we prove Theorem 1. Let t be the total number of sessions, and $\tilde{t}$ the number of compromised sessions in Phase I of the attack. In each of the $\tilde{t}$ compromised sessions the adversary sends βN bits to its host machine. In each of the t sessions, the user sends k original bits of the key X to its host. Define $\kappa = k/N$. Thus, at the end of Phase I, the adversary has at most

$\tilde{t}\beta N + tk = (\tilde{t}\beta + t\kappa)N$ bits of information of X.[7] We now recall the following basic yet important lemma.

Lemma 1 (c.f. [NZ96]). Let Y and Z be any two (correlated) random variables. Suppose that Y is a m-source, and Z takes values in $\{0,1\}^\ell$. Then for every $\nu > 0$, with probability at least $1-\nu$ over $z \leftarrow Z$, $Y|_{Z=z}$ is a $(m - \ell - \log(1/\nu))$-source.

Informally the lemma says that if Y has m bits of min-entropy, then for any ℓ-bit long Z that is arbitrarily correlated to Y, Y has roughly $m - \ell$ bits of residual min-entropy even if Z is given.

Let random variable W denote the $(\tilde{t}\beta + t\kappa)N$ bits of information on X the adversary has at the end of Phase I. Applying Lemma 1 to X and W, letting $\nu = 2^{-k}$, we have that with probability at least $1 - 2^{-k}$ over $w \leftarrow W$, $X|_{W=w}$ is a $(1 - \tilde{t}\beta - (t+1)\kappa)N$-source. Suppose that such a good w occurs and fix such a good w.

We now use a fundamental lemma proved in [NZ96] and refined in [Vad04], which states that taking a random sample of bits from a random variable essentially preserves its entropy rate. The lemma presented below, which suffices in our case, is a special case of the lemma proven in [NZ96, Vad04] where the samples are uniform (c.f. [DHRS04]).

Lemma 2. For every $N \geq k$ and $\delta > 0$, for every δN-source Y over $\{0,1\}^N$, for at least a $1 - \varepsilon$ fraction of subsets $S \subset [N]$ with $|S| = k$, Y_S is ε-close to some $\delta k/2$-source, where $\varepsilon = 2^{-\Omega(\delta k / \log(1/\delta))}$.

Applying Lemma 2 to $\delta = 1 - \tilde{t}\beta - (t+1)\kappa$ and $Y = X|_{W=w}$, we have that for at least a $1-\varepsilon$ fraction of subsets $S \subset [N]$ with $|S| = k$, where ε is defined in Lemma 2, there is a $\delta k/2$-source which is ε-close to $X_S|_{W=w}$. Suppose that such a good subset S of k indices is chosen, and let Z be a $\delta k/2$-source which is ε-close to $X_S|_{W=w}$. By the definition of min-entropy the probability that an adversary can guess the correct value of Z is at most $2^{-\delta k/2}$. It follows from the definition of statistical distance that the probability that the adversary outputs the correct value of the substring X_S, given W and S, is at most $\varepsilon + 2^{-\delta k/2}$.

Thus, accumulating all the errors in the analysis, we conclude that the probability that the adversary succeeds in impersonating the user, that is, the probability that the adversary outputs the correct value of X_S at the end of Phase II, is at most $\nu + \varepsilon + \varepsilon + 2^{-\delta k/2} = 2^{-k} + 2^{-\delta k/2} + 2^{-\Omega(\delta k / \log(1/\delta))}$.

Appendix B: Non-Malleable Coin Tossing in Our Model

We first briefly describe a non-malleable coin tossing protocol in our model based on non-interactive non-malleable commitment in the common reference string (CRS) model [DIO98, CKOS01]. The construction is based on standard techniques. Again we note that in our model, there is no need to set up an extra CRS. Part of the secret key can be used in place of a CRS, and results in the CRS model apply here.

