Intrusion-Resilience via the Bounded-Storage Model*

Stefan Dziembowski**

Institute of Informatics,
Warsaw University, Poland
and
Institute for Informatics and Telematics
CNR Pisa, Italy

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: session-key generation and entity authentication. Our method is based on the results from the Bounded-Storage Model.

1 Introduction

In the contemporary Internet environment, computers are often exposed to 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 user is usually advised to use virus and spyware removal tools. These 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

[*] This is an extended version of a report [Dzi05] that appeared on the eprint archive.

[**] Partially supported by the EU ECRYPT grant IST-2002-507932 and by the Polish KBN grant 4 T11C 042 25. Part of this work was carried out during the tenure of an ERCIM fellowship. Another part of this work was done when the author was employed at the Institute of Mathematics of the Polish Academy of Sciences.

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 a password memorized by the user then the virus can wait until the password is typed and then record the key-strokes.

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 [IR02] (see also [DFK$^+$03]) and can be viewed as a combination of forward and backward security.[1] A cryptosystem is *forward-secure* if an 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. [DvOW92,Kra96]), digital signatures (this research was initiated by Ross Anderson, see [And02]) and public-key encryption [CHK03]. 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 this was achieved by distributing the secret key among a group of participants (e.g. in [IR02] 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. [Dod00]). The differences from 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.

Our model can be viewed as a generalization of the model of Kelsey and Schneier [KS99]. In their model the adversary is allowed to access individual bits of the secret key (this is justified by an assumption that the access to the memory is slow). In this model they show a simple authentication protocol (the secret key is a long random string of bits; in order to verify the authenticity of the client the server asks for the values of some randomly chosen positions of the secret key). In Sect. 5.2 we show that this protocol is also secure in our model.

Independently[2] (but earlier) a similar model was introduced by Dagon et al. [DLL05]. They propose a system (called *VAST*) for securely storing secret data

[1] There seems to be some confusion in the literature about the terminology. What is called *forward security* in [And02] is called *backward security* in [IR02,DFK$^+$04]. In this paper we use the terminology of [IR02].

[2] We became aware of this work after submitting our paper to TCC.

on devices that can be subject to an intrusion e.g. by a virus. They assume that such data is encrypted by a weak (human-memorized) password (let T denote the resulting ciphertext and let π be the password). To prevent the adversary from downloading T and cracking the password (i.e. performing a dictionary attack on π) on his own machine, they design their protocols so that T is too large to be fully downloaded. In order for this to make sense they need to assume that the computing power of the virus is limited (so the virus cannot perform the password-cracking on the victim's machine). This is in contrast to our model, where we can grant the virus a right to perform an arbitrary (polynomial-time) computation on the victim's data. Another difference is that they assume that the adversary does not have a full access to the victim's machine. In particular when the user is interacting with VAST the virus should not have access to the keyboard. This is because when the user enters the password π to the machine the virus can learn π by recording the key-strokes.

1.2 Our contribution

We propose a new method for constructing intrusion-resilient protocols for the session-key generation and entity authentication (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 of huge size (e.g. K is of size 5 GB). More precisely, we will grant the adversary the power to break into the honest user's machine and take full control over it. We will assume that the adversary is able to perform arbitrary (efficient) computation on victim's data. Clearly, during the period of the break-in one cannot hope for much security, since the adversary has a complete knowledge about the behavior of one of the honest users (and hence she can e.g. impersonate the user or steal the session key). So the intrusion-resilience is the maximum what we can hope for. We achieve it by assuming that the amount of data that the adversary can retrieve is much smaller than K (say it is 0.5 GB). This assumption may be quite practical as in many situations transmitting unnoticeably 0.5 GB of data is hard. Observe that if the secret key is of size 1 KB then the virus can e.g. post it on some Usenet group, so that the author of the virus can download it anonymously. Clearly this is much harder if the secret is huge.

Another motivation is that protocols that are secure in our model have a high level of resiliency against side-channel analysis [KSWH00]. Recall that the side-channel attacks allow the adversary to obtain some information about the users' secrets by observing the behavior of the implementation of the protocol. In practice the full protection against such attacks is hard, and we can only hope for minimizing the amount of leaked information. The assumptions that we make in our model guarantee that even if some information about the secrets is leaked, the protocols are still secure.

