Intrusion-Resilience via the Bounded-Storage Model

Stefan Dziembowski[⋆]

Institute of Informatics,
Warsaw University

and

Institute of Mathematics
Polish Academy of Sciences

std@mimuw.edu.pl

June 14, 2005

Abstract. We introduce a new method of achieving intrusion-resilience in the cryptographic protocols. More precisely we show how to preserve security of such protocols, even if a malicious program (e.g. a virus) was installed on a computer of an honest user (and it was later removed). The security of our protocols relies on the assumption that the amount of data that the adversary can transfer from the infected machine is limited (however, we allow the adversary to perform any efficient computation on user's private data, before deciding on what to transfer). We focus on two cryptographic tasks, namely: authenticated key exchange and entity authentication. Our method is based on the results from the Bounded-Storage Model.

1 Introduction

In the contemporary Internet environment the computers are often exposed to the attacks of malicious programs, which can monitor the machines and steal the secret data. This type of software can be secretly attached to seemingly harmless programs, or can be installed by worms or viruses. In order to protect against these threats a computer user is usually advised to use virus and spyware removal tools. This tools need to be frequently updated (as the new viruses spread out very quickly). Nevertheless for an average PC user it is quite inevitable that his computer is from time to time infected by a malicious process (which is later removed by an appropriate tool).

This phenomenon can be particularly damaging if the user runs some cryptographic programs on his machine. This is because in most of cryptographic tasks (encryption, authentication) the user needs to posses (and store somewhere) a secret key s. If the user does not store s outside of the machine (e.g. on a trusted hardware that will later participate in the protocol), then it seems that there is little that can be done to preserve the security, as the malicious process can always steal s (and then impersonate the honest user, or decrypt his private communication). If the protocol is based on the password memorized by the user then the virus can wait until the password is typed and record the key-strokes.

[⋆] Partially supported by the EU *ECRYPT* grant IST-2002-507932.

In this paper we propose a method for constructing *intrusion-resilient cryptographic protocols*, i.e. such protocols that remain secure even after the adversary gained access to the victim's machine (and later lost this access). The security of our protocols is based on a novel assumption that the amount of data that the adversary is allowed to transfer from the victim's machine is limited (however, we allow the adversary to perform any efficient computation on user's private data, before deciding on what to transfer). In the security proofs we make use of the theory of the Bounded Storage Model (see Section 3).

1.1 Previous work

Intrusion-resilience was introduced in [15] (see also [11]) and can be viewed as combination of forward and backward security.[1] A cryptosystem is *forward-secure* if the exposure of a secret key at some particular time t does not affect the security of the sessions of the protocol that ended before t. It was studied in context of key-exchange (see e.g. [16]), digital signatures (this research was initiated by Ross Anderson, see [1]) and public-key encryption [6]. A cryptosystem is *backward-secure* if the exposure of a secret key at time t does not affect the security of the sessions of the protocol that started after t. So far it was achieved by distributing the secret key among a group of participants (e.g. in [15] this group consist of two players: a *signer* and a *home base*). One has to make an assumption that the entire group is never compromised by the adversary at the same time.

Cryptosystems that remain secure even in case of a partial leakage of the secret key were already studied in the area of *Exposure-Resilient Cryptography* (see e.g. [9]). The differences with our model are as follows: (1) they consider only the leakage of *individual* bits of the secret keys and (2) the keys in their protocols are short.

1.2 Our contribution

We propose a new method for achieving intrusion-resilience (the main novelty of our approach is the new method of achieving backward-security; the forward-security is achieved in a fairly standard way). The assumption that we make is that the secret key is too large to be transfered by the attacker from the victim's machine (e.g. K is of a size 5 GB). More precisely, we will assume that the adversary (after taking the controll over the machine) is able to perform arbitrary (efficient) computation on it's data, but the amount of data that she can retrieve is much smaller than K (e.g. it is 0.5 GB). This assumption may be quite practical as usually transmitting unnoticably 0.5 GB of data is hard. Our method is based on the theory of the Bounded-Storage Model. In this paper we focus on the symmetric authenticated key exchange and entity authentication. It remains an open problem to provide protocols for other tasks (as asymmetric encryption and signature schemes) in this model.

[1] There seems to be some confusion in the literature about the terminology. What is called *forward security* in [1] is called *backward security* in [15, 10]. In this paper we use the terminology of [15].

