On Forward-Secure Storage*
Extended Abstract

Stefan Dziembowski

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

Abstract. We study a problem of secure data storage in a recently introduced *Limited Communication Model*. We propose a new cryptographic primitive that we call a *Forward-Secure Storage (FSS)*. This primitive is a special kind of an encryption scheme, which produces huge (5 GB, say) ciphertexts, even from small plaintexts, and has the following non-standard security property. Suppose an adversary gets access to a ciphertext $C = E(K, M)$ and he is allowed to compute any function h of C, with the restriction that $|h(C)| \ll |C|$ (say: $|h(C)| = 1$ GB). We require that $h(C)$ should give the adversary no information about M, even if he later learns K.

A practical application of this concept is as follows. Suppose a ciphertext C is stored on a machine on which an adversary can install a virus. In many cases it is completely infeasible for the virus to retrieve 1 GB of data from the infected machine. So if the adversary (at some point later) learns K, then M remains secret.

We provide a formal definition of the FSS, propose some FSS schemes, and show that FSS can be composed sequentially in a secure way. We also show connections of the FSS to the theory of compressibility of NP-instances (recently developed by Harnik and Naor).

1 Introduction

One of the main problems in the practical data security are the attacks of malware like Internet worms, Trojan horses or viruses. For an average user it is quite common that his PC gets from time to time infected by some malicious program. Once installed, such a program can take full control over the infected machine, and it may steal some confidential message M that is stored on the machine. A natural solution for this problem is to use encryption and to store only a ciphertext $C = E(K, M)$, where K is some secret key. Clearly, for the security of this method we need to assume that the key K is stored outside of

* 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. This work was partly supported by the EU project IST-3-016004-IP-09 *Sensoria* the Polish KBN grant Nr: 4 T11C 042 25.

the machine: in user's memory or on some other safe device. If the key K leaks to the adversary, then it may seem that all the security is inevitably lost, since the adversary that knows C and K can decrypt M.

In this paper we show how to limit the bad consequences of the key leakage. We will use the methods of a newly-introduced [12] *Limited Communication Model (LCM)*[1]. The main idea is as follows. Of course, if the intruder knows K at the moment *when* he got access to C then he can easily decrypt M. But what if he learned K *after* he got access to C? Clearly, if C is small then the intruder can simply retrieve C and wait until he learns K. So what if C is large (e.g. 5 GB)? In this case retrieving the entire C from the infected machine may be much harder. So, suppose that the adversary got access to the machine on which C is stored (and for the moment he has no information about K). Assume that he can perform some arbitrary computation on C and that he may retrieve the result $h(C)$. Clearly, if $|h(C)| = |C|$ then he may simply set $h(C) = C$, so let us assume that $|h(C)| \ll |C|$ (e.g. $|h(C)| = 1$ GB and $|C| = 5$ GB). Suppose that later the adversary learns K. Clearly, if E is some standard encryption scheme, then the knowledge of $h(C)$ and K allows the adversary to obtain some partial information on M.

We propose a special type of encryption schemes that guarantee that the adversary that knows $h(C)$ and K does not get any substantial information on M. We call this new primitive a *Forward-Secure Storage (FSS)*. We define its security, propose implementations, and show connections to the theory of the compressibility of NP-instances [16] that was recently developed by Harnik and Naor (see Sect. 2), the Bounded Storage Model and the Hybrid Bounded Storage Model (see Sect. 3).

Some proofs and details are omitted because of the page limit and appear in an extended version of this paper [13].

1.1 How Realistic Is Our Scenario?

We believe that FSS is an interesting notion just from the theoretical point of view. Still, it may also find some practical applications. Let us now consider the question how realistic is the assumption that the key K leaks *after* the adversary lost access to C. Here we list the cases when this can happen.

1. If the key K is stored on some safe device (a floppy-disc, say) then this device may be physically stolen. The key K may also leak to the adversary if the floppy-disc is inserted in some compromised device (this can happen if the user uses the same key to encrypt data on several machines).
2. If the key K is a human-memorized password, then the adversary can crack the password by trying all possible passwords in the dictionary. While this operation may be hard to perform in an unnoticeable way directly on the victim's machine, it is usually doable if the attacker downloads $C = E(K, M)$, and then performs the dictionary attack on his own machine. This scenario was already considered in [9] in a weaker model (see Sect. 2). Note that it is

[1] This name was proposed in [5].

unclear how to formally model the fact that the adversary *can* crack passwords on his own machine, but he *cannot* do it on the victim's machine. We discuss it is Sect. 8.

3. Even if the key K is cryptographically strong and it does not leak, the adversary may still hope that at some point in the future he can break it, when new cryptanalytic methods, or more computing power are available.

Practicality of the Assumptions Note that the current storage prices are extremely low. Today's price of one blank DVD (which has almost 5 GB of storage) is around 50 cents, and new HD-DVD and Blu-Ray technologies (which allow to store up to 50 GB on one disc) are entering the market right now. At the same time, downloading a 1 GB from an average PC connected to the Internet can still take considerable time. Also, observe that in many cases the adversary will not perform the attack if he can be traced down, and it may be quite hard to download 1 GB in an untraceable way (unlike downloading 1 KB of data, which can be posted e.g. on a Usenet newsgroup). One can also consider limiting the possible amount of retrieved data by constructing devices with artificially slow memory access (this was considered in [20]).

1.2 Our Contribution

We define the notion of the Forward-Secure Storage (Sect. 4), distinguishing between three levels of security: information-theoretic, computational, and hybrid (which is a mix of the former two). We show constructions of FSS schemes (Sect. 6). We prove that FSS is secure if composed sequentially (Sect. 7) and show how FSS can be used to protect encrypted data with a human-memorized key (Sect. 8). We show the connections with the theory of compressibility of NP-instances (Sect. 5). Note that (except of Sect. 8) we do not use the Random Oracle Assumption [2].

2 Related Work

The Work of [9] Consider the scenario in Point 2 in Sect. 1.1 (i.e. K is a human-memorized password). This was studied already in [9]. The difference between our model and theirs is that they do not allow the intruder to perform arbitrary computation on the victim's machine. The only thing that the adversary can do is to retrieve some *individual bits* $(C_{i_1}, \ldots, C_{i_s})$ of the ciphertext $C = (C_1, \ldots, C_t)$. This may seem unrealistic, since the malicious programs can easily perform computations on the victim's data. In this model they propose an efficient solution (a system called *VAST*) for the problem of secure storage. Their solutions assumes the Random Oracle Model. We will later refer to the model of [9] as a *Bounded-Retrieval Model*. This term was introduced in [7].

Intrusion Resilience of [12] The model that we consider in this paper was proposed in [12][2]. It is shown in [12] that in this model one can implement protocols for the entity authentication and the session-key generation. The security

[2] Some of the ideas of [12] were also independently discovered in [5].

of the protocols in [12] is proven in the Random Oracle Model. This assumption was later removed in [5]. We will refer to the model of [12] (and of our paper) as the *Limited Communication Model*.

The Theory of [16] One of our main inspirations is the theory of the *compressibility of NP-instances* recently developed by Harnik and Naor [16] (see also [11]). Suppose that we are given an NP-language $L \subseteq \{0,1\}^*$. We say that L is *π-compressible (to another NP-language L')* if there exists a polynomial-time *π-compressing algorithm* $\mathcal{C} : \{0,1\}^* \to \{0,1\}^*$ such that (1) $x \in L$ if and only if $\mathcal{C}(x) \in L'$, and (2) for every $x \in \{0,1\}^*$, we have $|\mathcal{C}(x)| \leq \pi(|x|)$. We say that L is *compressible (to L')* if π is polynomial in $\log |x|$ and in $w(x)$, where $w(x)$ is the maximal size of the witness for the instances of length $|x|$ (otherwise we say that it is *incompressible*).

A compressing algorithm is *witness-retrievable* if there exists a polynomial-time algorithm W such that if v is an NP-witness for $x \in L$, then $W(v)$ is an NP-witness for $\mathcal{C}(x)$.

Showing that there exists an incompressible NP-language (under the standard cryptographic assumptions) is currently an open problem. In an updated version of [16][3] the authors show, however, that if one-way functions exist then there does not exist a witness-retrievable compressing algorithm for SAT.