Our method is based on the theory of the Bounded-Storage Model (see Sect. 3). In the BSM one constructs protocols secure under the assumption that the amount of data that the adversary can store is smaller that the amount of data that can be broadcasted (e.g. by a satellite). The fact that he theory of the BSM has applications here may seem surprising at the first sight, as in some sense the assumptions in the BSM are opposite to ours. However, as it turns out, these models show similarities and in fact theorems that were proven in the BSM are useful for us.

Our exposition is rather informal, as we mostly aim at introducing the model and showing its power, not at providing ready to use practical solutions for concrete problems. For the same reason we do not provide numerical examples and we do not give comparisons between security levels of different schemes presented in the paper. Nevertheless, we believe that the protocols provided here (or their variants) may find practical applications.

Finally, let us note that our results are proven in the random oracle model (see Sect. 2.4).

1.3 The contribution of [CDD$^+$05]

The entity authentication protocol that we present in our paper was independently constructed and analyzed by Cash et al. [CDD$^+$05]. Moreover, they improve our results by constructing a session-key generation protocol without the random oracle assumption. They also provide some concrete numerical examples of the parameter values that can be used in practical implementations.

2 Preliminaries

2.1 Probability theory

The *min-entropy* of a probability distribution P_X is defined as

$$H_\infty(X) := \min_{x \in \mathcal{X}}(-\log_2(P_X(x))).$$

If X is a random variable and A is an event then P_X is the distribution of X and $P_{X|A}$ is a conditional distribution of X given A. In this case we define $H_\infty(X) := H_\infty(P_X)$ and $H_\infty(X \mid A) := H_\infty(P_{X|A})$. For more on min-entropy and its relation to the standard Shannon entropy see e.g. [Cac97].

Let the *statistical distance* between random variables X and X' distributed over the same set $\mathcal{X}$ be defined as

$$\delta(X, X') := \frac{1}{2}\sum_{x \in \mathcal{X}} |X(x) - X'(x)|$$

We will also say that X *is* $\delta(X, X')$*-far from* X'. If U is a random variable with uniform distribution over $\mathcal{X}$ then define $d(X) := \delta(X, U)$. The above notation extends in a natural way to probability distributions.

2.2 Message Authentication Codes

We will use the following (simplified) security definition of the Message Authentication Codes (MACs). For a more complete definition the reader may consult e.g. [Gol04]. MAC is an algorithm which takes as an input a security parameter 1^k, a random secret key $S \in \{0,1\}^{\lambda(k)}$ (where λ is some polynomial) and a message $M \in \{0,1\}^*$. It outputs an *authentication tag* $MAC_S(M, 1^k))$ (we will sometimes drop 1^k). It is *secure against an adaptive chosen-message attack* if any probabilistic polynomial time (PPT) adversary (taking as input 1^k) has negligible[3] (in k) probability of producing a valid pair $(M, MAC_S(M, 1^k))$, after seeing an arbitrary number of pairs

$$(M_1, MAC_S(M_1, 1^k)), (M_2, MAC_S(M_2, 1^k)) \ldots$$

(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 $(G, encr, decr)$, where G is a PPT *key-generation algorithm* taking as input 1^k and returning as output a (private-key,public-key) pair (E, D), *encr* is an polynomial-time algorithm taking as input 1^k, a message $M \in \{0,1\}^*$ and a public key E and returning a *ciphertext* $C = encr_E(M)$, and *decr* is an algorithm taking as input a private key D a ciphertext C and returning a message $M' = decr_D(C)$. We require that always $M = decr_D(encr_E(M))$. 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 $C = encr_S(M_r)$, where $r \in \{0,1\}$ is random. She has to guess r. We say that $(G, encr, decr)$ is *semantically secure* [GM84] 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 [Gol04].

2.4 Random Oracle Model

We prove the security of our protocol in the *Random Oracle Model* [BR93]. More precisely, we will model a hash function $H : \{0,1\}^i \to \{0,1\}^j$ as a *random oracle*, i.e. a black box containing a random function $h : \{0,1\}^i \to \{0,1\}^j$. 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] A function $f : \mathcal{N} \to \mathcal{R}$ is *negligible (in k)* if for every $c \geq 1$ there exists k_0 such that for every $k \geq k_0$ we have $|f(k)| \leq k^{-c}$.