Our exposition is rather informal, as we mostly aim at demonstrating the power of the model, not at providing ready to use practical solutions for concrete problems. Nevertheless we believe that the protocols provided here (or their variants) may find practical applications.

2 Preliminaries

2.1 The measure of non-uniformity

We will use the following measure of non-uniformity. If p is a probability distribution over some set $\mathcal{U}$ then its distance from uniform distribution is given by the following formula:

$$d(p) := \tfrac{1}{2} \sum_{u \in \mathcal{U}} \left| p(u) - \frac{1}{|\mathcal{U}|} \right| .$$

We will extend this notation to random variables in a natural way.

2.2 Message Authentication Codes

We will use a following (simplified) definition of security of the Message Authentication Codes (*MAC*s). For more complete definition the reader may consult e.g. [13]. *MAC* is an algorithm takes as an input a security parameter 1^k, a random secret key $S \in \{0,1\}^{\nu(k)}$ (where ν is some polynomial) and a message $M \in \{0,1\}^*$. It outputs an *authentication tag* $\mathit{MAC}_S(M, 1^k)$) (we will sometimes drop 1^k). It is *secure against adaptive chosen-message attack* if any polynomial probabilistic time (*PPT*) adversary (taking as input 1^k) has negligible[2] (in k) chances of producing a valid pair $(M, \mathit{MAC}_S(M, 1^k))$, after seeing an arbitrary number of pairs

$$(M_1, \mathit{MAC}_S(M_1, 1^k)), (M_2, \mathit{MAC}_S(M_2, 1^k)) \ldots$$

(where $M \notin \{M_1, M_2, \ldots\}$) even when $M_1, M_2, \ldots$ were adaptively chosen by the adversary.

2.3 Public-Key Encryption

A *public-key encryption scheme* is a triple $(G, \mathit{encr}, \mathit{decr})$, where G is a *key-generation algorithm* taking as input 1^k and returning as output a (private-key,public-key) pair (E, D), encr is an algorithm taking as input 1^k, a message $M \in \{0,1\}^*$ and a public key E and returning a *ciphertext* $C = \mathit{encr}_E(M)$, and decr is an algorithm taking as input a private key D a ciphertext C and returning a message $M' = \mathit{decr}_D(C)$. We require that always $M = \mathit{decr}_D(\mathit{encr}_E(M))$. Let $\mathcal{E}$ be a polynomial time adversary which is given 1^k and E. Her goal is to win the following game. She produces two messages M_0 and M_1 (of the same length). Then, she is given a ciphertext $C = \mathit{encr}_S(M_r)$, where $r \in \{0,1\}$ is random. She has to guess r. We say that $(G, \mathit{encr}, \mathit{decr})$ is *semantically secure* [14] if any polynomial time adversary has chances at most negligibly (in k) better that 0.5. More on the definitions of secure public-key encryption can be found e.g. in [13].

[2] A $f : \mathcal{N} \to \mathcal{R}$ function is *negligible in* k if for every $c \geq 1$ there exists k_0 such that for every $k \geq k_0$ we have $|f(k)| \leq k^{-c}$.

2.4 Random Oracle Model

We prove the security of our protocol in the *Random Oracle Model* [3]. More precisely, we will model a hash function $H : \{0,1\}^i \rightarrow \{0,1\}^j$ as a *random oracle*, i.e. a black box containing a random function $h : \{0,1\}^i \rightarrow \{0,1\}^j$. We assume that every party (including the adversary) has access to this oracle, i.e. can ask it for the value of h on any (chosen by her) arguments.

3 Bounded Storage Model

We will use the results from the Bounded-Storage Model, introduced by Maurer in [18]. So far, this model was studied in the context of *information-theoretically secure* encryption [2, 12, 17, 21], key-agreement [5], oblivious transfer [4, 8] and time-stamping [19]. In this model one assumes that a random t-bit string R (called a *randomizer*) is either temporarily available to the public (e.g. the signal of a deep space radio source) or broadcast by one of the legitimate parties. We assume that the memory s of the adversary is smaller than t and therefore she can store only partial information about R. It was shown [2, 12, 17, 21] that under this assumption the legitimate parties Alice and Bob, sharing a short secret key Y initially, can generate a very long n-bit one-time pad X with $n \gg |Y|$ about which the adversary has essentially no information.

More formally, Alice and Bob share a short secret *initial key* Y, selected uniformly at random from a key space $\mathcal{Y}$, and they wish to generate a much longer n-bit *expanded key* X (i.e. $n \gg \log_2 |\mathcal{Y}|$).

