Subversion-Resilient Authenticated Encryption without Random Oracles

Pascal Bemmann[1], Sebastian Berndt[2], Denis Diemert[1], Thomas Eisenbarth[2], Tibor Jager[1]

[1] Bergische Universität Wuppertal, Wuppertal, Germany
{bemmann, diemert, tibor.jager}@uni-wuppertal.de
[2] Universität zu Lübeck, Lübeck, Germany {s.berndt, thomas.eisenbarth}@uni-luebeck.de

Abstract. In 2013, the Snowden revelations have shown subversion of cryptographic implementations to be a relevant threat. Since then, the academic community has been pushing the development of models and constructions to defend against adversaries able to arbitrarily subvert cryptographic implementations. To capture these strong capabilities of adversaries, Russell, Tang, Yung, and Zhou (CCS'17) proposed CPA-secure encryption in a model that utilizes a trusted party called a *watchdog* testing an implementation before use to detect potential subversion. This model was used to construct subversion-resilient implementations of primitives such as random oracles by Russell, Tang, Yung, and Zhou (CRYPTO'18) or signature schemes by Chow et al. (PKC'19) but primitives aiming for a CCA-like security remained elusive in any watchdog model. In this work, we present the first subversion-resilient authenticated encryption scheme with associated data (AEAD) without making use of random oracles. At the core of our construction are subversion-resilient PRFs, which we obtain from weak PRFs in combination with the classical Naor–Reingold transformation. We revisit classical constructions based on PRFs to obtain subversion-resilient MACs, where both tagging and verification are subject to subversion, as well as subversion-resilient symmetric encryption in the form of stream ciphers. Finally, we observe that leveraging the classical Encrypt-then-MAC approach yields subversion-resilient AEAD. Our results are based on the trusted amalgamation model by Russell, Tang, Yung, and Zhou (ASIACRYPT'16) and the assumption of honest key generation.

Keywords: Subversion · Authenticated Encryption · Symmetric Cryptography

1 Introduction

While many cryptographic primitives nowadays have sound security proofs based on widely believed complexity-theoretic assumptions *implementing* these primitives securely is highly non-trivial as many possible attacks are not captured by the formal security models. For example, a malicious party can intentionally

embed a *covert channel* into an implementation of a cryptographic application. This kind of subversion is widely known as *kleptography* or *algorithm substitution attacks* (ASAs) and was first studied by Young and Yung [42].

The most prominent real-world example is the modification of the `Dual_EC` pseudorandom number generator [40] by the NSA. This pseudorandom generator involves two constants P and Q. If these constants are chosen independently, `Dual_EC` is secure [16], but the *designer* of the implementation (or the standard) can easily construct two dependent constants P and Q allowing them to reconstruct the state. We refer to the work of Checkoway et al. [19] for a more in-depth discussion. While the possibility of a backdoor due to [40] has been known since 2007, it was not known whether such backdoors would be used by law enforcement agencies such as the NSA. This changed with the revelation of internal NSA documents by Edward Snowden in 2013. These documents explicitly talk about *Project Bullrun*, where resources are used to "Insert vulnerabilities into commercial encryption systems, IT systems, networks, and endpoint communications devices used by targets." and "Influence policies, standards, and specifications for commercial public key technologies." [34]. These revelations reignited the interest in subversion attacks with the work by Bellare, Paterson, and Rogaway [10]. In general, the subverter $\mathcal{A}$ of a primitive Π has two roles: First, they provide a subverted implementation denoted by $\widetilde{\Pi}$. Second, they participate in the usual security experiment and aim to break the security guarantees provided by the unsubverted implementation of Π.

1.1 Subversion-Resilience Models

While many different subversion attacks were studied in different scenarios, the universal stateless attack by Bellare, Jaeger, and Kane [7] showed that it is *impossible* to prevent subversion attacks against symmetric encryption schemes without additional assumptions. Later, this impossibility result was extended by Berndt and Liśkiewicz [12] to also hold for all randomized algorithms. To circumvent these impossibility results different models were used. Here, we focus on the most prominent models, which are *cryptographic reverse firewalls*, *self-guarding schemes*, the *immunization model*, and the *watchdog model*. We emphasize that these models seem incomparable as they rely on different assumptions. Each use case requires a careful investigation of which model reflects the considered setting best.

Cryptographic reverse firewalls. Mironov and Stephens-Davidowitz [32] introduced *cryptographic reverse firewalls*. In this model a third party called *firewall* resides between the communicating parties involved in a cryptographic protocol. These firewalls "sanitize" the communication between the parties, usually by rerandomizing the messages sent by the parties. As the firewalls are modeled as non-subverted algorithms, this prevents leakage of sensitive information by the subverted implementation. An important feature of these firewalls is the fact that they do *not* have access to the secret keys of the parties and do not "provide security", i.e., they do not help non-subverted implementations in achieving security objectives. As described above, the typical approach for reverse firewalls

is to rerandomize the communication, see, for example, [4,17,18,20,25,32]. Even though many cryptographic primitives allow for rerandomization, primitives that aim for authenticity do not allow this. To still guarantee subversion-resilience in the firewall model, one needs to revert to strong assumptions. For example, Mironov and Stephens-Davidowitz [32] either make use of a symmetric bilinear map where the *inverse computational Diffie–Hellman assumption*, which is strictly stronger than the computational Diffie–Hellman assumption [32], holds or need a *strongly rerandomizable* asymmetric encryption scheme, for which no candidate is currently known. This allows them to design IND-CCA-secure protocols. Alternatively, Bossuat et al. [15] equip the reverse firewall with a key shared with both endpoints it aims to protect, which deviates from the standard assumptions described above.

Self-guarding schemes. Fischlin and Mazaheri [27] introduced the notion of *self-guarding schemes* that split each scheme into an *initialization phase* and a *computation phase*. Here, the initialization phase is supposed to be unsubverted, and thus outputs produced during that phase can be used to sanitize the output of the possibly subverted computation phase.

Immunization. Dodis, Ganesh, Golovnev, Juels and Ristenpart [24] formalized an *immunization model* for pseudorandom generators, where an immunization function is applied to the output of the generator. Depending on the knowledge of the subverter about this function, the authors show different approaches for choose the immunization function in such a way that the output of the subverted pseudorandom generator is indistinguishable from the output of an honest pseudorandom generator. In both the *semi-private* and the *private* model, the authors were able to construct such functions.

Watchdog model. Bellare, Paterson, and Rogaway [10] introduced a model in which a trusted monitoring party called *watchdog*[3] tests a primitive for subversion. Russell, Tang, Yung, and Zhou [37] introduced a non-black-box-variant of this model, called the *trusted amalgamation model*, where the designer of a primitive is allowed to split the primitive into different components that each can individually be checked the watchdog. The complete primitive is then amalgamated from these components by the *trusted* (i.e., not subverted) amalgamation function. This amalgamation function should thus be as simple as possible. For several cryptographic primitives subversion-resilient constructions were proposed, for example, trapdoor one-way permutations [37], pseudorandom generators [11, 37], symmetric IND-CPA-encryption schemes [38], hash functions [27], asymmetric IND-CPA-encryption schemes [11, 38], random oracles [5, 23, 39][4], signature schemes [21, 37], and key encapsulation mechanisms [11].

This paper also uses the trusted amalgamation model, so let us take a closer look at important details of this model. First, the order of the quantification is important, as discussed by Russell, Tang, Yung, and Zhou [37]. In the following, fix some primitive Π. One possibility to define that Π is subversion-resilient is

[3] They introduced the concept of detecting subversion rather than a "watchdog".
[4] Note that the proof in [39] contained an error that was later fixed in [13].

to demand that for each subverter $\mathcal{A}$, there is a watchdog WD such that either 1) the watchdog WD detects the subversion provided by $\mathcal{A}$ or 2) the subverted implementation provided by $\mathcal{A}$ does not weaken the security guarantees. While, at first glance, this model closely resembles the usual cryptographic security model, the watchdog now needs to depend on the adversary and, to protect against multiple different adversaries, *all* corresponding watchdogs need to be deployed. A much more desirable solution is to use a single *universal* watchdog WD such that every subverter $\mathcal{A}$ will be detected by this *single* watchdog. In our work, we use simple universal watchdogs that only sample uniformly random inputs and check the possibly subverted implementation against the honest specification.

For simplicity, we assume throughout this work that the attacker always provides stateless implementations. Similarly to the approach by Russell, Tang, Yung, and Zhou, we can also allow *rewindable* stateful implementations [37, Rem. 2.5]. This means that the watchdog is allowed to rewind the state of the implementation and test it for various inputs starting from the same state. If these are not rewindable, time bombs as introduced by Fischlin and Mazaheri [27] are possible, which seem to be unpreventable by an universal offline watchdog.

1.2 Towards Subversion-Resilient Authenticated Encryption

There has been huge progress over the last years and for many cryptographic primitives it was shown how to construct them in a subversion-resilient manner. However, authenticated encryption (AE in the following) where both encryption and decryption are subject to subversion has not been achieved in an offline watchdog model. The main challenge in constructing subversion-resilient AE is protecting the decryption algorithm. This is due to input-trigger attacks, which cannot be avoided without additional assumptions (such as trusted operations or the trusted amalgamation model) or using heavy machinery such as random oracles in a model where the decryption algorithm is modeled as a black box algorithm.