3 Bounded Storage Model

We will use the results from the Bounded-Storage Model, introduced by Maurer in [Mau92]. So far, this model was studied in the context of *information-theoretically secure* encryption [ADR02,DM04b,Lu04,Vad04,Din05], key-agreement [CM97,DM04a], oblivious transfer [CCM98,Din01,DHRS04] and time-stamping [MSTS04]. 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 has been shown in [ADR02,DM04b,Lu04,Vad04] 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, so 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[4], 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, the size of Eve's memory and the length of the output of f are functions of k, i.e. $t = \tau(k)$, $s = \sigma(k)$, and $n = \nu(k)$ (with $\nu(k) \geq k$). Also, 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 (σ, τ, ν, μ)*-secure in the bounded-storage model* if for

[4] Formally, a sequence of probabilities $p_0, p_1, \ldots$ is *overwhelming* if the function $f(k) = 1 - p_k$ is negligible.

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 $\nu(k)$-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.[5]

Several key expansion functions were proven secure in the past couple of years (see for example [ADR02,DM04b,Lu04,Vad04]). In the next section we present an example of such a function, taken from [DM04b]. We have chosen the function of [DM04b] because we believe that it is the simplest one. The reader familiar with the BSM literature can safely skip the next section.

3.1 The scheme of [DM04b].

The randomizer $R \in \mathcal{R} = \{0,1\}^t$ is interpreted as being arranged in a matrix with m rows, denoted $R(1), \ldots, R(m)$, 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 Fig. 1). The initial key $Y = (Y_1, \ldots, Y_m) \in \mathcal{Y} = \{1, \ldots, l\}^m$ selects one starting point within each row, and the expanded key $X = (X_1, \ldots, X_n)$ 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 $f : \mathcal{R} \times \mathcal{Y} \rightarrow \{0,1\}^n$ is defined as follows. For $r \in \mathcal{R}$ and $Y = (Y_1, \ldots, Y_m) \in \mathcal{Y}$,

$$f(R,Y) := \left(\bigoplus_{i=1}^{m} R(i, Y_i), \ldots, \bigoplus_{i=1}^{m} R(i, Y_i + n - 1) \right), \tag{1}$$

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

The above function f was proven secure in [DM04b], assuming that memory of the adversary has a size that is a constant fraction $c < 1$ of the randomizer. For the practically looking parameters this constant should be around 8%, i.e. $\sigma(k) := \tau(k) \cdot 0.08$. See [DM04b] for details.

4 Intrusion-Resilient Session-Key Generation

By *session-key generation* 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* session-key generation.

[5] Formally we could do it by allowing $\mathcal{E}$ and f to take extra random inputs $r_{\mathcal{E}}$ and r_f, resp. This does not give any extra power to the adversary, for the following reasons: (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 herself 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 $r_{\mathcal{E}}$.

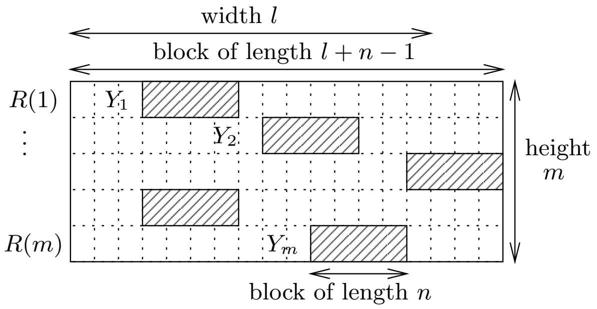

Fig. 1. Illustration of the scheme for deriving an expanded n-bit key $X = (X_1, \ldots, X_n)$, to be used as a one-time pad, from a short secret initial key $Y = (Y_1, \ldots, Y_m)$. The randomizer R is interpreted as a $m \times (l + n - 1)$ matrix with rows $R(1), \ldots, R(m)$ of length $l+n-1$. 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 (polynomially 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 $T_1, T_2, \ldots$ (the number of sessions will be bounded). At the beginning of the session the users are allowed to get some fresh random input. At the end of each session T_i the users output a new *session key* κ_i. (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 do the following.