In a first phase, a t-bit random string R is available to all parties, i.e., the randomizer space is $\mathcal{R} = \{0,1\}^t$. Alice and Bob apply a known *key-expansion function*

$$f : \mathcal{R} \times \mathcal{Y} \rightarrow \{0,1\}^n$$

to compute the expanded key as $X = f(R, Y)$. Of course, the function f must be efficiently computable and based on only a very small portion of the bits of R such that Alice and Bob need not read the entire string R.

The adversary Eve $\mathcal{E}$ can store arbitrary s bits of information about R, i.e., she can apply an arbitrary storage function

$$h : \mathcal{R} \rightarrow \mathcal{U}$$

for some $\mathcal{U}$ with the only restriction that $|\mathcal{U}| \leq 2^s$. The memory size during the evaluation of h does not need to be bounded. The value stored by Eve is $U = h(R)$. After storing U, Eve looses the ability to access R. All she knows about R is U. In order to prove as strong a result as possible, one assumes that Eve can now even learn Y, although in a practical system one would of course keep Y secret.

A key-expansion function f is secure in the bounded-storage model if, with overwhelming probability, Eve, knowing U and Y, has essentially no information about X. To be more precise, let us introduce a security parameter k which is an additional input of f and of Eve. Let us assume that the length of the randomizer and the size of Eve's memory are functions of k, i.e. $t = \tau(k)$ and $s = \sigma(k)$. Also, let the length of the output of f be a function λ of k and assume that the set of the initial keys is always equal to $\{0,1\}^{\mu(k)}$, for some function μ. We say that function f is $(\sigma, \tau, \lambda, \mu)$*-secure in the bounded-storage model* if for any Eve (with memory at most $\sigma(k)$) the statistical distance of the conditional probability distribution $P_{X|U=u,Y=y}$ from uniform distribution over the $\lambda(k)$-bit strings is negligible, with overwhelming probability over values u and y. Above we assumed that the adversary and the function f are deterministic, but note that we would not loose any security by allowing them to be randomized (formally we could do it by allowing $\mathcal{E}$ and f to take extra random inputs $r_{\mathcal{E}}$ and r_f, resp.).[3]

Several key expansion functions were proven secure in the past couple of years [2, 12, 17, 21]. In the next section we present an example of such function, taken from [12].

3.1 The scheme of [12].

The randomizer $R \in \mathcal{R} = \{0,1\}^t$ is interpreted as being arranged in a matrix with m rows, denoted $R(1), \dots, R(m)$, for some $m \geq 1$ called the *height* of the randomizer. Each row consists of $l + n - 1$ bits, for some $l \geq 1$ called the *width* of the randomizer. Hence $t = m(l+n-1)$ and R can be viewed as an $m \times (l+n-1)$ matrix (see Figure 1). The initial key $Y = (Y_1, \dots, Y_m) \in \mathcal{Y} = \{1, \dots, l\}^m$ selects one starting point within each row, and the expanded key $X = (X_1, \dots, X_n)$ is the component-wise XOR of the m blocks of length n beginning at these starting points Y_i, i.e.,

$$X = f(R, Y),$$

where $f : \mathcal{R} \times \mathcal{Y} \to \{0,1\}^n$ is defined as follows. For $r \in \mathcal{R}$ and $k = (k_1, \dots, k_m) \in \mathcal{Y}$,

$$f(r,k) := \left(\bigoplus_{i=1}^{m} r(i, k_i), \dots, \bigoplus_{i=1}^{m} r(i, k_i + n - 1) \right), \tag{1}$$

where $r(i,j)$ denotes the jth bit in the ith row of r. This is illustrated in Figure 1.

The above function f was proven asymptotically secure in [12], assuming that memory of the adversary has a size that is a constant fraction $c < 1$ of the randomizer. For the practically looking parameters (see [12] for details) this constant should be around 8%, i.e. $\sigma(k) := \tau(k) \cdot 0.08$.

4 Intrusion-Resilient Authenticated Key Exchange

By *authenticated key exchange* we mean a protocol that allows two parties (that share a long-term symmetric key) to agree securely on a session key even in presence of a malicious adversary that can obstruct their communication. Below, we describe what we mean by *intrusion-resilient* authenticated key exchange.