Russell, Tang, Yung, and Zhou [39] showed how to make random oracles subversion-resilient. This then leads to subversion-resilient signatures by Chow et al. [21], where both the signing and verification algorithm can be subverted while heavily relying on the subversion-resilient random oracle. The authors also showed how to construct subversion-resilient signatures in the standard model, where key generation and signing are subverted. While making use of random oracles may also directly lead to subversion-resilient AE, this work explores the possibility of achieving this notion *without* resorting to random oracles.

Another approach to fix subversion without random oracles was proposed by Ateniese, Francati, Magri, and Venturi [3]. Here, the authors show how to sanitize deterministic algorithms where all algorithms can be subverted, including the sanitizer. They present a transformation from an arbitrary algorithm to a subversion-resilient one. Their approach uses a secret (but tamperable) random source to generate the keys and the public parameters. However, the results in [3] only apply to deterministic primitives, for which it is well-known that they do

not achieve CPA security. In a setting where the adversary is allowed to freely choose inputs to its oracles, referred to as unconstrained games in [3], subversion-resilience is achieved by using an online watchdog, i.e., a watchdog which has access to transcripts of the considered security experiment. We, on the other hand, focus on offline watchdogs, which only get oracle access to the subverted algorithms *before* the security experiment is executed. As the construction of a subversion-resilient MAC where both the tagging and the verification algorithms may be subverted and the adversary chooses the input to its oracle is a crucial building block in our work, the results of [3] cannot be applied.

Finally, Armour and Poettering showed in a series of works [1, 2] several attacks on decryption of AEAD and verification of message authentication. Similar approaches are used by Russell, Tang, Yung, and Zhou [38] who effectively rerandomize the inputs to subverted algorithms. Unfortunately, this cannot be directly applied to decryption/verification, as input trigger attacks cannot be avoided without assuming that rerandomization is done as a trusted operation.

Hence, we deduced the following main research question of this work:

Is it possible to construct subversion-resilient authenticated encryption without random oracles in an offline watchdog model while only assuming non-cryptographic building blocks to be trusted?

In this paper, we answer this question affirmatively.

On the difficulties of constructing subversion-resilient AE. Before we describe our solution, it is instructive to understand and recognize the difficulties in constructing subversion-resilient authenticated encryption. The first major obstacle lies in the existence of *input trigger attacks*, first formalized by Degabriele, Farshim, and Poettering [22]: Such an attack modifies the underlying algorithm only on a single, arbitrary input $x^\star$ called the trigger. Whenever the algorithm is given $x^\star$ as input, it deviates from the specification by, e.g., outputting the secret key. As these triggers are chosen randomly by the attacker, no offline watchdog can detect the presence of these triggers. Thus, Degabriele, Farshim, and Poettering proposed a solution using an online watchdog. Now, these triggers are naturally connected to security experiments that model a *search problem*. Namely, in the final communication step of these experiments, the adversary usually sends some input which is directly evaluated by the underlying primitive. This direct transfer of information from the attacker to the primitive leads to trigger attacks, as the attacker can simply choose to submit such a trigger that solves the search problem. Now, security experiments that aim to secure the authenticity of information are typically modeled as search problems, where the task of the attacker is to produce some kind of forgery. Hence, such primitives are quite vulnerable to trigger attacks. Furthermore, even if one only considers *decision problems*, a direct transfer of information from the attacker to the primitive still allows for the use of input triggers.

1.3 Our Contribution

In this work, we show that subversion-resilient AE can be achieved without the use of random oracles. We use the trusted amalgamation model proposed by Russell, Tang, Yung, and Zhou [37, 38], which also inspired our work. To overcome input triggers, we need to avoid search problems and primitives that take direct input from the adversary. At the core of our construction are weak PRFs, i.e., PRFs that are only indistinguishable from random only if evaluated on *random* inputs. Our main contributions can be summarized as follows.

Weak PRFs are subversion-resilient. We first observe that these weak PRFs are naturally subversion-resilient, i.e., *any* implementation is indistinguishable from random given it passes the watchdog's check.

Subversion-resisilient PRFs from weak PRFs. As a next step, we use the classical Naor–Reingold transformation [33] that transforms a weak PRF into a PRF that can be queried *arbitrarily*. We prove that the Naor–Reingold construction also transforms a subversion-resilient weak PRF into a subversion-resilient PRF. In the context of subversion and the trusted amalgamation model, the trusted amalgamation applying the Naor–Reingold transformation can thus be seen as a *trusted data structure*, as the adversarially chosen inputs are only used to choose random keys in a trusted manner.

Subversion-resilient AE from PRFs. Given subversion-resilient PRFs, the classical "PRF-as-MAC" approach also guarantees subversion-resilience by assuming canonical verification and a trusted comparison operation. Making use of subversion-resilient PRFs again, we show that the classical randomized counter mode is also subversion-resilient assuming a trusted $\oplus$ operation, which can even be generalized to stream ciphers. From both a subversion-resilient MAC and encryption scheme, we then prove that subversion-resilient authenticated encryption can be achieved via the classical "Encrypt-then-MAC" approach.

1.4 Discussion

Finally, we discuss our contribution and the assumptions we make in this work.

Subversion-resilient AEAD from ROs. Note that given a subversion-resilient RO, as proposed in [39], one could replace the subversion-resilient PRF in our work by the RO and would obtain the similar results. However, argueably a subversion-resilient RO is a much stronger assumption than the existence of a weak PRF (in the standard model) that we base our results on in this work. To obtain a subversion-resilient RO, Russell et al. also make use of a trusted XOR operation and thus also need the same trust-assumptions as we do in our work.

Previous work on subversion-resilient MACs. In previous work, Fischlin, Janson, and Mazaheri [26] also observed that weak PRFs are a helpful tool to defend against backdoors. They show that a backdoored weak PRF implies a public key encryption scheme, arguing that the difference in performance can be easily detected. Further, they show that applying the randomized cascade (RC) construction by Maurer and Tessaro[31] to a weak PRF immunizes HMAC against

backdoors. As discussed later, using the RC construction also works for our construction, but requires to model the used prefix-free encoding as a trusted building block. Also, while Fischlin, Janson and Mazaheri focus on the properties of HMAC as a PRF, we focus on the subversion-resilience property of a MAC and its role in the Encrypt-then-MAC approach. As our model does not include detection based on the performance time of the subverted algorithm, we base the security of our construction on the subversion-resilience of weak PRFs.

Honest key generation. Contrary to previous works [21,38] we dismiss modeling subversion of key generation as this is typically only an *abstraction* for some means to derive a secure key. Even if one can construct key generation that could be executed by a single party, it is not clear how both parties would end up with the same key, potentially requiring a secure channel for key transportation. But to do this, both parties need to participate in a key-exchange protocol, which usually allows for a wide range of possible subversions (see, e.g., [25,32]). Hence, our work can be extended by some approach to derive uniform keys.

Weak PRFs. One may think that using weak PRFs instead of "standard" PRFs may be sufficient. However, it is not clear how to obtain MACs from (subversion-resilient) weak PRFs, as the security of MACs against forgery attacks are modeled as a search problem. While we would be able to answer all tagging queries via random queries to the PRF (e.g., via a Carter-Wegman-style [41] construction), handling the final forgery query is a challenge, as this query is directly made on the verification algorithm.

Trusted operations. We make use of several trusted operations, as it is not hard to see that some sort of trusted operations are needed to avoid trigger attacks proposed in previous works [1, 2, 22]. While being necessary, we aimed to minimize the number of trusted operations. Our approach only uses a trusted comparison and a trusted XOR. We believe that both of these (non-cryptographic operations) are simple enough to be either regarded as trusted or realized in hardware in a trusted manner. Not using a trusted XOR operation would most likely imply the need for some sort of rerandomization of ciphertexts before decrypting to remove biases. To the best of our knowledge, there is no AE scheme fulfilling such a property. Further, a trusted comparison seems unavoidable as otherwise a verification or decryption algorithm could reject or accept chosen inputs (since an adversarially chosen input is fed into a subverted component), as input triggers are again possible.

Relation to immunized PRGs. As described above, Dodis et al. [24] constructed subversion-resilient pseudorandom generators in both the *semi-private* and the *private* immunization model. We believe that classical constructions of PRFs from PRGs such as the one due to Goldreich, Goldwasser, and Micali [28] can be used to also obtain PRFs in the immunization model. But, while our constructions rely on an offline watchdog and the amalgamation assumption, the constructions in the immunization model rely on the fact that parts of the implementation (i.e., the immunization function) is hidden from the subverter. These assumptions are orthogonal to each other.

2 Subversion-resilience

In this section, we define the notion of subversion-resilience and loosely follow the approach by Russell, Tang, Yung, and Zhou [37]. Before we define the actual notions, we first describe our general notation and the overall setting.