1. Read all the internal data of the victim.
2. Compute an arbitrary function Γ on this data. We will model it by asking the adversary to produce a description of Γ as a boolean circuit. The only restriction that we put on Γ is that the length of its output is limited (observe however that since Eve is polynomially-bounded the size of Γ has to be polynomial). Note also 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.
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},
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 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[6]). 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. [Kra96]) 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 compromise all sessions except of T_{test} (we will actually allow the adversary to ,,compromise a session" that has already ended some time ago). Finally, let us remark that in this model we assume that the players can reliably 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. [CK01]). 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 definition (e.g. in the style of [CK01]) remains an open task.

The *session-key generation scheme* is a tuple $(A, B, \alpha, \beta, \gamma, \delta, \chi)$, where α, β, γ, δ, χ 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 $\kappa_i \in \{0,1\}^{\delta(k)}$. If the traffic was not disturbed by the adversary then they have to output the same value.

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

4. Now the adversary may choose to *compromise the session* T_i (each session T_i may be compromised at most once in the entire execution of the protocol). In this case the following happens.
   (a) Eve produces a description of a boolean circuit C (which models the virus) computing a function $\Gamma : \{0,1\}^w \to \{0,1\}^{\gamma(k)}$ (w is an arbitrary value). Clearly we will always have $\gamma(k)\chi(k) < \alpha(k)$, since otherwise Eve could retrieve the entire secret key K. The size of C is arbitrary (however, it has to be polynomial in the security parameter, as the adversary is polynomially-bounded).
   Note that we assume a uniform bound $\gamma(k)$ on the amount of bits that the adversary is allowed to steal in each compromised session. More generally, one could give a bound on the *total* number of bits retrieved by the adversary in all compromised sessions.
   (b) Eve learns the value of $\Gamma(r_A, r_B, K)$.

   Observe that the function Γ ,,has a complete view" of 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 compromise a session (in the same way as 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 them at any later time). 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. Alternatively, we could introduce a special type of session-key-queries [CK01] that the adversary may ask to learn κ_i after the end of T_i.

Let $\mathcal{C}$ be the set of all compromised sessions. Clearly, the adversary wins 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 $\kappa' \in \{0,1\}^{\delta(k)}$ 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* $(A, B, \alpha, \beta, \gamma, \delta, \chi)$ *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) greater than $1/2$.*

4.3 The protocol for intrusion-resilient session-key generation

Preliminaries Let f be (σ, τ, ν, μ)-secure in the BSM. 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 $\lambda(k)$. Let $H : \{0,1\}^{\nu(k)} \rightarrow \{0,1\}^{\lambda(k)}$ 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. [DvOW92,Kra96]): Alice will (1) generate an ephemeral (public key, private key) pair[7] and (2) 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).[8] 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 $K = (R_A, R_B) \in \mathcal{R}^2$ and hence $\alpha(k) := 2 \cdot \tau(k)$. 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 $(E, MAC_{S'}(\text{A:}E))$ to Bob.
5. Bob verifies the correctness of the authentication tag. If it is correct then he generates a random κ_i and sends $(encr_E(\kappa_i), MAC_{S'}(encr_E(\text{B:}\kappa_i))$ 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.

[7] *Ephemeral key* is a key that is generated just for some particular session (and it is erased later).

[8] In [DvOW92,Kra96] it is actually done by exchanging Diffie-Hellman ephemeral keys, i.e. doing authenticated Diffie-Hellman key agreement.

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 compromises some sessions than at any point of the execution of the scheme, then 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 compromised at most i sessions do far), and
- $i \cdot \lambda(k)$ (since she could have learned i keys of the MAC scheme[9])

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

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

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

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

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

4.4 The security of the protocol

We prove the following.

Theorem 1. *The protocol in Sect. 4.3 is intrusion resilient.*

Proof (sketch). Fix some uncompromised 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.[10]

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 are

[9] 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.

[10] Consider for example the scheme from Sect. 3.1. Write $Y_A = (Y_1, \ldots, Y_m)$. Suppose the adversary stored the first row ($R_A(1)$) of R_A (she should have enough memory to do it) and she modified $Y_A = (Y_1, \ldots, Y_m)$ (sent is Step 1) 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, $f_A(R_A, Y_A) \oplus f_A(R_A, Y'_A)$ (and hence $S_A \oplus S_B$) is known to the adversary.

negligible. This is because the distributions of S_A and S_B are negligibly far from a uniform distribution over $\{0,1\}^{\nu(k)}$ and we assumed that $\nu(k) \geq k$. 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 are 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 $(G, \mathit{encr}, \mathit{decr})$).