[3] This is because of the following: (1) the input r_f is obsolete since if $\mathcal{E}$ is randomized then having r_f clearly does not change anything as $\mathcal{E}$ can simply choose r_f himself and encode it into the description of f, (2) the input $r_{\mathcal{E}}$ is obsolete since a computationally unbounded $\mathcal{E}$ can always (for any value of k) find the „optimal" $r_{\mathcal{E}}$.

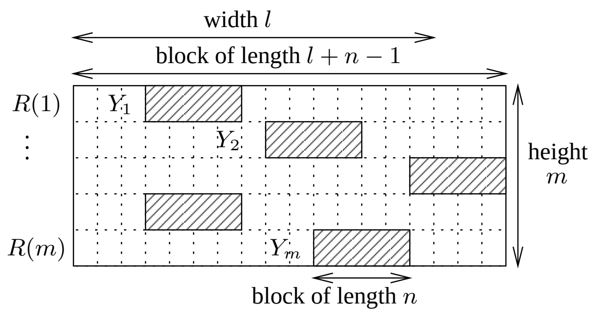

Fig. 1. Illustration of the scheme for deriving an expanded n-bit key $X = (X_1, \dots, X_n)$, to be used as a one-time pad, from a short secret initial key $Y = (Y_1, \dots, Y_m)$. The randomizer R is interpreted as a $m \times (l + n - 1)$ matrix with rows $R(1), \dots, R(m)$ of length $l + n - 1$. The expanded key X is the component-wise XOR of m blocks of length n, one selected from each row, where Y_i is the starting point of the ith block within the ith row $R(i)$.

4.1 An informal description of the model

First, let us fix the basic terminology. The honest users Alice A and Bob B will be attacked by a (computationally bounded) *adversary* Eve $\mathcal{E}$. The adversary is allowed (1) to eavesdrop and to store the entire communication between Alice and Bob (2) to fabricate messages or to prevent them from arriving and (3) to (periodically) install malicious programs on the honest user's machines (see below). Such a program will be called a *virus*. We assume that the honest users share a long-term secret key K generated randomly. The time is divided into sessions $T_1, T_2, \dots$ (the number of sessions will be bounded). At the beginning of the session the users are allowed to get some fresh random input (otherwise clearly we could not hope for intrusion-resilience, as the virus could simply predict the future behavior of the parties). At the end of each session T_i the users output a new *session key* κ_i generated via a *session-key exchange protocol*. (In practice, once κ_i is generated, the users will utilize κ_i for secure communication.) For simplicity assume that each execution of the protocol is always initiated by Alice.

After being installed the virus can (1) read all the internal data of the victim, (2) compute an arbitrary (efficient) function Γ on this data[4] (the only restriction that we put on Γ is that the length of it's output is limited), and (3) send the result of the computation back to the adversary. Note, that we assume that the adversary is not allowed to modify the programs running on the users' machines. Informally speaking the goal of the adversary is to successfully *break some test session* T_{test} (of her choice), by achieving one of the following goals:

1. learn κ_{test},

[4] We will model it by simply asking the adversary to produce a description of Γ (as a boolean circuit). The size of the circuit does not need to be restricted (although in practice it would probably be), as long as it is polynomial. Observe, that we do not need to consider the case of *interactive* viruses (that would be allowed to engage in a interactive massage exchange with the adversary), since the circuit may contain the description of the entire state of the adversary.

2. convince at least one of the players to accept some κ'_{test} about which the adversary has some significant information, or
3. make A and B agree on different keys.

Clearly, if the adversary installs a virus on one of the users' machines in the session T_{test} then she can instruct the virus to retrieve κ_{test} (since in a usual scenario the session key κ_i is short[5]). Therefore, we are interested only in the adversary breaking those sessions T_{test} during which no virus was installed (neither on the machine of A nor on the one of B).

Traditionally when considering forward security (see e.g. [16]) one allows the adversary to learn all the session keys except of the challenge key κ_{test}. In our model this ability of $\mathcal{E}$ comes from the fact that the adversary can corrupt all sessions except of T_{test} (we will actually allow the adversary to ,,corrupt a session" that has already ended some time ago). Finally, let us remark that in this model we assume that the players can erase their data (in particular, after the session T_i the players would erase κ_i). Actually, we will assume that the only data that is not erased between the sessions is the secret key K.