All-Or-Nothing Transform A scenario similar to ours was consider in [27] (and in several subsequent papers), where the notion of the *All-Or-Nothing Transform* was introduced. Namely, [27] proposes an encryption method where it is hard to get any information on M, given K and most (but not all) of the bits of $C = E(K, M)$. Note that the fundamental difference between [27] and our work is that [27] assumes that the adversary knows some *individual bits* of C (like in the Bounded Retrieval Model, described above), while in our model the adversary can compute an arbitrary function h of C (with $|h(C)| \ll |C|$). See also Sect. 9.

3 Tools

Bounded-Storage Model We will use the results from the Bounded-Storage Model (BSM), introduced by Maurer in [24]. This model was studied in the context of *information-theoretically secure* encryption [1, 15, 23, 28], key-agreement [4, 14], oblivious transfer [3, 10] and time-stamping [25]. In this model one assumes that a random string $R \in \{0,1\}^{t_{\text{BSM}}}$ (called a *randomizer*) is either temporarily available to the public or broadcast by one of the legitimate parties. The honest users, Alice and Bob share a short secret *initial key* $K \in \{0,1\}^{m_{\text{BSM}}}$, selected uniformly at random, and they apply a known *key-expansion function* $f : \{0,1\}^{m_{\text{BSM}}} \times \{0,1\}^{t_{\text{BSM}}} \to \{0,1\}^{n_{\text{BSM}}}$ to compute the *derived string* $X = f(K, R) \in \{0,1\}^{n_{\text{BSM}}}$ (usually $n_{\text{BSM}} \gg m_{\text{BSM}}$). Later X can be used, e.g., as a key for the one-time pad encryption. The function f must be efficiently

[3] Available at http://www.cs.technion.ac.il/˜harnik.

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.

We assume that the adversary $\mathcal{A}_{\mathrm{BSM}}$ (that is a computationally-unbounded Turing Machine) can compute an arbitrary function h of R, with the sole restriction that the output $U = h(R)$ of this computation has a limited size: $U \in \{0,1\}^{s_{\mathrm{BSM}}}$, where $s_{\mathrm{BSM}} \ll t_{\mathrm{BSM}}$. The adversary is allowed to store in his memory only U. After R disappears, the adversary should have essentially no information about X, even if he learns K. To define the security more formally, we consider the following game:

BSM - distinguishing game

Phase 1 R is generated randomly and passed to the adversary. The adversary can perform arbitrary computations and he can store some value $U = h(R) \in \{0,1\}^{s_{\mathrm{BSM}}}$. He is not allowed to store any other information. Then, the randomizer disappears.

Phase 2 The adversary learns K. Let $b \in \{0,1\}$ be chosen uniformly at random. Define

$$\hat{X} := \begin{cases} f(K,R) & \text{if } b = 0 \\ \text{a random element of } \{0,1\}^{n_{\mathrm{BSM}}} & \text{otherwise.} \end{cases}$$

and send $\hat{X}$ to the adversary. The adversary has to guess b.

We say that *the adversary $\mathcal{A}_{\mathrm{BSM}}$ breaks the BSM scheme f with an advantage ϵ* if his chances of guessing b correctly are $0.5+\epsilon$. To reason about the security in the asymptotic way, let us introduce a security parameter k which is an additional input of f and of the adversary. Let us assume that the parameters $m_{\mathrm{BSM}}, n_{\mathrm{BSM}}$ and t_{BSM} are functions of k. For a function $\sigma : \mathbb{N} \to \mathbb{N}$ we say that function f is *σ-secure in the bounded-storage model* if any adversary with memory at most $s = \sigma(k)$ breaks the scheme f with a negligible[4] advantage. Several key expansion functions [1, 15, 23, 28] were proven secure in the past couple of years.[5]

Observe that in some sense the assumptions in the BSM are opposite to the assumptions in the Limited Communication Model, since in the BSM we assume that transmission of large amount of data is cheaper than storing it. Nevertheless (as observed already in [12]), it turns out that the theorems proven in BSM are useful in the LCM.

***Hybrid* Bounded-Storage Model** The *Hybrid Bounded-Storage Model* [14] is defined as follows. Suppose that K is generated by a computationally-secure key-agreement protocol. Clearly, an infinitely-powerful adversary can break such a key-agreement. Therefore, assume that the computational power of the adversary is restricted (to polynomial time) until the randomizer disappears (at the end

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

[5] In these papers security of a BSM function is defined in a slightly different way. Namely, it is required that the distribution of X is statistically close to uniform from the point of view of the adversary (who knows K and U). It is easy to see that these definitions are in fact equivalent.

of Phase 1). Of course, when the adversary later gains the unlimited computing power, he can compute K (if he recorded the transcript of the key-agreement), but this should not be dangerous, since in the BSM the security holds even when the initial key K is given to the adversary (in Phase 2). In [14] it was shown that this reasoning is not correct. Namely, they show an example of a (very artificial, but computationally-secure under the standard assumptions) key agreement that cannot be safely used in this context. For more details see [14] or [17], where this model was recently formalized and generalized.

Private Information Retrieval A PIR [6, 21] scheme $(\mathcal{U}, \mathcal{D})$ is a protocol for two parties, a user $\mathcal{U}$ and a database $\mathcal{D}$, allowing the user to access database entries in a way that $\mathcal{D}$ cannot learn which information $\mathcal{U}$ requested. More precisely, the database content can be modeled as a string $x = (x_1, \ldots, x_l) \in \{0,1\}^l$, and $\mathcal{U}$ wants to access some bits $x_{i_1}, \ldots, x_{i_q}$ of x, such that $\mathcal{D}$ does not learn $I := i_1, \ldots, i_q$. It is not relevant whether $\mathcal{U}$ learns more than $x_{i_1}, \ldots, x_{i_q}$. A typical PIR protocol proceeds in three stages. First, $\mathcal{U}$ sends a query, depending on I. Let $\mathcal{Q}(I)$ denote the query for indices in I. Second, $\mathcal{D}$ computes the reply $\mathcal{R}(\mathcal{Q}(I), x)$ and sends it to $\mathcal{U}$. Third, $\mathcal{U}$ extracts $x_{i_1}, \ldots, x_{i_q}$ from $\mathcal{R}(\mathcal{Q}(I), x)$. The scheme is computationally private if no efficient distinguisher can distinguish $\mathcal{Q}(I)$ from $\mathcal{Q}(I')$, for any I and I' such that $|I| = |I'|$. To avoid trivial solutions, we require that $|\mathcal{R}(\mathcal{Q}(I), x)| < l$.

Several PIR schemes were proven secure under different intractability assumptions. For example the scheme proposed in [21] is based on the computational difficulty of the quadratic residuosity problem, and in [22] it was shown how to construct a PIR protocol from any one-way trapdoor permutation. In [8] it was shown that the assumption that PIR exists implies the existence of the Oblivious Transfer.

Symmetric Encryption A *symmetric encryption scheme* is a pair (E, D) of polynomial-time algorithms. Algorithm E takes as input a security parameter 1^k, a key $K \in \{0,1\}^{m_{\mathrm{sym}}}$, and a message $M \in \{0,1\}^{n_{\mathrm{sym}}}$ and outputs a *ciphertext* $C = E(K, M) \in \{0,1\}^{l_{\mathrm{sym}}}$ (we will assume that $m_{\mathrm{sym}}, n_{\mathrm{sym}}$ and l_{sym} are functions of k). Algorithm D takes as input 1^k, K and C and outputs M'. We require that always $D(K, E(K, M)) = M$. The security of an encryption scheme is defined as follows. Consider an adversary $\mathcal{A}_{\mathrm{enc}}$ that is a probabilistic polynomial-time machine that can specify two messages M^0 and M^1 (of the same length). Later, he receives $C = E(K, M^c)$ for a random key K and a random bit $c \in \{0,1\}$, and he has to guess c. If $\mathcal{A}_{\mathrm{enc}}$ guesses c correctly with probability $0.5 + \epsilon$ we will say that he *breaks* (E, D) *with advantage* ϵ. We say that (E, D) is *computationally secure* if any $\mathcal{A}_{\mathrm{enc}}$ breaks (E, D) with advantage that is negligible in k.