2.1 Notation and Model

Notation. Recall that we need to distinguish between the *specification* of a primitive Π and the *implementation* of Π provided by the adversary. To make the distinction between an honest specification of a primitive Π and a (possibly) subverted implementation more explicit, we use the following notation throughout this paper. We denote by $\widehat{\Pi}$ the specification of the primitive and by $\widetilde{\Pi}$ the implementation of that primitive provided by the adversary.

A *security experiment* Exp for a cryptographic primitive Π with security objective GOAL involves one party, namely the adversary $\mathcal{A}$ trying to break the security objective against $\widehat{\Pi}$. In contrast, *subversion experiment* ExpSR is executed with an implementation of the considered primitive by the adversary and consists of three phases involving two parties: In the first phase, the *adversary* $\mathcal{A}$ provides provides a subverted implementation $\widetilde{\Pi}$. This implementation is then examined by a *watchdog* WD that tries to detect the subversion in the second phase. Finally, in the third phase, the adversary $\mathcal{A}$ takes part in the security experiment, where the subverted implementation is used. In the following, we always treat $\mathcal{A}$ as pair $(\mathcal{A}_0, \mathcal{A}_1)$, where $\mathcal{A}_0$ provides the subverted implementation $\widetilde{\Pi}$ and $\mathcal{A}_1$ takes part in the security experiment. As usual, we denote the security parameter by λ.

Amalgamation. As discussed earlier, preventing subversion attack in a purely black-box way is not possible, as universal undetectable attacks are known, e.g., by Berndt and Liśkiewicz [12]. Russell, Tang, Yung, and Zhou [37] thus introduced a non-black-box model called the *trusted amalgamation model*. While a primitive $\Pi = (\Pi_1, \ldots, \Pi_r)$ usually consists of a few different algorithms, the trusted amalgamation model splits all of these components into *subroutines*. For example, an encryption scheme usually consists of $r = 3$ algorithms $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$, but these might be composed of several subroutines used in different places. The trusted amalgamation model makes the use of these subroutines more explicit by representing a primitive as the list of subroutines $\pi = (\pi_1, \ldots, \pi_n)$ and a *trusted amalgamation function* Am that takes this list and produces the algorithms corresponding to the primitive. Let us get back to the example given above, $\mathsf{Am}(\pi) = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ that consists of subroutines π_i. We will allow the subverter to individually subvert the subroutines π_i arbitrarily by providing implementations $\widetilde{\pi}_i$, but assume that the amalgamation function is not subject to subversion. The security experiment is then played on $\widetilde{\Pi} = \mathsf{Am}(\widetilde{\pi})$. This assumption is usually justified by making this amalgamation function as simple as possible such that it can be checked automatically. For example, the amalgamation in the construction of Bemmann, Chen, and Jager [11] and in the

constructions of Russell, Tang, Yung, and Zhou [38] only handles inputs and outputs of different subroutines and makes use of a few XOR operations. In the following, we thus represent the specification $\widehat{\Pi}$ of a primitive as $\widehat{\Pi} = (\mathsf{Am}, \pi)$, where $\pi = (\pi_1, \ldots, \pi_n)$ is the list of subroutines. We also need to consider the amalgamation function for a *single* algorithm Π_i of a primitive, which we denote by Am_i. That is, the amalgamation $\mathsf{Am}(\pi) = (\mathsf{Am}_1(\pi), \ldots, \mathsf{Am}_r(\pi))$ actually consists of a vector of amalgamation functions such that there is a function for each algorithm of the primitive. As a shortcut, we simply write $(\mathsf{Am}, \widehat{\Psi})$ if a construction uses a subversion-resilient $\widehat{\Psi} = (\mathsf{Am}_\Psi, \psi)$ as a building block.

Split-program model. In addition to trusted amalgamation, Russell, Tang, Yung, and Zhou [37] also used the *split-program* methodology. Similar to modern programming techniques, it is assumed that randomness generation is split from a randomized algorithm. The randomness generator and the deterministic algorithm can then be tested individually by the watchdog. We also use this methodology in our work.

Randomness generation. The constructions in this paper rely on "good" (i.e., in particular trusted) randomness being available. For this, either one of the constructions proposed by Russell, Tang, Yung, and Zhou [38] or Bemmann, Chen, and Jager [11] can be used. Both works contain constructions generating randomness that is indistinguishable from random for the adversary providing the implementation *without* using random oracles. For this paper, we can use both constructions. Hence, for simplicity, we abstract away the randomness generation and assume that our constructions generate uniformly random bits, while being able to test randomized algorithm on selected random coins. This assumption allows us to simplify notation and focus on our contributions to enable authentication in the presence of subversion.

2.2 Subversion-Resilience

Next, we define the notion of subversion-resilience. Intuitively, we extend a "conventional" security experiment Exp by a preceding check for subversion of the primitive. Afterwards, the security experiment is executed. This is illustrated in Fig. 1. As we study both decision (i.e. indistinguishability) and search (i.e. unpredictability) problems in this paper, we associate with experiment Exp a "baseline win probability" denoted by δ that gives the winning probability of a naive attacker, i.e., $\delta = 0$ for search problems and $\delta = 1/2$ for decision problems. To extend Exp, we first run $\mathcal{A}_0$ to obtain a subverted implementation $\widetilde{\pi}$ (Fig. 1, l. 1). The watchdog WD then tests the implementation before we run the security experiment Exp with adversary $\mathcal{A}_1$ on the subverted implementation $\widetilde{\Pi} = \mathsf{Am}(\widetilde{\pi})$ as usual (Fig. 1, l. 3). The variable state is only used to synchronize $\mathcal{A}_0$ and $\mathcal{A}_1$. Throughout this work we use the convention that the watchdog outputs "true" in the case that subversion is detected. To formalize subversion-resilience, consider the next definition and the corresponding security experiment shown in Fig. 1.

Definition 1. *A specification of a primitive* $\widehat{\Pi} = (\mathsf{Am}, \pi)$ *is* GOAL-secure under subversion in the offline watchdog model with trusted amalgamation *if one*

$\mathsf{ExpSR}^{\mathrm{GOAL},\widehat{\Pi}}_{\mathsf{WD},\mathcal{A}}(1^\lambda)$

1 : $(\widetilde{\pi}, \mathsf{state}) \xleftarrow{\$} \mathcal{A}_0(1^\lambda)$

2 : $b_{\mathsf{WD}} \leftarrow \mathsf{WD}^{\widetilde{\pi}}(1^\lambda)$

3 : **return** $\mathsf{Exp}^{\mathrm{GOAL},\mathsf{Am}(\widetilde{\pi})}_{\mathcal{A}_1(\mathsf{state})}(1^\lambda)$

Fig. 1: The security experiment for GOAL-security under subversion.

can efficiently construct a ppt watchdog algorithm WD *such that for any ppt adversary* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *it holds*

$$\mathsf{AdvSR}^{\mathrm{GOAL},\widehat{\Pi}}_{\mathcal{A}}(1^\lambda, \delta) \text{ is negligible} \quad \text{or} \quad \mathsf{Det}_{\mathsf{WD},\mathcal{A}}(1^\lambda) \text{ is non-negligible}$$

where $\mathsf{AdvSR}^{\mathrm{GOAL},\widehat{\Pi}}_{\mathcal{A}}(1^\lambda, \delta) = |\Pr[\mathsf{ExpSR}^{\mathrm{GOAL},\widehat{\Pi}}_{\mathsf{WD},\mathcal{A}}(1^\lambda) = 1] - \delta|$ *and* $\mathsf{Det}_{\mathsf{WD},\mathcal{A}}(1^\lambda) = |\Pr[\mathsf{WD}^{\widetilde{\pi}}(1^\lambda) = 1] - \Pr[\mathsf{WD}^{\pi}(1^\lambda) = 1]|$ *using the experiment shown in Fig. 1, with* $\delta \in \{0, \frac{1}{2}\}$ *indicating whether a search or a decision problem is considered.*

Note that $\mathsf{AdvSR}^{\mathrm{GOAL},\widehat{\Pi}}_{\mathcal{A}}(1^\lambda, \delta)$ is not parameterized by the watchdog WD. We chose this approach to simplify notation, as the testing of the watchdog does not influence the advantage of the adversary directly. For public key encryption, this model is not equivalent to the model proposed by Russell, Tang, Yung, and Zhou [38], where the adversary has access to a subverted encryption oracle. For symmetric encryption, our more general definition captures theirs with some differences in syntax.

As mentioned earlier, we assume stateless subversion and that key and randomness generation are trusted. In order to shorten notation, we call primitives just GOAL-*secure under subversion.*

2.3 Achieving Subversion-resilience

To prove our upcoming PRF construction subversion-resilient, we use an observation made by Russell, Tang, Yung, and Zhou [37]. If a deterministic primitive is only given inputs according to a *public* distribution and the implementation deviates from the specification with some probability δ (with inputs chosen according to this public input distribution), then a ppt watchdog can detect this with probability at least δ. Hence, in order to stay undetected, the number of inputs the implementation deviates from the specification needs to be negligible.