4.2 A more formal description of the model

We are now going to define the model more formally. Our definitions are inspired by the definitions of the security of key-exchange protocols (esp. [7]). For the sake of simplicity we assume that the protocol is executed just between two fixed parties, and concurrent execution of the sessions is not allowed, i.e. the users simply execute one session after another. Giving a complete security definition (e.g. in the style of [7]) of intrusion-resilience in our model, remains an open task.

The *key exchange scheme* is a tuple $(A, B, \alpha, \beta, \gamma, \delta, \chi)$, where $\alpha, \beta, \gamma, \delta, \chi$ are some polynomials and A and B are interactive Turing machines, taking as input a security parameter 1^k and a secret key $K \in \{0,1\}^{\alpha(k)}$. The adversary $\mathcal{E}$ is a PPT Turing Machine taking as input 1^k. The execution is divided into the sessions $T_1, T_2, \ldots, T_{\chi(k)}$. The execution of each T_i looks as follows:

1. The machines A and B receive uniformly (and independently) chosen random inputs $r_A \in \{0,1\}^{\beta(k)}$ and $r_B \in \{0,1\}^{\beta(k)}$ (respectively).
2. Machines start exchanging messages. The adversary can eavesdrop the messages. She can also prevent some of the messages from arriving to the destination and fabricate new messages. At the beginning A sends a unique message $start$ to B (so the adversary knows that a new session started).
3. At the end of the session the machines (privately) output an agreed key $\kappa_i \in \{0,1\}^{\delta(k)}$ (hopefully they output the same value).
4. Now adversary may choose to *corrupt the session* T_i (each session T_i may be corrupted at most once in the entire execution of the protocol). In this case she produces a description of a boolean circuit C (which models the virus) computing a function $\Gamma : \{0,1\}^w \rightarrow \{0,1\}^{\gamma(k)}$ (w is an arbitrary value, we will always

[5] Even if one would develop a scheme in which κ_i is too large to be retrieved, the adversary could simply instruct the virus to steal the data that is encrypted with κ_i.

have $\gamma(k)\chi(k) < \alpha(k)$) such that the size of C is arbitrary (however, it has to be polynomial in the security parameter, as the adversary is polynomially-bounded); she learns the value of $\Gamma(r_A, r_B, K)$. Note that the circuit Γ may depend on the messages communicated between the parties in this session (as the adversary can eavesdrop the communication). Therefore Γ „has a complete view" on the internal states of the parties during the session. Thus in particular the value of $\Gamma(r_A, r_B, K)$ may include the encoding of κ_i (if this is the wish of the adversary). Also note that our model is actually stronger than what we need in practice (as we assume that Γ has simultaneous access to both A and B, without restricting the amount of data that she needs to transfer between the parties, to perform the computation).

5. The adversary may decide to make a corruption like in Point 4 even long time after the session T_i is finished (one can imagine that the descriptions of the states of A and B at the end of T_i are deposited somewhere and the adversary may decide to access it anytime). This may seem an artificial strengthening of the model. However, in fact it simplifies things, as it allows us to model the fact that κ_i may become known to the adversary at some later point. An alternative would be to introduce special type of session-key-queries [7] that the adversary may ask to learn κ_i after the end of T_i.

Let $\mathcal{C}$ be a set of all sessions that were ever corrupted. Clearly, the adversary succeeds if for some session $T_i \notin \mathcal{C}$ users A and B outputted different keys. If this is not the case then at the end of the execution the adversary decides that some $T_{test} \notin \mathcal{C}$ will be her *test-session*. In this case her task will be to distinguish κ_{test} from a truly random key of the same length. Of course we need to require that at least one of A and B actually outputted some key κ_{test} (by blocking the message flow the adversary can clearly prevent the parties from reaching any agreement). The distinguishing game is as follows:

1. Let $r \in \{0, 1\}$ be random.
2. If $r = 0$ then pass κ_{test} to the adversary. Otherwise generate a random $K' \in \{0, 1\}^{\delta(k)}$ and pass it to the adversary. The adversary outputs some $r' \in \{0, 1\}$. We say that she *won the distinguishing game* if $r = r'$.

Definition 1. *We say that a key generation scheme as above is* intrusion-resilient *if for any PPT $\mathcal{E}$*

1. *the chances that in some session $T_i \notin \mathcal{C}$ machines A and B outputted different keys are negligible (in k), and*
2. *the chances that $\mathcal{E}$ wins the distinguishing game, are at most negligibly (in k) grater than $1/2$.*

For simplicity we assumed that the adversary is allowed to retrieve $\gamma(k)$ bits of data in each corrupted session. More generally, one could give a bound of the total number of bits retrieved by the adversary in all corrupted sessions.