Pseudorandom Generators A *pseudorandom generator (PRG)* is a polynomial-time algorithm G that takes as input a security parameter 1^k, and a *seed* $K \in \{0,1\}^{m_{\mathrm{PRG}}(k)}$ and outputs a much longer string $G(K)$. A PRG G

is *computationally-secure* if any polynomial-time adversary is not able to distinguish $G(K)$ from a truly random string (with a non-negligible advantage). This can be formalized in a similar way as the symmetric encryption in the definition above. It was shown in [18] that a pseudorandom generator can be built from any one-way function.

Oblivious Transfer An *Oblivious Transfer (OT)* is a protocol between a Sender S (that takes as an input $(b_0, b_1) \in \{0,1\}^2$ and a security parameter 1^k) and a Receiver R (with an input $c \in \{0,1\}$ and the security parameter 1^k). After the execution of the protocol, the Receiver should know b_c (we allow a negligible probability of error). This property is called *correctness* of the protocol. Moreover: (1) the Receiver should have essentially no information on b_{1-c} (this is called *privacy of the Sender*) and (2) the Sender should have essentially no information on c (this is called *privacy of the Receiver*). In this paper we assume that the security holds in the honest-but-curious setting (i.e. even the corrupted parties follow the protocol). More formally, the security is defined as follows, let $\mathrm{OT}(c; b_0, b_1; 1^k)$ denote the execution of the OT protocol (with the inputs c, b_0, b_1 and 1^k). To define the privacy of the Sender we require that any polynomial-time Receiver should not be able to distinguish between $\mathrm{OT}(0; b, 0; 1^k)$ and $\mathrm{OT}(0; b, 1; 1^k)$, and between $\mathrm{OT}(1; 0, b; 1^k)$ and $\mathrm{OT}(1; 1, b; 1^k)$ (for $b \in \{0,1\}$). Similarly, to define the privacy of the Receiver we require that any polynomial-time Sender should not be able to distinguish (with a non-negligible advantage) between $\mathrm{OT}(0; b_0, b_1; 1^k)$ and $\mathrm{OT}(1; b_0, b_1; 1^k)$ (for $b_0, b_1 \in \{0,1\}$).

An *infinitely-often Oblivious Transfer (ioOT)* is an Oblivious-Transfer protocol for whose correctness holds only for infinitely many values $k_0, k_1, \ldots$ of the security parameter k. For all values not in $\{k_0, k_1, \ldots\}$ the Receiver instead of learning b_c gets $\perp$, except with negligible probability.

An *Oblivious-Transfer with an inefficient algorithm for the Receiver* is an OT protocol where the running time of the algorithm for the Receiver is not limited, and where the privacy of the Sender holds when the Receiver is computationally-unbounded. This notion is non-standard (we are not aware of any previous use of it) and we need it for purely theoretical purposes (in Sect 6.3). Clearly, such a protocol itself has no practical significance, as the security of the Receiver is still protected only computationally, and thus the honest Receiver is assumed to have more commutating power than the dishonest Sender.

4 Definition of the Forward-Secure Storage

The main idea of the Forward-Secure Storage is as follows. It can be viewed as a randomized symmetric encryption scheme, where the encryption algorithm Encr produces (for a given secret key K and a message $M \in \{0,1\}^{s_{\mathrm{FSS}}}$) a huge ciphertext $C = \mathrm{Encr}(K, M) \in \{0,1\}^{t_{\mathrm{FSS}}}$. One can imagine that C is stored on a machine which can be attacked by an adversary. Once the adversary gets access to the infected machine, he can perform an arbitrary computation on C, with the restriction that the output U has to be of size at most $s \ll t$. Once the adversary learned U, he looses access to C. Then, we assume that the key leaks, i.e. the adversary is given K. We require that (U, K) should not give him any significant information about M. We model it in a standard indistinguishability-style, i.e. we assume that the adversary knows in advance (i.e. before he got access to C) that the message M is a random member of a (chosen by him) two-element set $\{M^0, M^1\}$ and his task is to find out whether $M = M^0$ or $M = M^1$. This reflects the fact that the adversary may have already some a priori information about M.

Formally, a *Forward-Secure Storage (*FSS*)* scheme is a pair of polynomial-time randomized Turing Machines $\Phi = (\mathrm{Encr}, \mathrm{Decr})$. The machine Encr takes as input a *security parameter* 1^k, a *key* $K \in \{0,1\}^{m_{\mathrm{FSS}}}$ and a *plaintext* $M \in \{0,1\}^{n_{\mathrm{FSS}}}$ and outputs a *ciphertext* $C \in \{0,1\}^{t_{\mathrm{FSS}}}$. The algorithm Decr takes as input a security parameter 1^k, a key $K \in \{0,1\}^{m_{\mathrm{FSS}}}$, and a ciphertext $C \in \{0,1\}^{t_{\mathrm{FSS}}}$, and it outputs a string $M' \in \{0,1\}^{n_{\mathrm{FSS}}}$.[6] We will assume that $m_{\mathrm{FSS}}, n_{\mathrm{FSS}}$ and t_{FSS} are some functions of k. The following correctness property has to be satisfied with probability 1: $\mathrm{Decr}(1^k, K, \mathrm{Encr}(1^k, K, M)) = M$. We will sometimes drop the parameter 1^k. To define the security of an FSS scheme take a function $\sigma : \mathbb{N} \to \mathbb{N}$ and consider a *σ-adversary* $\mathcal{A}_{\mathrm{FSS}}$ (that we model as a randomized Turing Machine), that plays the following game against an oracle $\Omega^{\Phi}_{\mathrm{FSS}}(c)$ (where the challenge bit $c \in \{0,1\}$ is random), for some fixed value of the security parameter k.

FSS - distinguishing game

1. The adversary gets 1^k, produces two messages $M^0, M^1 \in \{0,1\}^{n_{\mathrm{FSS}}(k)}$ and sends them to $\Omega^{\Phi}_{\mathrm{FSS}}(c)$.
2. $\Omega^{\Phi}_{\mathrm{FSS}}(c)$ selects a random key $K \in \{0,1\}^{m_{\mathrm{FSS}}(k)}$ and computes $C = \mathrm{Encr}(1^k, K, M^c)$.
3. The adversary gets access to C and can compute an arbitrary value $U \in \{0,1\}^{\sigma(k)}$. The adversary can store U, but he is not allowed to store any other information (i.e. the entire description of the internal state of $\mathcal{A}_{\mathrm{FSS}}$ has to be included in U).
4. The adversary learns K and has to guess c.

We say that an adversary $\mathcal{A}_{\mathrm{FSS}}$ *breaks the scheme Φ with an advantage ϵ* if his probability of winning the game is $0.5 + \epsilon$. We say that an FSS scheme Φ is *σ-computationally-secure* if every probabilistic polynomial-time $\mathcal{A}_{\mathrm{FSS}}$ breaks Φ with advantage that is negligible in k. We say that an FSS scheme is *σ-information-theoretically (IT)-secure* if is secure even against an adversary that is computationally unbounded. Recall that one of the possible applications of FSS is the protection against the situation when the key K is broken because more computing power is available (see Sect. 1.1, Point 3). One can model it by assuming that the adversary gets infinite computing power, *after* he retrieved

[6] For some applications (e.g. when we expect that the ciphertext will be decrypted often) we may prefer schemes where the algorithm Decr needs to look only on a small number of the bits in C (if the plaintext is small). Actually, the schemes that we construct in this paper have this property.

U.[7] Such an adversary will be called a *hybrid adversary.* If an FSS scheme is secure against a hybrid σ-adversary, we will say that it is *σ-secure in the hybrid model*. Observe that this model is very closely related to the Hybrid BSM (see Section 3).

5 Connections with the Theory of [16]

In this section we discuss the connections of the theory of FSS to the theory of compressibility of NP-instances [16] (see Sect. 2). The lemmas that we show in this section have a cryptanalytic nature, i.e. we show implications of the form: if a certain language is compressible then a certain FSS scheme can be broken. This is because the theory of [16] concerns the worst-case complexity. To prove implications in the opposite direction, one would need to develop a theory of problems which are incompressible *on average* (this notion was recently defined in [11]). Let $\Phi = (\mathrm{Encr}, \mathrm{Decr})$ be an FSS scheme with the parameters $m_{\mathrm{FSS}}, n_{\mathrm{FSS}}$, and t_{FSS} as before. Define an NP-language $L_\Phi := \{(M, C) : \exists K.M = \mathrm{Decr}(K, C)\}$.