In the second case, the parties easily discover that the adversary was interfering with their communication. This is because if the adversary wants to prevent them from discovering this, then she 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.

Now suppose that the adversary wants to distinguish κ_i from a random key, after the session is completed. If she compromises some future session T_j then she can of course recover the key S' used in session T_i (if she stored Y_A and Y_B from 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. □

4.5 An alternative protocol

In this section we show another variant of the protocol from Sect. 4.3. The main difference is that instead of using a BSM-secure key derivation function f, we will use a function $\tilde{f} : \mathcal{R} \times \mathcal{Y} \to \{0,1\}^k$ that is not BSM-secure, but still works for our purposes. Again, let k be a security parameter and suppose that the randomizer R is a random element from $\mathcal{R} = \{0,1\}^{\tau(k)}$. Let $\mathcal{Y} := \big\{(Y_1, \ldots, Y_k) \in \{1, \ldots, \tau(k)\}^k \mid Y_1 < \cdots < Y_k\big\}$. Thus $\mathcal{Y}$ can be viewed as a set of all k-element subsets of $\{1, \ldots, \tau(k)\}$. First, define

$$\varphi((R_1, \ldots, R_{\tau(k)}), (Y_1, \ldots, Y_k)) := (R_{Y_1}, \ldots, R_{Y_k}).$$

Let H be a hash function. We set

$$\tilde{f}(R, Y) := H(\varphi(R, Y)).$$

In other words: we just pick random positions of the secret key, concatenate them and hash the result. Of course usually $\tilde{f}$ is not secure in the BSM as the hash functions belong to the complexity-theoretic world. However, if we model H as a random oracle, then the value of $\tilde{f}(R, Y)$ is random from the point of view of the adversary, unless she managed to guess the value of $\varphi(R, Y)$. So, if we want to use $\tilde{f}$ instead of f in the protocol from Sect. 4.3, then we have to show that the probability of any adversary of guessing $\varphi(R, Y)$ correctly, is negligible (for the appropriate choice of the parameters), even when the adversary is given $h(R)$ and Y (for some $h : \{0,1\}^{\tau(k)} \to \{0,1\}^{\sigma(k)}$ chosen by her). If we model the adversary's guess as a function g we can formalize this requirement as follows.

Lemma 1. *Suppose $\sigma(k) = (1-\delta)\tau(k) - k$, for an arbitrary $\delta > 0$. For arbitrary functions $h : \{0,1\}^{\tau(k)} \to \{0,1\}^{\sigma(k)}$ and $g : \{0,1\}^{\sigma(k)} \to \{0,1\}^{k}$ we have that*

$$P\left(\varphi(R, Y) = g(h(R), Y)\right) \tag{3}$$

is negligible.

For the proof we need two other lemmas. The first lemma (proven in [CM97], see Lemma 3) is quite simple. It roughly states that the knowledge of s bits of a random string R reduces its min-entropy by around s, with a high probability.

Lemma 2 ([CM97]). *Let R be a random variable uniformly distributed over $\{0,1\}^t$. Let $h : \{0,1\}^t \to \{0,1\}^s$ be an arbitrary function. Then, with probability at least $1 - 2^k$ the variable $h(R)$ takes a value u such that*

$$H_\infty(R \mid h(R) = u) \geq t - s - k.$$

The second lemma (proven in [NZ96], see Lemma 11) is more complicated. Informally speaking, it states that if $R \in \{0,1\}^t$ is a random string with min-entropy $\delta \cdot t$ and $Y \in \mathcal{Y}$ is chosen uniformly at random, then $\varphi(R, Y) \in \{0,1\}^k$ has (with high probability) a min-entropy close to $\delta' k$, where δ' is some constant.

Lemma 3 ([NZ96]). *Let P_R be a probability distribution over $\{0,1\}^t$ with min-entropy δt. Suppose R is chosen according to P_R. Then, with probability at least $1 - \epsilon$ (over the choice of $y = Y$) the distribution of $P_{\varphi(R,y)}$ is ϵ-far from some distribution $P_{X'}$ whose min-entropy is $\delta' k$ where $\delta' := c\delta / \log(\delta^{-1})$ and $\epsilon := \max(2^{-ck}, 2^{-c\delta' l})$ for some constant c.*

Actually, the lemma that is proven in [NZ96] is stronger, as it does not require Y to be entirely uniform (see [NZ96] for details). We are now ready for the proof of Lemma 1.