4.3 The protocol for intrusion-resilient authenticated key exchange

Preliminaries Let f be a $(\sigma, \tau, \lambda, \mu)$-secure expansion function. Let *MAC* be a message authentication scheme secure against adaptive chosen message attack. Assume that for a security parameter 1^k the length the secret key of *MAC* is $\nu(k)$. Let $H : \{0,1\}^{\lambda(k)} \to \{0,1\}^{\nu(k)}$ be a hash function (modeled as a random oracle). Let $(G, encr, decr)$ be a semantically secure public-key encryption scheme. In order to achieve forward-security we will use the public-key encryption in a standard way (see e.g. [16, 1]): Alice will generate an ephemeral (public key, private key) pair [6] and send the public key (in an authenticated way) to Bob, Bob will generate the session key κ and send it (encrypted with Alice's public key) back to Alice (who can later decrypt κ).[7] Afterwards, the ephemeral keys are erased.

The protocol Fix some value of the security parameter k. Let $\mathcal{R} = \{0,1\}^{\tau(k)}$ and let $\mathcal{Y} = \{0,1\}^{\mu(k)}$. Assume that Alice and Bob share a random secret key $K = (R_A, R_B) \in \mathcal{R}^2$. In each session T_i the players execute the following protocol.

1. Alice generates a random $Y_A \in \mathcal{Y}$ and sends it to Bob.
2. Bob generates a random $Y_B \in \mathcal{Y}$ and sends it to Alice.
3. Both parties calculate $S := f(R_A, Y_A) \oplus f(R_B, Y_B)$ and $S' := H(S)$.
4. Alice generates a public key E and sends $(E, \mathit{MAC}_{S'}(\text{A:}E))$ to Bob.
5. Bob verifies the correctness of the authentication tag. If it is correct then he generates a random κ_i and sends $(encr_E(\kappa_i), \mathit{MAC}_{S'}(encr_E(\text{B:}\kappa_i))$ to Alice. He outputs κ_i.
6. Alice verifies the correctness of the authentication tag. If it is correct then she decrypts κ_i and outputs it.
7. The players erase all their internal data (including κ_i and random inputs), except of the long-term key K.

The role of labels „A:" and „B:" is to prevent the adversary from bouncing the message sent by Alice in Step 4 back to her in Step 5.

The bound on the amount of retrieved data An important parameter that needs to be fixed is the amount of data that the virus can retrieve in each session, i.e. the value of $\gamma(k)$. If the adversary corrupts some sessions than at any point of the execution of the key-exchange scheme she knows the value of some function $\tilde{h}$ of K. We can think about $\tilde{h}$ as changing dynamically after each session. After execution of i sessions the length of the output of $\tilde{h}$ is at most the sum of

- $i \cdot \gamma(k)$ (since she could have corrupted at most i sessions do far), and
- $i \cdot \nu(k)$ (since she could have learned i keys of *MAC* scheme[8])

[6] *Ephemeral key* is a key that is generated just for some particular session (and it is later erased).
[7] In [16] it is actually done by exchanging Diffie-Hellman ephemeral keys.
[8] We have to add it because the definition of the security of *MAC* does not imply the secrecy of all the bits of the key.

Since the maximal number of sessions is $\chi(k)$ we know that the output of $\tilde{h}$ is of a length at most

$$\chi(k) \cdot (\gamma(k) + \nu(k)).$$

Therefore if we want this value to be at most $\sigma(k)$ we have to set

$$\gamma(k) := \sigma(k)/\chi(k) - \nu(k). \tag{2}$$

This ensures that the information that Eve has about K is at most $\sigma(k)$ bits.

4.4 Sketch of the security proof

We give an informal security proof. Consider some uncorrupted session T_i. Let us first consider the case when the adversary wants to break it by disrupting (by stealing and substituting messages) the communication. Let S_A and S_B be the values of S computed by A and B (resp.) in Step 3. If the execution of the protocol was not disturbed by the adversary then we have $S_A = S_B$. By the security of f in the BSM the adversary has almost no information about the values S_A and S_B (i.e. their distribution is negligibly far from uniform from her point of view). Note that this holds even if she was disrupting the communication between the parties. The only thing that the adversary could possibly do is to force S_A and S_B to be such that they are not equal, but they are not independent either. For example by modifying the message Y_A (sent in Step 1) she could make $S_A \oplus S_B$ to be equal to some value $S_\oplus$ chosen by her.[9]