Lemma 1. *Suppose Φ is such that the length n_{FSS} of the encrypted message is longer than the length m_{FSS} of the key. For a function $\sigma : \mathbb{N} \to \mathbb{N}$ let π be such that $\pi(n_{\mathrm{FSS}}(k) + t_{\mathrm{FSS}}(k)) = \sigma(k)$ (for every $k \in \mathbb{N}$). We have the following:*

1. *if L_Φ is π-compressible then Φ can be broken by a hybrid σ-adversary, and*
2. *if L_Φ is π-compressible with witness retrievability, then Φ can be broken by a computational σ-adversary.*

Proof. Let us first consider Point 1. Let $\mathcal{C}$ be the algorithm that compresses L_Φ to some language L'. We construct an adversary $\mathcal{A}_{\mathrm{FSS}}$ that breaks Φ. First, the adversary $\mathcal{A}_{\mathrm{FSS}}$ produces two random messages M^0 and M^1. When he gets access to C (which is equal to $\mathrm{Encr}(K, M^c)$, for random K and c), he retrieves $U = \mathcal{C}(M^0, C)$ (clearly $|U| \leq \sigma(|C|)$). Later, when he gets infinite computing power, he checks if $(M^0, C) \in L_\Phi$ (by the properties of the compression algorithm, he can do it by checking if $U \in L'$). If $(M^0, C) \in L_\Phi$ then the adversary guesses "$c = 0$", otherwise he guesses "$c = 1$". Clearly, if $c = 0$ (which happens with probability 0.5) then his guess is correct with probability 1. If $c = 1$ then his guess may be incorrect (i.e. he may guess "$c = 0$"). But this happens only if there exist K' such that $M^0 = \mathrm{Decr}(K', C)$ (for $C = \mathrm{Encr}(K, M^1)$). Clearly the number of messages M^0 such that $M^0 = \mathrm{Decr}(K', C)$ is at most equal to the number $2^{m_{\mathrm{FSS}}}$ of the keys K'. Thus (since the total number of messages is $2^{n_{\mathrm{FSS}}} \geq 2 \cdot 2^{m_{\mathrm{FSS}}}$), the probability that for a random M^0 there exists K' such that $M^0 = \mathrm{Decr}(K', C)$ is at most 0.5. Therefore, the total probability that $\mathcal{A}_{\mathrm{FSS}}$ guesses c correctly is at least $0.5 \cdot (1 + 0.5) = 0.75$.

[7] Note that in this case it would make sense to consider a slightly weaker model, where the adversary does not receive K in Step 4 (and has to recover it using his unlimited computing power). For simplicity we assume that the adversary receives K anyway.

Essentially the same adversary can be used for the proof of Point 2. The only difference is that in order the determine if $U \in L'$ the (computationally-bounded) adversary obtains K and checks if $W(K)$ is a witness for U (where W is the algorithm for retrieving the witness). □

Lemma 2. *Let Φ be an FSS scheme whose length n_{FSS} of the ciphertext is polynomial in the security parameter. Moreover, suppose that the output of the algorithm* Decr *depends only on a polynomial number of the bits of the ciphertext. If SAT is compressible, then Φ can be broken by a hybrid σ-adversary, where σ is a polynomial.*

Clearly, the statement of the lemma has any practical meaning only if we can show candidates for FSS schemes whose length t_{FSS} of the ciphertext is superpolynomial in k (and the other parameters are polynomial in k). Such a candidate can, e.g., be the scheme Φ^{G}_{c2} that we will construct Sect. 6.3. The scheme Φ^{G}_{c2} will be constructed from a BSM-secure function f, and the length of the ciphertext will be equal to $t_{\mathrm{BSM}} + n_{\mathrm{BSM}}$ (where t_{BSM} is the length of the randomizer, and n_{BSM} is the length of the plaintext). Therefore, it is enough to show a BSM-secure function with a superpolynomial length of the randomizer. An example of such a function is the function of [15] (for an appropriate choice of the parameters). Observe that Lemma 2 implies that if Φ^{G}_{c2} is secure against an adversary with memory of a polynomial size (assuming that one-way functions exits, say), then SAT is incompressible.

Proof (of Lemma 2, sketch). We will show that if SAT is compressible, then L_{Φ} is compressible. By Lemma 1 this will suffice. It is enough to show a polynomial-time algorithm, that given an input (M, C) produces a Boolean formula $\phi_{M,C}(x_1, \ldots, x_q)$ (where $x_1, \ldots, x_q$ are free variables) such that: (1) $\phi_{M,C}$ is satisfiable if and only if $(M, C) \in L_{\Phi}$, and (2) the number q of the free variables is polynomial in K and $\log |(M, C)|$.[8] Such a formula can be constructed in essentially the same way as the formula in the proof of Lemma 2.14 of [16]. The details appear in [13]. □

Algorithm Encr **Has to be Randomized** Further exploring the analogies between the theory of [16] and FSS, we show how the method developed in [16] can be used to prove that the algorithm Encr has to be randomized. This in particular means that algorithm Encr cannot use part of its own key K as a seed for a PRG and obtain the necessary randomness this way. Intuitively, it is quite obvious. Here we give a formal proof.

Observation 1 *Any FSS scheme with deterministic encryption can be broken by a computational σ-adversary where $\sigma(k) = k$.*

Proof (sketch). This method is taken from [16] (Claim 2.3). The adversary selects a random hash function $H : \{0,1\}^{t_{\mathrm{FSS}}(k)} \to \{0,1\}^k$ from a family of pairwise-independent hash functions. Then, he chooses two arbitrary distinct messages

[8] In [16] this is called a *W-reduction.*

M^0 and M^1. When he obtains $C = \mathrm{Encr}(K, M^i)$ he computes $U = H(C)$ and retrieves it. Later (when he receives K), he can test if $i = 0$ or $i = 1$ by finding i for which

$$H(\mathrm{Encr}(K, M^i) = U). \tag{1}$$

By the pairwise independence of H the probability that 1 holds for both $i = 0$ and $i = 1$ is equal to 2^{-k}, and hence it is negligible. □

Clearly, the statement of the lemma holds also for the FSS schemes that are allowed to have a random input r of a logarithmic length (i.e. $|r| = \log k$). This is because in this case a polynomial-time adversary can simply test if (1) holds by examining all possible random inputs $r \in \{0,1\}^{\log(k)}$.

6 Construction of FSS Schemes

In this section we construct an IT-secure FSS scheme Φ_{IT} (Sect. 6.2) and an FSS scheme Φ_{c1} (Sect. 6.1), that is computationally secure, assuming that one-way functions exist. The advantage of Φ_{c1} is that it allows to encrypt messages that are much longer than the key (which, by Shannon Theorem, cannot be the case if the IT security is required).

The main drawback of the scheme Φ_{c1} is that it can be trivially broken in the hybrid model. A natural question to ask is whether there exists an FSS scheme that is secure in the hybrid model. We propose a construction of an FSS scheme Φ^G_{c2} (from a pseudorandom generator G) about which we conjecture that it is secure in the hybrid model. We are able to prove neither the hybrid nor even the computational security of Φ^G_{c2} from the assumption that G is a computationally-secure PRG. We argue about the security of Φ^G_{c2} in a non-standard way (see Sect. 6.3). Our results are summarized in the following table:

scheme	IT-security	hybrid security	computational security
Φ_{IT}	secure	secure	secure
Φ^G_{c2}	not secure	conjectured secure	conjectured secure
Φ_{c1}	not secure	not secure	secure if one-way functions exist

In our constructions we will use a function f that is σ-secure in the BSM. Let us now fix such a function (one could take e.g. the function of [15], or any other function that was proven secure in the BSM), and let $m_{\mathrm{BSM}}, n_{\mathrm{BSM}}$ and t_{BSM} be the parameters defined in Sect. 3.