Lemma 1. *Consider an implementation* $\widetilde{\Pi} := (\widetilde{\pi}_1, \ldots, \widetilde{\pi}_k)$ *of a specification* $\widehat{\Pi} = (\widehat{\pi}_1, \ldots, \widehat{\pi}_k)$*, where* $\pi_1, \ldots, \pi_k$ *are deterministic algorithms. Additionally, for each security parameter* λ*, public input distributions* $X^1_\lambda, \ldots, X^k_\lambda$ *are defined respectively. If there exists a* $j \in [k]$ *such that* $\Pr[\widetilde{\pi}_j(x) \neq \widehat{\pi}_j(x) : x \xleftarrow{\$} X^j_\lambda] = \delta$*, this can be detected by a ppt offline watchdog with probability at least* δ*.*

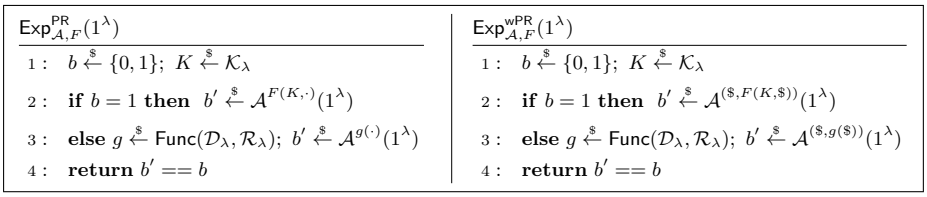

Fig. 2: The security experiment for (weak) PRFs. Here, \$ denotes an input argument chosen uniformly at random from $\mathcal{D}_\lambda$ upon any query issued by the adversary. Further, if $K \in \mathcal{K}_\lambda$, the oracle $F(K, \cdot)$ can only be queried on elements of $\mathcal{D}_\lambda$.

An instructive example to understand the usefulness of this lemma is the following. Suppose that we are given a single function f and a probability distribution X on the domain of f. In an experiment, the adversary can now issue a query, where $x \xleftarrow{\$} X$ is drawn and the pairs $(x, f(x))$ is given to the adversary. The goal of the adversary is to obtain a sample $(x^\star, \widetilde{f}(x^\star))$, where $x^\star \in X^*$ for some subset $X^\star \subseteq \mathrm{Supp}(X)$ such that $\widetilde{f}(x^\star) \neq \widehat{f}(x^\star)$ where $\mathrm{Supp}(X)$ is the subset of values the variable X can take. Clearly, if the adversary can only perform a bounded number of samples, the density of $X^\star$ wrt. X cannot be arbitrarily small. But, as the distribution X is publicly known, a watchdog can also sample according to X and check the implementation $\widetilde{f}$ against the specification $\widehat{f}$ on these samples. Then, it is not hard to see that the adversary wins if the watchdog distinguishes the implementation from the specification.

3 Pseudorandom Functions

Intuitively, a PRF is a keyed function $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ associated with a key space $\mathcal{K}$, that is indistinguishable from a function sampled uniformly at random from the set of all functions $\mathcal{D} \to \mathcal{R}$. More formally, $\mathcal{K} = \bigcup_{\lambda\in\mathbb{N}} \mathcal{K}_\lambda$, $\mathcal{D} = \bigcup_{\lambda\in\mathbb{N}} \mathcal{D}_\lambda$, and $\mathcal{R} = \bigcup_{\lambda\in\mathbb{N}} \mathcal{R}_\lambda$. Additionally, we use $\mathsf{Func}(\mathcal{D}, \mathcal{R})$ to denote the set of all functions mapping elements from $\mathcal{D}$ to $\mathcal{R}$. In this paper, we only consider spaces that are subsets of $\{0,1\}^*$.[5] Let us recall the standard definition of (weak) PRFs.

Definition 2. *Let $T \in \{\mathsf{wPR}, \mathsf{PR}\}$ and let $\mathsf{Exp}^T_{\mathcal{A},F}$ be defined as shown in Fig. 2. We define $\mathsf{Adv}^T_{\mathcal{A},F}(1^\lambda) := |\Pr[\mathsf{Exp}^T_{\mathcal{A},F}(1^\lambda) = 1] - 1/2|$. We say that F is* pseudorandom *if $\mathsf{Adv}^{\mathsf{PR}}_{\mathcal{A},F}(1^\lambda)$ is negligible for all ppt adversaries $\mathcal{A}$. Further, we say that F is* weakly pseudorandom *if $\mathsf{Adv}^{\mathsf{wPR}}_{\mathcal{A},F}(1^\lambda)$ is negligible for all ppt adversaries $\mathcal{A}$.*

Weak PRFs are subversion-resilient. The first observation is that since all inputs given to the PRF are distributed uniformly at random, they follow a

[5] We actually only require that we can sample uniform elements of $\mathcal{D}$ and $\mathcal{K}$ efficiently and that $\mathcal{D}$ is a quasi group with operation $\oplus$.

distribution that is publicly known. This allows us to apply Lemma 1. For an implementation $\widetilde{F}$ of a specification $\widehat{F}$ of a weak PRF, let $\mathsf{Neq}_\lambda \subseteq \mathcal{K}_\lambda \times \mathcal{D}_\lambda$ be the set of inputs, where $\widetilde{F}$ deviates from the specification, i.e., $\mathsf{Neq}_\lambda = \{(K, x) \in \mathcal{K}_\lambda \times \mathcal{D}_\lambda \mid \widetilde{F}(K, x) \neq \widehat{F}(K, x)\}$. Now, consider the proportional amount p of Neq_λ, i.e., $p = \frac{|\mathsf{Neq}_\lambda|}{|\mathcal{K}_\lambda \times \mathcal{D}_\lambda|}$. As the input distribution of the weak PRF experiment is public, Lemma 1 now directly implies the existence of a ppt watchdog with detection probability p (simply testing $\widetilde{F}$ on uniformly random inputs). Hence, for an adversary to succeed in the subversion-experiment, p must be negligible. But, as all inputs to the weak PRF are drawn randomly, the probability that an adversary making q queries will ever encounter an input to the weak PRF that belongs to Neq_λ is bounded by $q \cdot p$ and is thus negligible for ppt adversarys, as q is bounded by a polynomial in λ, yielding the following theorem.

Theorem 1. *If F is weakly pseudorandom, then the trivial specification $\widehat{F} = F$ is weakly pseudorandom under subversion.*

3.1 Constructing Subversion-Resilient PRFs

In the following, we use the classical Naor–Reingold construction [33] to construct a (standard) PRF from a weak PRF that is subversion-resilient.

The Naor–Reingold construction. Let $F_w \colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a weak PRF. For the sake of simplicity, we only focus on the case that elements of $\mathcal{K}$, $\mathcal{D}$, and $\mathcal{R}$ are of equal length and refer the reader to the survey by Bogdanov and Rosen [14] for generalizations. We now construct a (standard) PRF $F^{(\ell)} \colon \mathcal{K}^{2\cdot\ell} \times \{0,1\}^\ell \to \mathcal{R}$ that is parameterized by some integer ℓ of the form $\ell = 2^r$ describing the message length. It is easiest to construct $F^{(\ell)}$ inductively. In the simplest case of $\ell = 1$, the key of $F^{(\ell)}$ consists of two randomly sampled keys of F (i.e., two random bit strings $K_0, K_1 \in \mathcal{K}$). On input $x \in \{0,1\}$, it returns K_x, i.e., $F^{(1)}((K_0, K_1), x) = K_x$. Given $F^{(\ell)}$, we construct $F^{(2\ell)}$ inductively as follows. A key of $F^{(2\ell)}$ consists of two keys $K_0^{(\ell)}$ and $K_1^{(\ell)}$ of $F^{(\ell)}$ (which in turn consists each of 2ℓ keys of F_w). On input $x = (x_1, x_2, \ldots, x_{2\ell})$, the function $F^{(2\ell)}$ applies $F^{(\ell)}$ with the first key $K_0^{(\ell)}$ to the first half of x to obtain a key for F_w and then computes $F^{(\ell)}$ with the second key on the second half of x to obtain a value. More formally,

$$F^{(2\ell)}((K_0^{(\ell)}, K_1^{(\ell)}), (x_1, \ldots, x_{2\ell})) = \\ F_w(F^{(\ell)}(K_0^{(\ell)}, (x_1, \ldots, x_\ell)), F^{(\ell)}(K_1^{(\ell)}, (x_{\ell+1}, \ldots, x_{2\ell}))).$$

An useful alternate interpretation is the following (shown in Fig. 3). The key of $F^{(2\ell)}$ consists of 2ℓ key pairs $(K_{i,0}, K_{i,1})$ for $i = 1, \ldots, 2\ell$. On input $(x_1, \ldots, x_{2\ell})$, we construct a complete binary tree of height r, where $r = \log(2\ell)$. The final level of this binary tree contains the 2ℓ leaves. To produce the output of $F^{(2\ell)}$, we now construct a *labeling* of the vertices. We first label the i-th leave of the tree with K_{i,x_i}, i.e., the message bit x_i determines whether we take $K_{i,0}$ or $K_{i,1}$. To obtain the label of an inner node v of the tree, we compute $F_w(\mathrm{left}(v), \mathrm{right}(v))$, where

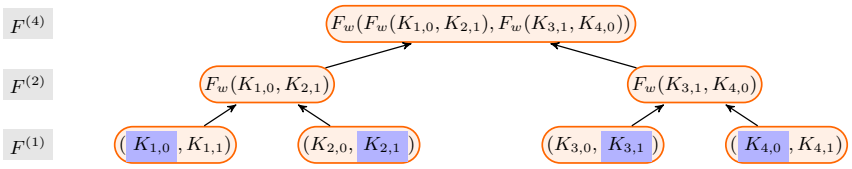

Fig. 3: The alternate interpretation of the Naor-Reingold construction as labeling of a complete binary tree for the value $x = (0, 1, 1, 0)$. The corresponding leaf values are marked in blue.

left(v) (resp. right(v)) is the label of the left (resp. right) child of v. Finally, the output of $F^{(2\ell)}$ is the label of the root of the tree. It is well-known that this construction gives a PRF $F^{(\ell)}$ if F_w is weakly pseudorandom.