This is why, before using S we hash it (in Step 3): $S' := H(S)$. Let $S'_A := H(S_A)$ and let $S'_B := H(S_B)$. Clearly the chances of $\mathcal{E}$ of guessing S_A or S_B in polynomial time are negligible. Therefore (since we model the hash function as a random oracle) we can assume that (except with negligible probability) from the point of view of $\mathcal{E}$ the distributions of the values S'_A and S'_B entirely uniform. Moreover, one of the following has to hold (except with negligible probability):

1. $S'_A = S'_B$, or
2. S'_A and S'_B are independent.

Assume that the first case holds. Then, the adversary is not able to fabricate messages in Steps 4 and 5, without breaking the MAC. The security of κ_i follows now from the security of the encryption scheme (if the adversary could distinguish κ_i from a random key, then she could clearly break the semantic security of $(G, encr, decr)$).

In the second case the parties easily discover that the adversary was interfering with their communication. This is because the adversary needs to create (in Steps 4 and 5) valid pairs (message,*MAC*), without having any information about the secret keys. Again, she cannot do it without breaking the *MAC*.

[9] Consider for example the scheme from Section 3.1. Write $Y_A = (Y_1, \ldots, Y_m)$. Suppose the adversary stored the first row ($R_A(1)$) of Alice's randomizer (she should have enough memory to do it) and she modified Y_A only on the first component (Y_1). Let Y'_A be the result of this modification. Clearly almost always $f_A(R_A, Y_A) \neq f_A(R_A, Y'_A)$, however $f_A(R_A, Y_A) \oplus f_A(R_A, Y'_A)$ is known to the adversary.

Now suppose that the adversary wants to distinguish κ_i from a random key, after the session is completed. If she now corrupts some future session T_j then she can of course recover the key S' used in session T_i. However, now it is too late (as the key S' is used only for authentication). Therefore, the security of κ_i again follows from the semantic security of the encryption scheme.

5 Intrusion-Resilient Entity Authentication

In this section we informally describe a practical intrusion-resilient method for entity authentication. In order to achieve such entity authentication one could of course use the scheme from Section 4, however this is an overkill and for practical applications a much simpler method suffices. The idea is as follows. Suppose a user U wants to authenticate to a server S. We are interested in methods of intrusion-resilient authentication for the user U. We will consider *only intrusions into U*. This corresponds to a practical situation in which the computers of the users are usually much more vulnerable for the attacks then the computer of the server.

Assume that the parties have already established a channel C between S and U that is authentic only from the point of view of the user (i.e. U knows that whatever comes through this channel is sent by U and whatever is sent through it can be read only by U). Now, the user wants to authenticate to the server. This is a typical scenario on the Internet, where C is established e.g. using SSL (and the server authenticates with a certificate). In such case usually U authenticates by sending his password over C. This method is clearly not intrusion-resilient because once a virus enters the machine of U he can retrieve the password (or record the key-strokes if the password is memorized by a human).

We propose an authentication method that is intrusion-resilient in the following sense: the adversary can authenticate as a user only during the period when the virus is installed on the victim's machine. After it is removed (e.g. by a virus-removal tool) the adversary looses the ability to impersonate the user. Again, we will use the assumption that the secret key K of the user is too large to be fully downloaded (and again, we allow the virus to perform arbitrary efficient computations of the victim's machine). Our method is not as safe as the methods based on the trusted hardware (e.g. RSA SecurID). Therefore it should not be recommend e.g. for e-banking. However it can be used in less sensitive applications (e.g. remote login to the local network), especially when it is combined with other methods authentication.

5.1 The protocol

We propose an intrusion-resilient authentication method based on the BSM key-derivation function. Let f be a function that is $(\sigma, \tau, \lambda, \mu)$-secure in the BSM. Fix some security parameter k. The secret key K is simply the randomizer $R \in \{0,1\}^{\tau(k)}$. The key is stored both on the user's machine and on the server. The protocol is as follows (all the communication is done via the channel C).