6.1 Computationally-Secure FSS

Let (E, D) be a symmetric encryption scheme with the parameters $m_{\mathrm{sym}}, n_{\mathrm{sym}}$ and l_{sym} defined as in Sect. 3. In this section we construct a computationally-secure FSS scheme $\Phi_{\mathrm{c1}} = (\mathrm{Encr}_{\mathrm{c1}}, \mathrm{Decr}_{\mathrm{c1}})$. Fix some security parameter k. The key of Φ_{c1} is interpreted as a pair (K, W), where $K \in \{0,1\}^{m_{\mathrm{BSM}}}$ is the initial key of f and $W \in \{0,1\}^{n_{\mathrm{BSM}}}$. Hence, the length m_{c1} of the key of Φ_{c1} is equal to $m_{\mathrm{BSM}} + n_{\mathrm{BSM}}$. We also assume that the length n_{BSM} of the string derived by the BSM function f is equal to the length of the key for the encryption scheme (E, D). The length n_{c1} of the plaintext is equal to the length n_{sym} of the plaintext of (E, D). The length t_{c1} of the ciphertext is equal to $t_{\text{BSM}} + l_{\text{sym}}$.

$\boxed{\mathrm{Encr}_{\text{c1}}((K, W), M)}$

1. Generate a random string $R \in \{0,1\}^{t_{\text{BSM}}}$.
2. Let $K' := f(K, R) \oplus W$.
3. Output $C = (R, E(K', M))$.

$\boxed{\mathrm{Decr}_{\text{c1}}((K, W), C)}$

1. Parse C as (R, X).
2. Let $K' := f(K, R) \oplus W$.
3. Output $D(K', X)$.

Lemma 3. *If f is σ-secure in the BSM and (E, D) is a computationally-secure symmetric encryption scheme, then Φ_{c1} is a σ-computationally-secure FSS scheme.*

Proof. Let $\mathcal{A}_{\text{comp}}$ be an adversary that for a security parameter k breaks Φ_{c1} with an advantage $\epsilon = \epsilon(k)$. We are going to construct adversaries: $\mathcal{A}_{\text{BSM}}$ and $\mathcal{A}_{\text{enc}}$, such that either $\mathcal{A}_{\text{BSM}}$ breaks the BSM scheme f (using memory of size $\sigma(k)$), or $\mathcal{A}_{\text{enc}}$ breaks the encryption scheme (E, D). The running time of these adversaries will be polynomial in the running time of $\mathcal{A}_{\text{comp}}$. Let ϵ_{BSM} denote the advantage of $\mathcal{A}_{\text{BSM}}$ in breaking f, and let ϵ_{enc} denote the advantage of $\mathcal{A}_{\text{enc}}$ in breaking (E, D). The adversary $\mathcal{A}_{\text{BSM}}$ simulates the adversary $\mathcal{A}_{\text{comp}}$ as follows.

$\boxed{\mathcal{A}_{\text{BSM}}}$

1. Start $\mathcal{A}_{\text{comp}}$ and pass the security parameter 1^k to $\mathcal{A}_{\text{comp}}$. Obtain from $\mathcal{A}_{\text{comp}}$ the messages M^0 and M^1.
2. During Phase 1 of the distinguishing game (see Sect. 3) do the following. Input the randomizer $R \in \{0,1\}^{t_{\text{BSM}}}$. Select a random bit $c \in \{0,1\}$ and a random string $K' \in \{0,1\}^{n_{\text{BSM}}}$. Send $(R, E(K', M^c))$ to $\mathcal{A}_{\text{comp}}$. Store in your memory the value $U \in \{0,1\}^{\sigma(k)}$ that $\mathcal{A}_{\text{comp}}$ stores.
3. In Phase 2 receive K and $\hat{X}$, compute $W = \hat{X} \oplus K'$ and pass (K, W) to $\mathcal{A}_{\text{comp}}$. If $\mathcal{A}_{\text{comp}}$ guesses c correctly, then output "$b = 0$" (i.e. guess that $\hat{X} = f(K, R)$). Otherwise output "$b = 1$" (i.e. guess that $\hat{X}$ is random).

The adversary $\mathcal{A}_{\text{enc}}$ simulates the adversary $\mathcal{A}_{\text{comp}}$ in the following way.

$\boxed{\mathcal{A}_{\text{enc}}}$

1. Start $\mathcal{A}_{\text{comp}}$ and pass the security parameter 1^k to $\mathcal{A}_{\text{comp}}$. Obtain from $\mathcal{A}_{\text{comp}}$ the messages M^0 and M^1. Output them.
2. Select randomly $R \in \{0,1\}^{t_{\text{BSM}}}, K \in \{0,1\}^{m_{\text{BSM}}}$ and $W \in \{0,1\}^{n_{\text{BSM}}}$. When you receive C (recall that $C = E(K', M^c)$, for a random K' and c, and the goal of $\mathcal{A}_{\text{enc}}$ is to guess c), pass (R, C) to $\mathcal{A}_{\text{comp}}$.
3. Pass (K, W) to $\mathcal{A}_{\text{comp}}$. Output the bit c that $\mathcal{A}_{\text{comp}}$ outputs.

Let us now look at the adversary $\mathcal{A}_{\text{BSM}}$ defined above. If $b = 0$ then $\hat{X} = f(K, R)$, and hence $K' = f(K, R) \oplus W$. Therefore in this case $\mathcal{A}_{\text{BSM}}$ simply simulated the normal execution of $\mathcal{A}_{\text{comp}}$ against the FSS scheme Φ^f_{c1} with the challenge bit c. Thus, in this case $\mathcal{A}_{\text{comp}}$ guesses c correctly with probability $0.5 + \epsilon$.

If $b = 1$ then K' is independent of the variable (R, K, W). Hence, $\mathcal{A}_{\mathrm{BSM}}$ simulated $\mathcal{A}_{\mathrm{comp}}$ in exactly the same way as $\mathcal{A}_{\mathrm{enc}}$. Thus, the probability that $\mathcal{A}_{\mathrm{comp}}$ guesses c correctly is $0.5 + \epsilon_{\mathrm{enc}}$. Therefore, the probability $0.5 + \epsilon_{\mathrm{BSM}}$, that $\mathcal{A}_{\mathrm{BSM}}$ guesses b correctly is equal to

$$\begin{aligned}&\tfrac{1}{2} \cdot P\left(\mathcal{A}_{\mathrm{BSM}} \text{ outputs } \text{``}b = 0\text{''} | b = 0\right) + \tfrac{1}{2} \cdot P\left(\mathcal{A}_{\mathrm{BSM}} \text{ outputs } \text{``}b = 1\text{''} | b = 1\right) \\ &= (0.5 + \epsilon)/2 + (0.5 - \epsilon_{\mathrm{enc}})/2 = (1 + \epsilon - \epsilon_{\mathrm{enc}})/2\end{aligned}$$

Therefore, we get that $\epsilon = \epsilon_{\mathrm{BSM}} + \epsilon_{\mathrm{enc}}$. Thus, if ϵ is non-negligible, then one of ϵ_{BSM} and ϵ_{enc} has to be non-negligible. Therefore, if Φ_{c1} is not secure, then either f of (E, D) is not secure either. □

Since the existence of one-way functions implies the existence of symmetric encryption schemes [18] (where the length of the plaintext is an arbitrary polynomial of the security parameter), we get the following.

Theorem 1. *If one way-functions exist, then there exists a σ-computationally-secure FSS scheme with parameters σ, m_{c1} and t_{c1} as above, where the length n_{c1} of the encrypted message is an arbitrary polynomial of the security parameter.*

6.2 IT-Secure FSS

The IT-secure FSS scheme Φ_{IT} can be constructed by substituting (in the construction from Sect. 6.1) the computationally-secure encryption scheme (E, D) with the one-time pad encryption. Clearly, in this case the length n_{IT} of the plaintext is equal to n_{BSM}, the length m_{IT} of the key is equal to $m_{\mathrm{BSM}} + n_{\mathrm{BSM}}$ (and hence it is greater than the length of the plaintext) and the length t_{FSS} of the ciphertext is equal to $t_{\mathrm{BSM}} + n_{\mathrm{BSM}}$. The security of this construction can be proven by essentially repeating the proof of Lemma 3. Therefore, get the following.