Theorem 2 ([33, Thm. 5.1][6]). *Let $\ell \in \mathbb{N}$ with $\ell = 2^r$. If F_w is weakly pseudorandom, then $F^{(\ell)}$ is pseudorandom.*

Now, observe that a non-subverted, honest call-structure to the underlying function F_w (which is trivially true due to our amalgamation assumption) directly implies the subversion-resilience of $F^{(\ell)}$. On the lowest level, $F^{(1)}$ will only return completely random values, which is clearly subversion-resilient. The inputs to $F^{(2)}$ are thus completely random values which follow a public input distribution and Lemma 1 directly implies subversion-resilience.

Theorem 3. *If $\widehat{F}$ is pseudorandom under subversion, then for each ℓ with $\ell = 2^r$, $\widehat{F^{(\ell)}}$ is pseudorandom under subversion.*

Proof. Our watchdog simply samples random keys and random inputs for the weak PRF F_w and checks for deviations from the specification. As for the subversion-resilience of F_w discussed above, let Neq_λ be the set of inputs for which $\widetilde{F_w}$ deviates from its specification. As shown before, by applying a watchdog that simply tests a sufficient number of random inputs to F_w, we know that the probability $p = \frac{|\mathsf{Neq}_\lambda|}{|\mathcal{K}_\lambda \times \mathcal{D}_\lambda|}$ is negligible. On the lowest level, corresponding to $F^{(1)}$, we only choose one of two random values. Hence, the watchdog can easily verify the correctness of $F^{(1)}$ as there are only constantly many different inputs. In the next level, corresponding to $F^{(2)}$, the function F_w is only applied to these completely random inputs. If an adversary makes $q \in \mathsf{poly}(\lambda)$ many queries, the probability that one of the calls to F_w on this level deviates from the specification is at most $q \cdot (\ell/2) \cdot p$, which is negligible. Conditioned on the event that all calls to F_w on the level corresponding to $F^{(2)}$ follow the specification, the inputs to the $q \cdot (\ell/4)$ calls to F_w on the level corresponding to $F^{(4)}$ are indistinguishable from random (due to the security of the specification of the weak PRF). Hence, with probability $q \cdot (\ell/4) \cdot p$, these inputs also do not belong to

[6] Naor and Reingold use the notion of a *synthesizer*, which are in our context equivalent to weakly PRFS [14].

Neq_λ, if the inputs on the level corresponding to $F^{(2)}$ do not belong to Neq. Let $E_{\ell'}$ be the event that all inputs on the level corresponding to $F^{(\ell')}$ do not belong to Neq_λ. By iterating the above argumentation, it is not hard to see that $\Pr[E_{\ell'} \mid E_{\ell'/2}] \geq 1 - q \cdot (\ell/\ell') \cdot p$ holds. From $\Pr[E_2] \geq 1 - q \cdot (\ell/2) \cdot p$, we can conclude via a simple induction that

$$\begin{aligned}\Pr[E_{\ell'}] &= \Pr[E_{\ell'} \mid E_{\ell'/2}] \cdot \Pr[E_{\ell'/2}] + \Pr[E_{\ell'} \mid \neg E_{\ell'/2}] \cdot \Pr[\neg E_{\ell'/2}] \\ &\geq \prod_{i=1}^{r'} (1 - q \cdot (\ell/2^i) \cdot p)\end{aligned}$$

for $\ell' = 2^{r'}$. Hence, all probabilities $\Pr[E_{\ell'}]$ are of the form $1 - \mathsf{negl}(\lambda)$ for a negligible function negl. We can thus conclude that the probability that any input to F_w belongs to Neq_λ is negligible. The original security guarantee due to Theorem 2 then directly implies the subversion-resilience of $F^{(\ell)}$. □

Alternative constructions. In principle any transformation from weak to standard PRFs can be used in our construction. We chose the Naor-Reingold construction, due to its simplicity and as it only requires the amalgamation function to act as a trusted data structure and no trusted operations. Alternatively, the randomized cascade construction by Maurer and Tessaro [31] can be used. There adversarially chosen messages are directly fed into a prefix-free encoding, which then needs to be modeled as a trusted operation in order to prevent input triggers. Another alternative is the IC construction by Maurer and Sjödin [30], where the input provided by the adversary is processed bitwise and either a weak PRF is executed or a previously computed value is used in an iterative process.

4 MAC

We now show how to construct subversion-resilient MACs using any subversion-resilient PRFs, e.g., the one from the previous section. Let us first recall the standard definition of MACs. A MAC works on a keyspace $\mathcal{K}$, message space $\mathcal{M}$, and tag space $\mathcal{T}$. Also, we have $\mathcal{K} = \bigcup_{\lambda\in\mathbb{N}} \mathcal{K}_\lambda$, $\mathcal{M} = \mathcal{M}_{\lambda\in\mathbb{N}}$, and $\mathcal{T} = \bigcup_{\lambda\in\mathbb{N}} \mathcal{T}_\lambda$.

Definition 3. *We call a triple* $\mathsf{MAC} = (\mathsf{KGen}, \mathsf{Tag}, \mathsf{Vf})$ *a* message authentication code (MAC) *for key space $\mathcal{K}$, message space $\mathcal{M}$, and tag space $\mathcal{T}$. The randomized* key generation algorithm KGen *produces upon the security parameter 1^λ as input a key $K \xleftarrow{\$} \mathcal{K}_\lambda$. The randomized* tagging algorithm Tag *is given a key $K \in \mathcal{K}_\lambda$ and a message $M \in \mathcal{M}_\lambda$ and returns a tag $T \in \mathcal{T}_\lambda$. The deterministic* verification algorithm Vf *is given a key K, a message M, and a tag T and returns a bit b.*

For correctness, we require that for all $K \in \mathcal{K}_\lambda$, for all $M \in \mathcal{M}_\lambda$, and all $T \in \mathrm{Supp}(\mathsf{Tag}(K, M))$, it holds $\mathsf{Vf}(K, T) = 1$. Next, we recall the standard security notion of (strong) unforgeability of MACs.

Definition 4. *Let* MAC *be a MAC and let* $\mathsf{Exp}^{\text{SUF-CMA}}_{\mathcal{A},\mathsf{MAC}}(1^\lambda)$ *be defined as shown in Fig. 4. We define* $\mathsf{Adv}^{\text{SUF-CMA}}_{\mathcal{A},\mathsf{MAC}}(1^\lambda) \coloneqq \Pr[\mathsf{Exp}^{\text{SUF-CMA}}_{\mathcal{A},\mathsf{MAC}}(1^\lambda) = 1]$ *and say that* MAC *is* strongly unforgeable under a chosen message attack*, or* SUF-CMA-*secure, if* $\mathsf{Adv}^{\text{SUF-CMA}}_{\mathcal{A},\mathsf{MAC}}(1^\lambda)$ *is negligible for all ppt adversaries $\mathcal{A}$.*

$\mathsf{Exp}^{\text{SUF-CMA}}_{\mathcal{A},\mathsf{MAC}}(1^\lambda)$

1 : $K \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$; $\mathsf{Query} \leftarrow \emptyset$

2 : $(M,T) \xleftarrow{\$} \mathcal{A}^{\mathsf{Tag}(K,\cdot)}(1^\lambda)$

3 : **return** $(M,T) \notin \mathsf{Query} \wedge \mathsf{Vf}(K,M,T) = 1$

Fig. 4: The forgery experiment for MACs. On input $M \in \mathcal{M}_\lambda$ the oracle $\mathsf{Tag}(K,\cdot)$ computes $T \xleftarrow{\$} \mathsf{Tag}(K,M)$, stores (M,T) in Query and returns T.