Proof (of Lemma 1). To simplify the notation we set $s := \sigma(k)$ and $t := \tau(k)$. First, observe that by Lemma 2 we have that (except with a negligible probability 2^{-k}) the variable $h(R)$ takes a value u such that

$$H_\infty(R \mid h(R) = u) \geq t - s - k = \delta t. \tag{4}$$

So, suppose that such u was selected. We are now going to apply Lemma 3. Thus set $\delta' = c\delta/(\log \delta^{-1})$ and $\epsilon = \max\left(2^{-ck}, 2^{-c\delta' k}\right)$ (where c is some constant). Observe that δ' is constant and ϵ is negligible. Therefore (by Lemma 3) we know that with overwhelming probability Y took a value y such that the conditional distribution of

$$P_{\varphi(R,Y) \mid h(R)=u,Y=y} \tag{5}$$

is at most ϵ-far from a distribution $P_{X'}$ with min-entropy $\delta' t$. Assume that this indeed happened. If we want to maximize (3) we have to let g choose an element with the maximal probability according to the distribution $P_{\varphi(R,Y) \mid h(R)=u,Y=y}$. Clearly this probability is at most $2^{-H_\infty(P_{X'})} + \epsilon = 2^{-\delta' t} + \epsilon$ which is negligible in k.

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 Sect. 4; however, this is an overkill and for practical applications a much simpler method suffices. The idea is as follows. We will construct an *intrusion-resilient* scheme that allows a user U to authenticate to a server S. 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 (1) whatever comes through this channel is sent by U and (2) 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 practice usually U authenticates to S 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).

In this section we propose an authentication method that is intrusion-resilient (in the same sense as the protocols in the previous sections). Again, we will use the assumption that the secret key K of the user is too large to be fully downloaded. We allow the virus to perform arbitrary computations[11] of the victim's machine.

5.1 Our protocol

Let f be a function that is (σ, τ, ν, μ)-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.

[11] The computational power of the virus does not need to be limited in this case.

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

Now assume that the adversary retrieved at most $\sigma(k)$ bits of R. More precisely, assume that the adversary knows a value $h(R)$, where h is a function with the range $\{0,1\}^{\sigma(k)}$. It is easy to see that (by the security of f) she has negligible chances of being able to reply correctly to the challenge Y. Observe that if the adversary replies (in Step 2) with some value X, and Alice rejects this answer, than the adversary learns exactly one bit of information about R (namely that $f(R, T) \neq X$), which should be added to the total number of „retrieved" bits (if one want to achieve the security against multiple impersonation attempts).

Note 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 The protocol of [KS99]

In this section we note that in the protocol from Sect. 5.1 one can use a simpler function f than the functions secure in the BSM. Namely, the server can simply ask (in Step 1) for the values of k random positions on K. Formally, the challenge in Step 1 is a random k-element subset of the set $\{1, \ldots, \tau(k)\}$. The function f in Step 2 is replaced with φ (where φ was defined in Sect. 4.5). This is exactly the protocol of [KS99] (however in that paper it was analyzed in a weaker model where the adversary is allowed to access only the individual bits of the secret key). The security of this protocol follows from the analysis in Sect. 4.5.

6 Discussion

The main drawback of our protocols is that during the intrusion the virus can impersonate the user (and the user may not even be aware that something wrong is happening). As a partial remedy we suggest that the user could be required to split the private key into 2 halves 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 she can make a local copy of the DVDs containing the key).

7 Open Problems

It remains an open problem to examine which variant of the protocols described above is the best for practical applications. We did not provide a comparison between the protocols based on the BSM key-expansion function and the protocols based on the function φ (Sect. 4.5 and 5.2), as such comparison should depend on the concrete parameters that one wants to optimize (the size of the communicated data, computing time, level of security). For some choice of these parameters (long computing time, high level of security) it may be even practical to use protocols that perform computations on the entire randomizer. For example in the protocol in Sect. 4.5 one could use function $\tilde{f}$ that simply hashes the entire randomizer R concatenated with Y (i.e. set $\tilde{f}(R, Y) = H(R \cdot Y)$).