Theorem 2. *There exists a σ-IT-secure FSS scheme with the parameters σ, $m_{\mathrm{IT}}, n_{\mathrm{IT}}$ and t_{IT} as above.*

6.3 FSS Scheme With a Conjectured Hybrid Security

It is easy to see that the scheme Φ_{c1} can be broken in the hybrid model: the adversary can simply retrieve the second component $(E(K', M))$ of the ciphertext, and then break it by the exhaustive key-search. In this section we show the construction of an FSS scheme $\Phi^G_{\mathrm{c2}} = (\mathrm{Encr}_{\mathrm{c2}}, \mathrm{Decr}_{\mathrm{c2}})$, about which we conjecture that it is secure in the hybrid model. The components for our construction are: a pseudorandom generator G (see Sect. 3), and the σ-IT-secure scheme from Sect. 6.2. The problem with this construction is that we are not able to base (even the computational) security of Φ^G_{c2} on any standard assumption. We argue about the security of Φ^G_{c2} in the following way. Informally speaking, we show that if we are given a computational adversary that breaks Φ^G_{c2} (for some PRG G) then one can construct an ioOT protocol (from this G). Although this does not guarantee that Φ_{c2}^{G} is secure for any choice of G, it is an indication that for a Φ_{c2}^{G} is secure if G is some "standard" PRG (see discussion at the end of this section). In case of the hybrid security we have a weaker security argument: we show that if one can break Φ_{c2}^{G} in the computational model, then one can construct from G an ioOT protocol with an inefficient algorithm for the Receiver (see Sect. 3).

Fix some security parameter k. Let G be a PRG with some short seed of length m_{PRG}. The length m_{c2} of the key of Φ_{c2}^{G} is equal to m_{PRG}. The length n_{c2} of the encrypted message is equal to n_{IT} and the length t_{c2} of the ciphertext is equal to t_{IT}. The $\mathrm{Encr}_{\mathrm{c2}}$ and $\mathrm{Decr}_{\mathrm{c2}}$ (for a key $K \in \{0,1\}^{m_{\mathrm{c2}}}$, and a plaintext $M \in \{0,1\}^{n_{\mathrm{c2}}}$) are defined as: $\mathrm{Encr}_{\mathrm{c2}}(K,M) := \mathrm{Encr}_{\mathrm{IT}}(G(K),M)$ and $\mathrm{Decr}_{\mathrm{c2}}(K,C) := \mathrm{Decr}_{\mathrm{IT}}(G(K),C)$. Clearly, $\mathrm{Decr}_{\mathrm{c2}}(K,\mathrm{Encr}_{\mathrm{c2}}(K,M)) = M$, so it remains to argue about the security.

Computational Security of Φ_{c2}^{G} We start with considering the computational security of Φ_{c2}^{G}. Suppose a polynomial-time σ-adversary $\mathcal{A}_{\mathrm{comp}}$ breaks $(\mathrm{Encr}_{\mathrm{c2}}, \mathrm{Decr}_{\mathrm{c2}})$ with a non-negligible advantage $\epsilon(k)$ and G is a pseudorandom generator. We are going to construct an ioOT protocol (with an unconditional privacy of the Sender and a computational privacy of the Receiver). Before going to the construction let us present the intuition behind it. In [16] (Theorem 3.1) it is shown that if there exists a witness-retrievable compression algorithm for SAT then the existence of one-way functions implies existence of Oblivious Transfer. This is proven by (1) constructing a PIR protocol from a compression algorithm for SAT and then (2) using the result of [8] that PIR implies Oblivious Transfer. We proceed in a similar way, in some sense combining the proofs (1) and (2) into one. This is possible since, as we remarked before, an adversary that breaks an FSS scheme can be viewed as an compression algorithm (see Sect. 5). Note that in our proof we do not construct PIR from $\mathcal{A}_{\mathrm{comp}}$ (which may be quite hard in general). Note also that our method has some similarities with the one used recently in [11].

The details of the construction are as follows. We are going to construct an Oblivious Transfer protocol OT whose correctness holds only with probability non-negligibly greater than 0.5 (i.e. the probability that the Receiver outputs b_c after the execution of $\mathrm{OT}(b; c_0, c_1; 1^k)$ will be non-negligibly greater than 0.5, for any choice of c_0, c_1 and b). This suffices to construct an ioOT protocol, since by repeating the protocol polynomial number of times, the parties can reduce the error probability to negligible, for infinitely many values of the security parameter (the details can be found in [13]).

The main idea is that the Sender and the Receiver will simultaneously simulate the execution of $\mathcal{A}_{\mathrm{comp}}$, with the Sender holding the ciphertext C and the Receiver receiving just U. The protocol goes as follows.

$\boxed{\mathrm{OT}(c; b_0, b_1; 1^k)}$

1. The Receiver selects a random seed $K \in \{0,1\}^{m_{\mathrm{PRG}}}$. Let $K_c := G(K)$ and let $K_{1-c} \in \{0,1\}^{m_{\mathrm{FSS}}}$ be random. The Receiver sends (K_0, K_1) to the Sender.
2. The Sender starts the adversary $\mathcal{A}_{\mathrm{comp}}$ and passes the security parameter 1^k to him. Let M^0 and M^1 be the messages produced by $\mathcal{A}_{\mathrm{comp}}$.

The Sender selects random bits $d, x \in \{0,1\}$ and computes $C = \mathrm{Encr}_{\mathrm{c2}}(K_d, M^{b_d \oplus x})$).[9] He passes this value to (his simulated copy of) $\mathcal{A}_{\mathrm{comp}}$. Let U be the value that $\mathcal{A}_{\mathrm{comp}}$ stores in his memory.
The Sender sends (U, d, x) and the random input of $\mathcal{A}_{\mathrm{comp}}$ to the Receiver.

3. If $d \neq c$ then the Receiver outputs a random bit.[10] Otherwise the Receiver simulates the execution of $\mathcal{A}_{\mathrm{comp}}$ with the random input that he got from the Sender. The Receiver sends U and K to (his copy of) the adversary $\mathcal{A}_{\mathrm{comp}}$. Let B be the output of $\mathcal{A}_{\mathrm{comp}}$. The Receiver outputs $B \oplus x$.

Lemma 4. *The protocol* OT *protects the privacy of the Receiver.*

Proof (sketch). This is quite straightforward, as the Sender that distinguishes c from $1-c$ should also be able to distinguish a uniformly random string from a string produced by a PRG G. More formally, one can construct an adversary that breaks the PRG G by simulating the algorithm that distinguishes $\mathrm{OT}(c; b_0, b_1, 1^k)$ from $\mathrm{OT}(1-c; b_0, b_1; 1^k)$. The details appear in [13]. □

Lemma 5. *The protocol* OT *protects the privacy of the Sender.*

Proof (sketch). We need to show that the Receiver has negligible advantage in guessing b_{1-c}. Clearly, the only value from which the Receiver could learn anything about b_{1-c} is U. If $d = c$ then this value is independent of b_{1-c}, so we are done. Otherwise (when $d \neq c$), U is a function of the ciphertext $C = \mathrm{Encr}_{\mathrm{IT}}(K_{1-c}, M^{b_d \oplus x})$ (where K_{1-c} is uniformly random). To get some non-negligible information about b_{1-c}, the Receiver would need to distinguish with non-negligible advantage between M^0 and M^1 (knowing U and K_{1-c}). This, however, would contradict the security of Φ_{IT}. □

It is easy to see that the privacy of the Sender is actually protected unconditionally.

Lemma 6. *The* OT *protocol is an Oblivious Transfer protocol whose correctness holds with probability non-negligibly greater than* 0.5.

Proof. The privacy was proven in Lemmas 4 and 5. It suffices to show the correctness. Fix some security parameter 1^k. Let $\epsilon = \epsilon(k)$ be the advantage of $\mathcal{A}_{\mathrm{comp}}$. If $d \neq c$ (which happens with probability 0.5) the probability that R outputs a correct value is clearly 0.5.

Otherwise (if $d = c$) observe that the Receiver and the Sender simply simulated the execution of $\mathcal{A}_{\mathrm{comp}}$ against an oracle $\Omega_{\mathrm{FSS}}^{\Phi^G_{\mathrm{c2}}}(b_d \oplus x)$ (see Sect. 4), where $b_d \oplus x \in \{0,1\}$ is random. Therefore, the probability that $\mathcal{A}_{\mathrm{comp}}$ (and hence the Receiver) outputs $B = b_c$ is $0.5 + \epsilon(k)$. Thus, the total probability that $\mathcal{A}_{\mathrm{comp}}$ outputs b_c is equal to $\frac{1}{2} \cdot \frac{1}{2} + \frac{0.5+\epsilon}{2} = \frac{1}{2} + \frac{\epsilon}{2}$, which is clearly non-negligible. □

[9] The role of x is to guarantee that $M^{b_d \oplus x}$ is chosen *uniformly* at random from the set $\{M^0, M^1\}$, what is needed in the proof of Lemma 6.