4.1 MAC from PRFs

Consider the following generic construction of a (fixed-length) deterministic MAC based on a PRF. Let F be a keyed function $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$. We define $\mathsf{MAC}_F = (\mathsf{KGen}_F, \mathsf{Tag}_F, \mathsf{Vf}_F)$ with key space $\mathcal{K}$, message space $\mathcal{M}$, and tag space $\mathcal{T}$ such that KGen_F on input 1^λ outputs a uniform key $K \xleftarrow{\$} \mathcal{K}_\lambda$, Tag_F on input key $K \in \mathcal{K}_\lambda$ and message $M \in \mathcal{D}_\lambda$, returns $T = F(K,M)$, and Vf_F on input a key $K \in \mathcal{K}_\lambda$, message $M \in \mathcal{D}_\lambda$, and a tag $T \in \mathcal{R}_\lambda$, outputs 1 if and only if $T = \mathsf{Tag}_F(K,M)$. It is a well-known result by Goldreich, Goldwasser and Micali that a PRF can directly be used to construct a MAC which we recall.

Theorem 4 ([29]). *If F is pseudorandom, then MAC_F is* SUF-CMA*-secure.*

Theorem 4 guarantees that the subversion-resilience of the underlying function F directly transfers to MAC_F, if the $=$ operation during the verification is part of the trusted amalgamation. Thus, $\widehat{\mathsf{MAC}_F} = (\mathsf{Am}, \widehat{F})$, where $\widehat{F}$ is a subversion-resilient PRF, and Am calls the PRF for tagging and for verification it recomputes the MAC using the implementation of the PRF and compares the result with its input. Thus, the watchdog runs the watchdog of the subversion-resilient PRF. Finally, we can conclude that the PRF presented in Section 3.1 (built from a weak PRF) is a subversion-resilient deterministic MAC as well.

Theorem 5. *If $\widehat{F}$ is pseudorandom under subversion, then $\widehat{\mathsf{MAC}_F} = (\mathsf{Am}, \widehat{F})$ is* SUF-CMA*-secure under subversion assuming a trusted $=$ operation.*

Corollary 1. *The specification $\widehat{\mathsf{MAC}_{F^{(\ell)}}} = (\mathsf{Am}, \widehat{F^{(\ell)}})$ is* SUF-CMA*-secure under subversion assuming a trusted $=$ operation.*

5 Symmetric Encryption

In this section, we construct subversion-resilient symmetric encryption using a classical construction based on PRFs that has indistinguishable encryptions under a chosen-message attack. A symmetric encryption scheme works on a keyspace $\mathcal{K}$, message space $\mathcal{M}$, and ciphertext space $\mathcal{C}$. As usual, we have $\mathcal{K} = \bigcup_{\lambda\in\mathbb{N}} \mathcal{K}_\lambda$, $\mathcal{M} = \mathcal{M}_{\lambda\in\mathbb{N}}$, and $\mathcal{C} = \bigcup_{\lambda\in\mathbb{N}} \mathcal{C}_\lambda$.

$\underline{\mathsf{Exp}^{\text{IND\$-CPA}}_{\mathcal{A},\mathsf{SE}}(1^\lambda)}$

1 : $K \xleftarrow{\$} \mathsf{KGen}(1^\lambda);\ b \xleftarrow{\$} \{0,1\}$

2 : $b' \xleftarrow{\$} \mathcal{A}^{\mathsf{RoR}_b(\cdot)}(1^\lambda)$

3 : **return** $(b = b')$

Fig. 5: Security experiment for IND$-CPA-security, where $\mathsf{RoR}_0(M) = \$_K(M)$ and $\mathsf{RoR}_1(M) = \mathsf{Enc}(K, M)$ such that $\$_K(M)$ computes $C \xleftarrow{\$} \mathsf{Enc}(K, M)$ and if $C = \bot$, outputs $\bot$, and otherwise, outputs a random string of length $|C|$.

Definition 5. *We call a triple* $\mathsf{SE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *a symmetric encryption scheme* SE *with key space* $\mathcal{K}$*, message space* $\mathcal{M}$*, and ciphertext space* $\mathcal{C}$*. The randomized* key generation algorithm KGen *outputs upon the security parameter* 1^λ *as input a key* $K \xleftarrow{\$} \mathcal{K}_\lambda$*. The randomized* encryption algorithm Enc *is given a key* $K \in \mathcal{K}_\lambda$ *and a message* $M \in \mathcal{M}_\lambda$ *and returns either a ciphertext* $C \in \mathcal{C}_\lambda$ *or a symbol* $\bot$*. The deterministic* decryption algorithm Dec *is given a key* K *and a ciphertext* C*, and returns either a message* $M \in \mathcal{M}_\lambda$ *or the symbol* $\bot$*.*

We say that Π has *perfect correctness*, i.e., for all $K \in \mathcal{K}_\lambda$, all $M \in \mathcal{M}_\lambda$ and all $C \in \mathrm{Supp}(\mathsf{Enc}(K, M))$, we have $\mathsf{Dec}(K, C) = M$ if $C \neq \bot$. For security, we consider IND$-CPA-security (i.e., indistinguishability from random bits) [35,36], which can be shown to imply IND-CPA-security in the left-or-right sense (see, e.g., [6]) by a straightforward reduction.

Definition 6. *Let* SE *be a symmetric encryption scheme and let* $\mathsf{Exp}^{\text{IND\$-CPA}}_{\mathcal{A},\mathsf{SE}}(1^\lambda)$ *be defined as shown in Fig. 5. We define* $\mathsf{Adv}^{\text{IND\$-CPA}}_{\mathcal{A},\mathsf{SE}}(1^\lambda) \coloneqq |\Pr[\mathsf{Exp}^{\text{IND\$-CPA}}_{\mathcal{A},\mathsf{SE}}(1^\lambda) = 1] - 1/2|$ *and say that* SE *is* IND$-CPA*-secure if* $\mathsf{Adv}^{\text{IND\$-CPA}}_{\mathsf{SE}}(\mathcal{A})$ *is negligible for all ppt adversaries* $\mathcal{A}$*.*

Symmetric encryption from PRFs. In Section 3, we construct subversion-resilient PRFs. A classical use case of PRFs is the construction of symmetric encryption. Recall the following construction of a stream cipher $\mathsf{SE}_{\mathrm{KS}} = (\mathsf{KGen}_{\mathrm{KS}}, \mathsf{Enc}_{\mathrm{KS}}, \mathsf{Dec}_{\mathrm{KS}})$ based on a PRF $\mathrm{KS}\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$. KGen on input 1^λ outputs a uniform key $K \xleftarrow{\$} \mathcal{K}_\lambda$, $\mathsf{Enc}_{\mathrm{KS}}$ on input a key $K \in \mathcal{K}_\lambda$ and a message $M \in \mathcal{R}_\lambda$, outputs a ciphertext (IV, C), where $\mathsf{IV} \xleftarrow{\$} \mathcal{D}$ and $C := \mathrm{KS}(K, \mathsf{IV}) \oplus M$, and $\mathsf{Dec}_{\mathrm{KS}}$ on input a key $K \in \mathcal{K}_\lambda$ and a ciphertext $(\mathsf{IV}, C) \in \mathcal{D}_\lambda \times \mathcal{R}_\lambda$, outputs $M := \mathrm{KS}(K, \mathsf{IV}) \oplus C$.

Subversion-resilience of stream ciphers. Next, we show that the above construction is subversion-resilient if the underlying function KS is a subversion-resilient weak PRF. Thus, we consider $\widehat{\mathsf{SE}_{\mathrm{KS}}} = (\mathsf{Am}, \widehat{\mathrm{KS}})$ where $\widehat{\mathrm{KS}}$ is the specification of a subversion-resilient weak PRF and AM randomly chooses IVs and then calls the underlying PRF and applies the trusted $\oplus$ operation.

Theorem 6. *If* $\widehat{\mathrm{KS}}$ *is weakly pseudorandom under subversion, then* $\widehat{\mathsf{SE}_{\mathrm{KS}}}$ *is* IND$-CPA*-secure under subversion given that the randomness generation and* $\oplus$ *operation are trusted.*

Proof (Sketch). The watchdog runs the watchdog for $\widehat{\mathrm{KS}}$. The main idea of the proof is that the output of KS is indistinguishable from uniformly random bits for uniformly random IVs and any adversary, even under subversion. Thus for any message M, the output of Enc is indistinguishable from uniformly random bits under subversion for any adversary as well. Hence, $\widehat{\mathsf{SE}_{\mathrm{KS}}}$ is IND$-CPA-secure under subversion.

Key stream derivation of CTR. It remains to demonstrate that this construction can actually be instantiated. A popular instantiation of the above construction is the (randomized) counter mode (CTR$). The above construction in combination with $\mathrm{KS}_{\mathrm{CTR}}$ defined next yields CTR$. Given a PRF $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$, we define the function $\mathrm{KS}_{\mathrm{CTR}}\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}^\ell$ as

$$(K, \mathsf{IV}) \mapsto \big(F(K, \mathsf{IV}), F(K, \mathsf{IV} \oplus \langle 1 \rangle_n), \ldots, F(K, \mathsf{IV} \oplus \langle \ell - 1 \rangle_n)\big),$$

where $\ell \in \mathsf{poly}(\lambda)$ and $\langle i \rangle_n$ denotes the n-bit binary representation of $i \in \mathbb{N}$. Next, we show that $\mathrm{KS}_{\mathrm{CTR}}$ is a subversion-resilient weak PRF assuming that handling the state (i.e. the counter) is modeled as part of the amalgamation.