We first briefly and informally describe non-malleable commitment that was also introduced in [DDN00]. Non-malleable commitment is a primitive whereby a sender commits a value v to a receiver in such a way that

[7] It is possible to give a better analysis on the total number of original bits of X the user sends to the server in the t sessions, as the k-element subsets may overlap. However, since this part of the communication is supposedly dominated by $\tilde{t}\beta N$, the amount of information the adversary steals, we omit a refined analysis here.

1. The commitment hides all partial information of v until decommitment.
2. The sender cannot cause the receiver to open the commitment as a different value $v' \neq v$ during decomitment.
3. An active adversary, who is a man-in-the-middle, can not cause the receiver to open the commitment as a different value $\tilde{v}$ that is related to v in a meaningful way. More precisely,
   (a) Either they adversary relays all messages between the sender and receiver so that the receiver receives the original value v during decomitment;
   (b) Or for every polynomial-time computable relation $\mathcal{R}$, $\mathcal{R}(v, \tilde{v}) = 0$, where v is the original value the sender commits to and $\tilde{v}$ is value the receiver outputs during decomitment.

In [DIO98, CKOS01] it was shown that in the CRS model non-malleable commitment can be constructed efficiently and *non-interactively*, requiring that only a single message be sent from the sender to the receiver in both the commitment and decommitment phases.

Given such a non-interactive non-malleable commitment protocol, a 4-message non-malleable coin tossing protocol can be constructed as described in Figure 4.

Setup:

- $X \in_R \{0,1\}^N$: The secret key shared between a Alice and a Bob.
- $\mathsf{NMC} = (\mathsf{Commit}, \mathsf{Decommit})$: A non-interactive non-malleable commitment protocol in the CRS model, using part of X as the CRS.

Protocol:

1. Alice chooses random $x \in_R \{0,1\}^m$, computes the commitment $c_x = \mathsf{Commit}(x)$, and sends c_x to Bob.
2. Bob chooses random $y \in_R \{0,1\}^m$, computes the commitment $c_y = \mathsf{Commit}(y)$, and sends c_y to Alice.
3. Alice sends decommitment information d_x to Bob, who opens $x' = \mathsf{Decommit}(c_x, d_x)$. If $x' = \perp$ (that is, if $\mathsf{Decommit}$ rejects), then Bob rejects and aborts. Otherwise, Bob accepts and outputs $x' \oplus y$.
4. Bob sends decommitment information d_x to Alice, who opens $y' = \mathsf{Decommit}(c_y, d_y)$. If $y' = \perp$, then Alice rejects and aborts. Otherwise, Alice accepts and outputs $x \oplus y'$.

Figure 4: Non-Malleable Coin Tossing Protocol

However, based on the existing simulation proof paradigm, the security of such a protocol can only be proven when $m = O(\log k)$, where m is the number of coins tossed. The case that m is super-logarithmic can be handled by sequentially repeating the protocol but at the expense of yielding a non-constant number of rounds. In fact, it was proven in [KO04] that even for proving the security in the blackbox model of standard (malleable) tossing of super-logarithmically many coins, five rounds are necessary. Thus the above protocol cannot be proven secure for $m = \omega(\log k)$ within the standard proof paradigm.

In our case, m is the seed length of the local extractor, and it is sufficient yet also necessary to have $m = O(\log N + \log(1/\varepsilon)) = O(\log k + \log(1/\varepsilon))$, where ε is the maximum advantage of the adversary allowed. In the standard setting where the advantage ε is required to be negligibly small, $m = \omega(\log k)$, thus a non-constant number of rounds would be required. In principle, at least asymptotically, a slightly super-logarithmic m is sufficient in order for ε to be non-negligible. Thus this approach yields an almost constant-round protocol. We note that if we relax the requirement and allow ε to be inverse polynomial in k for an arbitrary fixed polynomial, then $m = O(\log k)$ sufficies and the above non-malleable coin tossing protocol is provably secure. In certain applications such a relaxation might not be serious when concrete parameters are plugged in. However, it is still more desirable to construct an efficient protocol with a small constant number of rounds where the advantage of the adversary is provably negligible. We leave this as an open problem.