[10] This is because in this case the Receiver has no significant information about b_c. Alternatively, we could instruct him to output some default value $\perp$.

Hybrid Security of Φ_{c2}^{G} One can essentially repeat the same construction for the hybrid adversary $\mathcal{A}_{\rm hybr}$ (instead of the computational adversary $\mathcal{A}_{\rm comp}$). The only difference is that, since we allow the adversary $\mathcal{A}_{\rm hybr}$ (that attacks Φ_{c2}^{G}) to have infinite computing power (in the final stage), then we need to assume that also the Receiver has infinite computing power (as otherwise he would not be able to simulate $\mathcal{A}_{\rm hybr}$ in Step 3). Thus, the resulting protocol has an inefficient algorithm for the Receiver (see Sect. 3).

Discussion Let us start with discussing the meaning of the construction of OT from a computational adversary $\mathcal{A}_{\rm comp}$. First, recall that constructing an OT protocol from a PRG (or, equivalently, from a one-way function) is a major open problem in cryptography. It was shown in [19] that such a construction cannot be done in black-box way (observe that our construction is non-black-box, as the adversary $\mathcal{A}_{\rm comp}$ is allowed to use the internal structure of G) and even constructing an ioOT from a one-way function would be a breakthrough.

A natural question is: which pseudorandom generators G are safe for the use in the construction of $\mathcal{A}_{\rm hybr}$. A highly informal answer is: "those whose security does not imply Oblivious Transfer". We leave formalizing this property as an open problem. Note that in principle there can exist (probably very artificial) pseudorandom generators which cannot be safely used in Φ_{c2}. In the next section we present an example that shows that something like this can happen if one replaces a PRG in Φ_{c2}^{G} with a slightly more general primitive.

Observe that one can also view our result as a win-win situation: either we win because we have FSS or we win because we have an ioOT from a one-way function. (This is how a result of a similar nature is interpreted in [11].) Interestingly, a similar reasoning was recently used also in [26] (using the language of [26] we could say that Φ_{c2}^{G} is *secure in Minicrypt*).

The argument for the security of Φ_{c2}^{G} in the hybrid model is much weaker, since we used a non-standard version on the OT protocol. Observe that (as argued in Sect. 5) showing the security of Φ_{c2}^{G} implies that SAT is incompressible (which may indicate that such a proof is difficult to find). In general, since there are no known techniques of basing incompressibility of NP-instances on standard assumptions, we conjecture that basing hybrid security of FSS schemes on such assumptions may also be hard.

Intuitively, the scheme Φ_{c2}^{G} should be secure because a computationally-limited adversary that has no (computational) information on $G(K)$ (when he has access to R) should not be stronger than a computationally-unlimited adversary that attacks the scheme $\Phi_{\rm IT}$ (where the only difference is that instead of $G(K)$ we use a truly random key). This is a similar reasoning to the one that was used to argue about the security of the Hybrid BSM (see Sect. 3). In the next section we argue that in certain cases this intuition may be misleading (by showing an example that is similar to the one shown for the Hybrid BSM in [14]).

A Scheme $\Phi_{c2}^{G,\alpha}$ A natural question to ask is whether there exist pseudorandom generators that cannot be safely used in Φ_{c2}^{G}. We leave this as an open problem.

What we can show, however, is that there exists a modification of the scheme Φ^G_{c2} (denote it $\Phi^{G,\alpha}_{c2} = (\mathrm{Encr}^{G,\alpha}_{c2}, \mathrm{Decr}^{G,\alpha}_{c2})$, where α is defined below) that has similar properties to Encr_{c2} (i.e. if one can break it, then one can construct an ioOT protocol), but for some particular choice of G and α it can be broken. This example shows that basing the security of Φ^G_{c2} on some standard assumptions about G may be hard. Because of the lack of space we present just a sketch of the construction. We concentrate here on the computational security; however, the same example works for the hybrid security. In the construction of α we will use a PIR scheme $(\mathcal{U}, \mathcal{D})$ (see Sect. 3). The nature of this construction is similar to the one of [14] (see also [17]).

Let $\Phi_{\mathrm{IT}} = (\mathrm{Encr}_{\mathrm{IT}}, \mathrm{Decr}_{\mathrm{IT}})$ be the $\sigma(k)$-IT-secure scheme from Sect. 6.2. The key of the scheme $\Phi^{G,\alpha}_{c2}$ has the form $K' = (K, \rho)$, where K is a seed for a PRG G, and ρ is a random input of the PIR user $\mathcal{U}$. Define Φ'_{c2} as $\mathrm{Encr}^{G,\alpha}_{c2}((K,\rho), M) := (\mathrm{Encr}_{\mathrm{IT}}(G(K), M), \alpha(K,\rho))$, where α is defined below. Recall that the first component of $\mathrm{Encr}_{\mathrm{IT}}(G(K), M)$ is a randomizer R, and the knowledge of a small number of bits of R (namely those that are used to compute the value of f) suffices to decrypt M (if the key $G(K)$ and the rest of the ciphertext are known). Let $I := i_1, \ldots, i_q$ be the indices of those bits. We can assume that q is much smaller than $\sigma(k)$.

Suppose that we run the user $\mathcal{U}$ (with the input I and the random input ρ). Let $\mathcal{Q}(I)$ be the query that $\mathcal{U}$ generates. We set $\alpha(K') := \mathcal{Q}(I)$. Clearly, (by the security of PIR) the knowledge of $\alpha(K')$ does not help a computationally-bounded adversary in distinguishing $G(K)$ from a random string. Hence, if there exists a computational σ-adversary $\mathcal{A}_{\mathrm{comp}}$ that breaks $\Phi^{G,\alpha}_{c2}$, then one can construct an OT protocol (in essentially the same way as we constructed the OT protocol).

Now, a computational σ-adversary can perform the following attack. He treats R as a database and retrieves the reply $U = \mathcal{R}(\mathcal{Q}(I), R)$ (by the properties of PIR we can assume that the length of this reply is much shorter than the length of the ciphertext). Later, when the adversary learns ρ, he can simulate the user (because K' includes the random input of the user), and use $\mathcal{U} = \mathcal{R}(\mathcal{Q}(I), x)$ to compute $R_{i_1}, \ldots, R_{i_q}$. Hence, the adversary can compute M.

7 Sequential Composability

Here we argue that FSS is *sequentially composable* in the following sense. Take some σ-computationally-secure scheme FSS $\Phi = (\mathrm{Encr}, \mathrm{Decr})$, and let $s = \sigma(k)$ (for some fixed k). Consider the following procedure. Suppose we start with a key K_0 (that is stored on some machine $\mathcal{M}_0$) and we encrypt (using FSS) some other key, K_1, with K_0 (and we store the ciphertext $C_1 = \mathrm{Encr}(K_0, K_1)$ on a machine $\mathcal{M}_1$, say). Then, we generate another key K_2 and encrypt if with K_1 (we store the ciphertext $C_2 = \mathrm{Encr}(K_1, K_2)$ on another machine $\mathcal{M}_2$), and so on (a ciphertext C_i is stored on a machine $\mathcal{M}_i$). At the end we encrypt some message M with the key K_{w-1} (let $C_w = \mathrm{Encr}(K_w, M)$ be the ciphertext). We show that this procedure is safe in the following sense. Suppose an adversary can break in, and take a temporary control over each of these machines (to simplify the model, suppose that the adversary can control one machine at a time, and he cannot take control over the same machine twice). Once the adversary broke into a machine $\mathcal{M}_i$, he can perform any computation on C_i and retrieve some value $U_i \in \{0,1\}^s$. Let $(\mathcal{M}_{i_1}, \ldots, \mathcal{M}_{i_{w+1}})$ be the order in which the adversary was breaking into the machines. We assume that the adversary is allowed to break into the machines in an arbitrary order, except that the order $(\mathcal{M}_0, \ldots, \mathcal{M}_w)$ is not allowed. We say that such a scheme is computationally secure if such an adversary has negligible advantage in determining whether $M = M^0$ or $M = M^1$ (for some chosen by him messages M^0 and M^1). It can be shown that this scheme is computationally secure, assuming that the scheme Φ is computationally secure. (Observe that if $w = 1$ then the security of this scheme is equivalent to the security of Φ.) The same holds for the IT and the hybrid security. Because of the lack of space, the formal definition of the sequential composition and the security proofs are given in [13].