Theorem 7. *If* $\widehat{F}$ *is pseudorandom under subversion, then* $\widehat{\mathrm{KS}_{\mathrm{CTR}}}$ *is a subversion-resilient weak PRF under the assumption that randomness generation and the* $\oplus$ *operation are trusted.*

Proof (Sketch). The watchdog for $\widehat{\mathrm{KS}_{\mathrm{CTR}}}$ runs the watchdog for $\widehat{F}$ as a subroutine. To prove that the $\mathrm{KS}_{\mathrm{CTR}}$ is secure even if the building block F is subverted, the main idea is as follows: If F is indistinguishable from random (even under subversion), then $\mathrm{KS}_{\mathrm{CTR}}$ is indistinguishable from random for uniformly random inputs as long as the sequence $(\mathsf{IV}, \ldots, \mathsf{IV} \oplus \langle \ell - 1 \rangle_n)$ does not overlap for two PRF queries. This is because an adversary directly could observe the structure and distinguish the function from random. By a simple argument, one can bound this probability by $\frac{q^2 \ell}{|\mathcal{D}|}$, which is negligible for polynomial block length ℓ, polynomial IV length $\log(|\mathcal{D}|)$, and a polynomial number of PRF queries q.

Note that $\mathrm{KS}_{\mathrm{CTR}}$ is not a PRF as the adversary can simply choose the IVs such that they overlap which enables the adversary to distinguish $\mathrm{KS}_{\mathrm{CTR}}$ from a truly random function with overwhelming probability. Finally, Theorem 6 and 7 imply that the randomized counter mode CTR$ is subversion-resilient.

Corollary 2. *Let* CTR$ *be the stream cipher construction above instantiated with* $\mathrm{KS}_{\mathrm{CTR}}$*. Then,* $\widehat{\mathrm{CTR\$}}$ *is* IND$-CPA*-secure under subversion given that the randomness generation and the* $\oplus$ *operation are trusted.*

Thus, IND$-CPA security directly follows from the security of the underlying PRF. While security is preserved, this does not automatically mean that correctness is preserved also: the decryption algorithm is *not* executed in the IND$-CPA security experiment, but is fundamental for correctness. As discussed by Russell, Tang, Yung, and Zhou [38] this would allow for censorship of chosen messages. If we would consider a black box decryption algorithm, perfect correctness is impossible to achieve, as a single input trigger (for example for $C^\star$ the decryption always output a constant value) violates the perfect correctness requirement while highly unlikely to being detected by a watchdog. Nevertheless, as in our construction the adversary only provides an implementation of the underlying PRF, we see that our construction automatically satisfies *perfect* correctness.

Theorem 8. *The specification* $\widehat{\mathsf{SE}_{\mathrm{KS}}}$ *is perfectly correct.*

Proof. Correctness follow from the "canonical decryption"[7] of the stream cipher as the same value as during the encryption procedure are computed. Thus, even if $\mathrm{KS}(K, IV)$ deviates from the specification, the subverted output cancels out by the $\oplus$ operation: $\widetilde{\mathsf{Dec}}(K, \widetilde{\mathsf{Enc}}(K, M)) = \widetilde{\mathrm{KS}}(K, IV) \oplus \widetilde{\mathrm{KS}}(K, IV) \oplus M = M$.

Note that previous works [38] also achieved correctness, but tolerated a negligible decryption error. This is because the authors viewed the decryption algorithm as an algorithm with a public input distribution and can check consistency with the specification up to a negligible failure probability. Due to more fine grained access to the decryption procedure, we can guarantee *perfect* correctness.

6 Authenticated Encryption

We now see that the classical Encrypt-then-MAC approach grants us subversion-resilient authenticated encryption, given subversion-resilient building blocks. An authenticated encryption scheme works on a keyspace $\mathcal{K}$, message space $\mathcal{M}$, data space $\mathcal{D}$, and ciphertext space $\mathcal{C}$. As usual, we have $\mathcal{K} = \bigcup_{\lambda\in\mathbb{N}} \mathcal{K}_\lambda$, $\mathcal{M} = \mathcal{M}_{\lambda\in\mathbb{N}}$, $\mathcal{D} = \bigcup_{\lambda\in\mathbb{N}} \mathcal{D}_\lambda$, and $\mathcal{C} = \bigcup_{\lambda\in\mathbb{N}} \mathcal{C}_\lambda$. In the following, we assume that the message space $\mathcal{M}_\lambda$ and the ciphertext space $\mathcal{C}_\lambda$ contain a special symbol $\perp$.

Definition 7. *We call a triple* $\mathsf{AD} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *a* symmetric encryption scheme with associated data *for key space $\mathcal{K}$, message space $\mathcal{M}$, data space $\mathcal{D}$, and ciphertext space $\mathcal{C}$. The randomized* key generation algorithm KGen *outputs a key* $K \xleftarrow{\$} \mathcal{K}_\lambda$ *upon input the security parameter* 1^λ*. The randomized* encryption algorithm Enc *is given a key $K \in \mathcal{K}_\lambda$, a message $M \in \mathcal{M}_\lambda$, associated data $D \in \mathcal{D}_\lambda$ and returns a ciphertext $C \in \mathcal{C}_\lambda$. The deterministic* decryption algorithm Dec *is given a key $K \in \mathcal{K}_\lambda$, a ciphertext $C \in \mathcal{C}_\lambda$, associated data $D \in \mathcal{D}_\lambda$ and returns a message $M \in \mathcal{M}_\lambda$.*

[7] By this we mean recomputing a value and applying it via $\oplus$ to the ciphertext in order to decrypt.

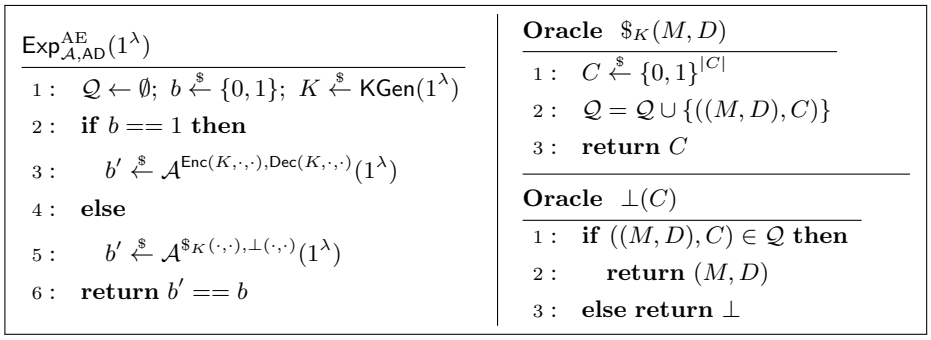

Fig. 6: The security experiment for AEAD. The oracle $\mathsf{Enc}(K,\cdot,\cdot)$ expects for $K \in \mathcal{K}_\lambda$ a message $M \in \mathcal{M}_\lambda$ and data $D \in \mathcal{D}_\lambda$, while $\mathsf{Dec}(K,\cdot,\cdot)$ expects a ciphertext $C \in \mathcal{C}_\lambda$ and data $D \in \mathcal{D}_\lambda$.

A *symmetric encryption scheme with associated data* $\mathsf{AD} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be *perfectly correct* if for all $K \in \mathcal{K}_\lambda$, $M \in \mathcal{M}_\lambda$, and $D \in \mathcal{D}_\lambda$ it holds $\mathsf{Dec}(K, \mathsf{Enc}(K,(M,D)),D) = (M,D)$.

Definition 8. *Let* AD *be a symmetric encryption scheme with associated data and let* $\mathsf{Exp}^{\mathrm{AE}}_{\mathcal{A},\mathsf{AD}}(1^\lambda) = 1$ *be defined as shown in Fig. 6. We define* $\mathsf{Adv}^{\mathrm{AE}}_{\mathcal{A},\mathsf{AD}}(1^\lambda) := \Pr[\mathsf{Exp}^{\mathrm{AE}}_{\mathcal{A},\mathsf{AD}}(1^\lambda) = 1]$ *and say that* AD *is* AE*-secure if* $\mathsf{Adv}^{\mathrm{AE}}_{\mathcal{A},\mathsf{AD}}(1^\lambda)$ *is negligible for all ppt adversaries* $\mathcal{A}$*.*

Usually, the experiment requires that the adversary does not ask for decryption of outputs of the encryption oracle. For $b = 0$ the experiment cannot output a message, since it is just given a random string. As this would trivially break security (and the adversary already knows the answer to the query), this case is excluded in most works. Hence, there is no need to manage the set Q explicitly. In the context of subversion this approach unfortunately also rules out a natural, challenging attack. An adversary could provide an implementation for the decryption algorithm, which instead of a certain message $M^\star$ simply outputs the secret key, allowing the adversary to trivially break security. If the decryption algorithm is modeled as a black box, this attack seems unavoidable since a watchdog cannot efficiently detect the trigger message $M^\star$. Even amalgamation of several subverted components cannot prevent this attack if no trusted component is ever used, as an input trigger for the "first" component can again directly lead to input trigger of the next subverted component and so on. Thus, some sort of trusted operation is necessary in order to defend against these kind of attacks. As seen by our construction of stream ciphers with "canonical decryption" in combination with a trusted XOR operation, perfect correctness is guaranteed. In order to include this attack in our model, we change the behavior of the oracles in the experiment for $b = 0$ by introducing book keeping of the queries. This allows the adversary to ask for decryptions of encryption queries without trivially breaking security. An interesting side effect of this definition is

that every scheme satisfying this security notion also *needs* to satisfy correctness (although only up to negligible failure probability), as the adversary would be able to distinguish the two worlds of the experiment otherwise.