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

8 Acknowledgments

We would like to thank Krzysztof Pietrzak and Bartosz Przydatek for helpful discussions, and the anonymous referees for their comments.

References

[ADR02] 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.

[And02] R. Anderson. Two remarks on public key cryptology. Technical report, University of Cambridge, Computer Laboratory, 2002.

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

[Cac97] Christian Cachin. *Entropy Measures and Unconditional Security in Cryptography.* PhD thesis, ETH Zurich, 1997. Reprint as vol. 1 of *ETH Series in Information Security and Cryptography*, ISBN 3-89649-185-7, Hartung-Gorre Verlag, Konstanz, 1997.

[CCM98] 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.

[CDD$^+$05] D. Cash, Y. Z. Ding, Y. Dodis, W. Lee, R. Lipton, and S. Walfish. Intrusion-resilient authentication and key agreement in the limited communication model. Manuscript, 2005.

[CHK03] R. Canetti, S. Halevi, and J. 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.

[CK01] R. Canetti and H. 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.

[CM97] C. Cachin and U. Maurer. Unconditional security against memory-bounded adversaries. In Burton S. Kaliski Jr., editor, *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 292–306. Springer, 1997.

[DFK$^+$03] Y. Dodis, M. K. Franklin, J. Katz, A. Miyaji, and M. 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.

[DFK$^+$04] Y. Dodis, M. K. Franklin, J. Katz, A. Miyaji, and M. Yung. A generic construction for intrusion-resilient public-key encryption. In Tatsuaki Okamoto, editor, *CT-RSA*, volume 2964 of *Lecture Notes in Computer Science*, pages 81–98. Springer, 2004.

[DHRS04] Y. Z. Ding, D. Harnik, A. Rosen, and R. Shaltiel. Constant-round oblivious transfer in the bounded storage model. In M. Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 446–472. Springer, 2004.

[Din01] Y. Z. Ding. Oblivious transfer in the bounded storage model. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 155–170. Springer, 2001.

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

[DLL05] D. Dagon, W. Lee, and R. J. Lipton. Protecting secret data from insider attacks. In *Financial Cryptography and Data Security, 9th International Conference, FC 2005, Roseau, The Commonwealth of Dominica, February 28 - March 3, 2005,*, pages 16–30, 2005.

[DM04a] S. Dziembowski and U. Maurer. On generating the initial key in the bounded-storage model. In Jan Camenisch and Christian Cachin, editors, *Advances in Cryptology — EUROCRYPT '04*, volume 3027 of *Lecture Notes in Computer Science*, pages 126–137. Springer-Verlag, May 2004.

[DM04b] S. Dziembowski and U. Maurer. Optimal randomizer efficiency in the bounded-storage model. *Journal of Cryptology*, 17(1):5–26, January 2004.

[Dod00] Y. Dodis. *Exposure-Resilient Cryptography.* PhD thesis, Massachussetts Institute of Technology, August 2000.

[DvOW92] W. Diffie, P. C. van Oorschot, and M. J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2(2):107–125, 1992.

[Dzi05] S. Dziembowski. Intrusion-resilience via the bounded-storage model. Cryptology ePrint Archive, Report 2005/179, 2005. http://eprint.iacr.org/.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[Gol04] O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications.* Cambridge University Press, New York, NY, USA, 2004.

[IR02] G. Itkis and L. 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.

[Kra96] H. Krawczyk. A versatile secure key-exchange mechanism for the internet. In *Proceedings of the 1996 Symposium on Network and Distributed System Security (SNDSS '96)*, pages 114–127. IEEE Computer Society, 1996.

[KS99] J. Kelsey and B. Schneier. Authenticating secure tokens using slow memory access. In *USENIX Workshop on Smart Card Technology*, pages 101–106. USENIX Press, 1999.

[KSWH00] J. Kelsey, B. Schneier, D. Wagner, and C. Hall. Side channel cryptanalysis of product ciphers. *Journal of Computer Security*, 8(2/3), 2000.

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

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

[MSTS04] T. Moran, R. Shaltiel, and A. Ta-Shma. Non-interactive timestamping in the bounded storage model. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 460–476, 2004.

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

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