8 Human-Memorized Passwords

In this section we show how FSS can be used if the data is protected by a password that is memorized by a human (see Sect. 1.1, Point 2). The main problem here is how to model the fact that the adversary is not able to perform a dictionary attack on the machine of the victim, but he can do it on his own machine. Note that when defining FSS we did not differentiate between the computational power of the adversary in different stages of the attack (except of the hybrid model where we simply assumed that at a certain moment the adversary gets an unlimited computing power). We will make use of a Random Oracle Assumption [2]. Such an oracle Ω^H can be viewed as a black-box which contains a function H chosen uniformly at random from a set of functions of a type $\{0,1\}^p \to \{0,1\}^{m_{\mathrm{FSS}}}$. Normally one assumes that every party in the protocol (including the adversary) has access to Ω^H. Here (to model the fact that the adversary cannot perform the dictionary attack on the victim's machine) we assume that the adversary can get access to Ω^H only *after* he lost access to C.

We construct an FSS scheme $(\mathrm{Encr}_{pass}, \mathrm{Decr}_{pass})$ that is secure if the secret key is a human-memorized password. Let $(\mathrm{Encr}, \mathrm{Decr})$ be an computationally secure FSS scheme with a key of length m_{FSS}. Let $\pi \in \{0,1\}^p$ be a password of the user. Define $\mathrm{Encr}_{pass}(\pi, M) := \mathrm{Encr}(H(\pi), M)$ and $\mathrm{Decr}_{pass}(\pi, M) := \mathrm{Decr}(H(\pi), M)$. (Observe that in this case our protocol has actually many similarities with the protocol of [9].) One can easily prove that if $(\mathrm{Encr}, \mathrm{Decr})$ is σ-computationally-secure then also $(\mathrm{Encr}_{pass}, \mathrm{Decr}_{pass})$ is secure against an adversary that can retrieve $\sigma(k)$ bits (in the model described above).

9 Open Problems

An interesting open problem is to determine if all computationally-secure pseudorandom generators G can be safely used in the construction of $\Phi^G_{\mathrm{c}2}$. If it turns out that the answer is "no" (e.g. there exists examples similar to those that exists for the scheme $\Phi_{c2}^{G,\alpha}$), then another open problem is to propose new definitions of computational indistinguishability, that would capture the fact that G can be safely used in the scheme Φ_{c2}. Clearly, this questions may have different answers for the computational and for the hybrid security.

The security argument for the hybrid security of Φ_{c2}^{G} is rather weak (since we use the non-standard notion of OT with inefficient algorithm for the Receiver). A natural open problem is to strengthen this argument, or to find alternative constructions.

It would also be interesting to examine the possibility of using the theory of All-or-Nothing transforms. Recall (see also Sect. 3) that the main difference between our model and the model considered in that theory is that they do not allow the adversary to perform arbitrary computations on the ciphertext. It may well be, however, that some of their constructions are actually secure also in our model (at least in the computational case). It is an open problem to prove it (if proving it using the standard assumptions is hard, then maybe one could at least use an argument similar to the one used in Sect. 6.3).

Note also that we did not provide any concrete examples of the parameters in our schemes. Doing it remains an open task.

Acknowledgments I would like to thank Danny Harnik for pointing to me an error in the previous version of this paper, and suggesting to me how to repair it. I am also grateful to Krzysztof Pietrzak and Bartosz Przydatek for helpful discussions, and to the anonymous CRYPTO reviewers, for lots of useful comments.

References

1. 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.
2. 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.
3. 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.
4. C. Cachin and U. Maurer. Unconditional security against memory-bounded adversaries. In *CRYPTO'97*, volume 1294 of *LNCS*, pages 292–306. Springer, 1997.
5. D. Cash, Y. Z. Ding, Y. Dodis, W. Lee, R. Lipton, and S. Walfish. Intrusion-resilient authentication in the Limited Communication Model. Cryptology ePrint Archive, Report 2005/409, 2005. http://eprint.iacr.org/.
6. B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private Information Retrieval. *Journal of the ACM*, 45(6):965–981, 1998.
7. G. Di Crescenzo, R. J. Lipton, and S. Walfish. Perfectly secure password protocols in the Bounded Retrieval Model. In *Theory of Cryptography Conference'06*, volume 3876 of *LNCS*, pages 225–244. Springer, 2006.

8. G. Di Crescenzo, T. Malkin, and R. Ostrovsky. Single Database Private Information Retrieval implies Oblivious Transfer. In *EUROCRYPT 2000*, pages 122–138, 2000.
9. D. Dagon, W. Lee, and R. J. Lipton. Protecting secret data from insider attacks. In *Financial Cryptography and Data Security*, pages 16–30, 2005.
10. Y. Z. Ding, D. Harnik, A. Rosen, and R. Shaltiel. Constant-Round Oblivious Transfer in the Bounded Storage Model. In *Theory of Cryptography Conference*, volume 2951 of *LNCS*, pages 446–472. Springer, 2004.
11. B. Dubrov and Y. Ishai. On the randomness complexity of efficient sampling. In *ACM Symposium on Theory of Computing*, pages 711–720, 2006.
12. S. Dziembowski. Intrusion-resilience via the Bounded-Storage Model. In *Theory of Cryptography Conference*, volume 3876 of *LNCS*, pages 207–224. Springer, 2006.
13. S. Dziembowski. On Forward-Secure Storage. Cryptology ePrint Archive, 2006. http://eprint.iacr.org.
14. S. Dziembowski and U. Maurer. On generating the initial key in the Bounded-Storage Model. In *EUROCRYPT '04*, volume 3027 of *LNCS*, pages 126–137. Springer, 2004.
15. S. Dziembowski and U. Maurer. Optimal randomizer efficiency in the Bounded-Storage Model. *Journal of Cryptology*, 17(1):5–26, January 2004.
16. D. Harnik and M. Naor. On the compressibility of NP instances and cryptographic applications. Electronic Colloquium on Computational Complexity, Report TR06-022, 2006.
17. D. Harnik and M. Naor. On everlasting security in the Hybrid Bounded Storage Model, July 2006. accepted to the 33rd International Colloquium on Automata, Languages and Programming.
18. J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.
19. R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *ACM Symposium on Theory of Computing*, pages 44–61, 1989.
20. J. Kelsey and B. Schneier. Authenticating secure tokens using slow memory access. In *USENIX Workshop on Smart Card Technology*, pages 101–106, 1999.
21. E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, Computationally-Private Information Retrieval. In *38th Annual Symposium on Foundations of Computer Science*, pages 364–373, 1997.
22. E. Kushilevitz and R. Ostrovsky. One-way trapdoor permutations are sufficient for non-trivial single-server Private Information Retrieval. In *EUROCRYPT*, pages 104–121, 2000.
23. C.-J. Lu. Encryption against storage-bounded adversaries from on-line strong extractors. *Journal of Cryptology*, 17(1):27–42, January 2004.
24. U. Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 5(1):53–66, 1992.
25. T. Moran, R. Shaltiel, and A. Ta-Shma. Non-interactive timestamping in the Bounded Storage Model. In *CRYPTO 2004*, volume 3152 of *LNCS*, pages 460–476. Springer, 2004.
26. K. Pietrzak. Composition implies adaptive security in Minicrypt, May 2006. accepted to *EUROCRYPT 2006*.
27. R. L. Rivest. All-or-nothing encryption and the package transform. In *Fast Software Encryption*, volume 1267 of *LNCS*, pages 210–218. Springer, 1997.
28. S. P. Vadhan. Constructing locally computable extractors and cryptosystems in the Bounded-Storage Model. *Journal of Cryptology*, 17(1):43–77, January 2004.