1. The server selects a random $Y \in \{0,1\}^{\mu(k)}$ and sends it to Bob.

2. Bob replies with $f(R, Y)$.
3. Alice verifies the correctness of Bob's reply.

It is easy to see that as long as the adversary does not retrieve from the user's machine more than $\sigma(k)$ bits, she has negligible chances of being able to reply correctly to the challenge Y. Observe that since we assume that the server is secure (i.e. there are no intrusions to him) hence one could generate K pseudo-randomly and just store the seed on the server. For example: the seed s could be a key to the block-cipher B and one could set $K := (B_s(1), B_s(2), \ldots, B_s(j))$, for some appropriate parameter j (this method allows for a quick access to any part of K).

5.2 Discussion

The main drawback of our protocol is that an on-line intruder can authenticate from the user's machine (without the user being even aware that something wrong is happening). As a partial remedy we suggest that the user could be required to authenticate with two private keys K_1 and K_2, and to store each of them on a separate DVD disc. In this case the authentication process would require physical action of replacing one DVD with another (assuming that there is only one DVD drive in the machine). Note, that this method does not work if we assume that the adversary is able to store large amounts of data on user's hard-disc (as in this case he can make a local copy of the DVDs containing the key).

6 Open Problems

We did not provide concrete numerical examples of the parameters of our schemes. This parameters depend on the choice of the key-expansion function f, as our protocols can be used with any function f that was proven secure in the BSM [2, 12, 17, 21]. For the entity authentication one could also consider a scheme in which the server simply asks (in Step 1, Section 5.1) for the values of k random positions on K.[10] It remains an open task to analyze the security of our protocols for the parameters that are relevant for the practical applications (e.g. length of the key is 10^{10} and the maximal amount of retrieved data is 10^9).

Another open problem is to implement other cryptographic tasks (as asymmetric encryption and signature schemes) in our model.

References

1. Ross Anderson. Two remarks on public key cryptology. Technical report, University of Cambridge, Computer Laboratory, 2002.
2. Y. Aumann, Y. Z. Ding, and M. O. Rabin. Everlasting security in the bounded storage model. *IEEE Transactions on Information Theory*, 48(6):1668–1680, 2002.

[10] One can analyze the security of such scheme using the the results from [20, 5].

3. 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.
4. C. Cachin, C. Crepeau, and J. Marcil. Oblivious transfer with a memory-bounded receiver. In *39th Annual Symposium on Foundations of Computer Science*, pages 493–502, 1998.
5. C. Cachin and U. Maurer. Unconditional security against memory-bounded adversaries. In *Advances in Cryptology - CRYPTO '97*, pages 292–306, 1997.
6. Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 255–271, 2003.
7. Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474, 2001.
8. Y. Z. Ding. Oblivious transfer in the bounded storage model. In *Advances in Cryptology - CRYPTO 2001*, pages 155–170, 2001.
9. Yevgeniy Dodis. *Exposure-Resilient Cryptography*. PhD thesis, Massachussetts Institute of Technology, August 2000.
10. Yevgeniy Dodis, Matt Franklin, Jonathan Katz, Atsuko Miyaji, and Moti Yung. A generic construction for intrusion-resilient public-key encryption. In *RSA Conference, Cryptography Track (CT-RSA)*, Feb 2004.
11. Yevgeniy Dodis, Matthew K. Franklin, Jonathan Katz, Atsuko Miyaji, and Moti Yung. Intrusion-resilient public-key encryption. In *Topics in Cryptology - CT-RSA 2003, The Cryptographers' Track at the RSA Conference 2003, San Francisco, CA, USA, April 13-17, 2003, Proceedings*, volume 2612 of *Lecture Notes in Computer Science*, pages 19–32, 2003.
12. Stefan Dziembowski and Ueli Maurer. Optimal randomizer efficiency in the bounded-storage model. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 17(1):5–26, January 2004.
13. Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA, 2004.
14. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
15. Gene Itkis and Leonid Reyzin. Sibir: Signer-base intrusion-resilient signatures. In *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*, pages 499–514, 2002.
16. Hugo Krawczyk. Intrusion-resilient public-key encryption. In *Proceedings of the 1996 Symposium on Network and Distributed System Security (SNDSS '96)*, page 114. IEEE Computer Society, 1996.
17. Chi-Jen Lu. Encryption against storage-bounded adversaries from on-line strong extractors. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 17(1):27–42, January 2004.
18. U. Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 5(1):53–66, 1992.
19. Tal Moran, Ronen Shaltiel, and Amnon Ta-Shma. Non-interactive timestamping in the bounded storage model. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 460–476, 2004.

20. Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996.
21. Salil P. Vadhan. Constructing locally computable extractors and cryptosystems in the bounded-storage model. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 17(1):43–77, January 2004.