6.1 Achieving Subversion-resilience via Encrypt-then-MAC

The classical way to construct authenticated encryption relies on the use of a MAC that is applied *after* encryption. Decryption then first verifies the MAC and afterwards performs the underlying decryption algorithm. Due to the strong unforgeability of the MAC, an adversary cannot make use of decryption and the security experiment reduces to IND$-CPA-security. This is useful for achieving subversion-resilience, as a symmetric encryption which is IND$-CPA secure under subversion does not give any security guarantees for the decryption algorithm and avoids input trigger attacks. While our construction of stream ciphers guarantees correctness under random inputs, this cannot be guaranteed if the adversary can freely choose the inputs for the decryption algorithm. The reason for this again are input trigger attacks, since no watchdog knows the distribution of the queries made by the adversary. The Encrypt-then-MAC approach with subversion-resilient MAC ensures that input trigger attacks are ruled out even if the verify algorithm is subverted. In combination with a subversion-resilient encryption scheme the adversary needs forge a MAC to obtain a decryption of a ciphertext that it did not obtain as an output of the encrypt oracle.

Encrypt-then-MAC. Let $\mathsf{MAC} = (\mathsf{Gen}, \mathsf{Tag}, \mathsf{Vf})$ be a MAC and $\mathsf{SE}_\$ = (\mathsf{KGen}_\$, \mathsf{Enc}_\$, \mathsf{Dec}_\$)$ be a symmetric encryption scheme. Then, the symmetric encryption scheme with associated data $\mathsf{AD} = \mathsf{AD}_{\mathsf{SE}_\$,\mathsf{MAC}} = (\mathsf{KGen}_{\mathsf{AD}}, \mathsf{Enc}_{\mathsf{AD}}, \mathsf{Dec}_{\mathsf{AD}})$ is defined as follows: The key generation algorithm $\mathsf{KGen}_{\mathsf{AD}}(1^\lambda)$ first obtains a key $K_{\mathsf{MAC}} \xleftarrow{\$} \mathsf{KGen}_{\mathsf{MAC}}(1^\lambda)$, another key $K_\$ \xleftarrow{\$} \mathsf{KGen}_\$(1^\lambda)$ and outputs $K = (K_{\mathsf{MAC}}, K_\$)$. The encryption algorithm $\mathsf{Enc}_{\mathsf{AD}}(K = (K_{\mathsf{MAC}}, K_\$), M, D)$ first calls the underlying encryption algorithm $\mathsf{Enc}_\$$ to compute a ciphertext $C_\$ \xleftarrow{\$} \mathsf{Enc}_\$(K_\$, M)$, then produces a tag $T \xleftarrow{\$} \mathsf{Tag}(K_{\mathsf{MAC}}, C_\$ \mid\mid D)$, and outputs $C = (C_\$, T)$. The decryption algorithm $\mathsf{Dec}_{\mathsf{AD}}(K = (K_{\mathsf{MAC}}, K_\$), C = (C_\$, T), D)$ first verifies the MAC of C by computing $b \xleftarrow{\$} \mathsf{Vf}(K_{\mathsf{MAC}}, C_\$ \mid\mid D, T)$. If $b = 0$ (i.e. the verification failed), it returns $\perp$. Else, it computes $M \xleftarrow{\$} \mathsf{Dec}_\$(K_\$, C_\$)$ and returns M. The straightforward reduction to the security of MAC and $\mathsf{SE}_\$$ can be used to obtain the following well-known theorem.

Theorem 9 ([9]). *If* MAC *is* SUF-CMA*-secure and* $\mathsf{SE}_\$$ *is* IND$-CPA*-secure, then* AD *is* AE*-secure.*

Correctness of SE is directly inherited by AD as the MAC does not change the correctness of the underlying encryption scheme.

Theorem 10. *If* $\widehat{\mathsf{SE}_\$}$ *is perfectly correct, then* $\widehat{\mathsf{AD}}$ *is perfectly correct.*

We prove that the Encrypt-then-MAC approach is indeed sound if both the encryption scheme as well as the MAC are subversion-resilient, following the proof idea of Bellare and Namprempre [8, 9].

Theorem 11. *If* $\widehat{\mathsf{SE}}$ *is* IND$-CPA*-secure under subversion,* $\widehat{\mathsf{MAC}}$ *is* SUF-CMA*-secure under subversion and* $\widehat{\mathsf{SE}}$ *is correct, then* $\widehat{\mathsf{AD}}$ *is* AE*-secure under subversion and correct.*

Proof (Sketch). The watchdog for $\widehat{\mathsf{AD}}$ runs the watchdog for $\widehat{\mathsf{SE}}$ and $\widehat{\mathsf{MAC}}$. Note that there is no need for testing the components in combination, as both allow arbitrary input distributions while being subverted and are executed independently. Assuming that the watchdog does not detect subversion, we can follow the proof of Bellare and Namprempre [8,9] for the Encrypt-then-MAC approach. The main difference is to base the security on the subversion-resilience of the building blocks rather then the "classical" security properties. However, these only differ by the preceded check by the watchdog and that usage of the implementation instead of the specifications. Thus, in the same way IND$-CPA security in the classical setting does grant any security guarantees for the decryption algorithm, we also do not need to immunize the subverted decryption algorithm. First, we replace the decryption algorithm for $b = 1$ with an oracle that always answers with the $\perp$ symbol, except if the input was obtained by querying the encrypt oracle. If the output of the encrypt oracle was handed to the decryption oracle, the adversary always obtains correct answer, either by the correctness of the encryption scheme for $b = 1$ or due to the book keeping if $b = 0$. Now assume C was not obtained via the encrypt oracle. In case C is malformed and the decryption oracle returns a $\perp$ symbol, the adversary cannot distinguish this from an oracle which always returns the $\perp$ symbol. The last case that remains is that C is a valid ciphertext and such that the encrypted message (M, D) was *not* issued to the encryption oracle. This case is again not possible, since any adversary reaching this case would have successfully forged a tag for (M, D), thus breaking subversion resilience of $\widehat{\mathsf{MAC}}$. Therefore we can "disable" the decryption oracle. Then however, we can base security entirely on the subversion-resilience of $\widehat{\mathsf{SE}}$, as the adversary can only make effective use of the encrypt oracle. We replace the encrypt oracle with an oracle returning random bits. This change is again indistinguishable by the subversion-resilience of $\widehat{\mathsf{SE}}$. We changed all oracles so that for both choices of b the adversary is given access to the same oracles, thus being unable to win the experiment apart from guessing the bit b.

Acknowledgements Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant agreement 802823.

References

1. Armour, M., Poettering, B.: Substitution attacks against message authentication. IACR Trans. Symm. Cryptol. **2019**(3), 152–168 (2019). https://doi.org/10.13154/tosc.v2019.i3.152-168

2. Armour, M., Poettering, B.: Subverting decryption in AEAD. In: Albrecht, M. (ed.) 17th IMA International Conference on Cryptography and Coding. LNCS, vol. 11929, pp. 22–41. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-35199-1_2
3. Ateniese, G., Francati, D., Magri, B., Venturi, D.: Public immunization against complete subversion without random oracles. In: Deng, R.H., Gauthier-Umaña, V., Ochoa, M., Yung, M. (eds.) ACNS 19. LNCS, vol. 11464, pp. 465–485. Springer, Heidelberg (Jun 2019). https://doi.org/10.1007/978-3-030-21568-2_23
4. Ateniese, G., Magri, B., Venturi, D.: Subversion-resilient signature schemes. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 2015. pp. 364–375. ACM Press (Oct 2015). https://doi.org/10.1145/2810103.2813635
5. Bauer, B., Farshim, P., Mazaheri, S.: Combiners for backdoored random oracles. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 272–302. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/978-3-319-96881-0_10
6. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: 38th FOCS. pp. 394–403. IEEE Computer Society Press (Oct 1997). https://doi.org/10.1109/SFCS.1997.646128
7. Bellare, M., Jaeger, J., Kane, D.: Mass-surveillance without the state: Strongly undetectable algorithm-substitution attacks. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 2015. pp. 1431–1440. ACM Press (Oct 2015). https://doi.org/10.1145/2810103.2813681
8. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Heidelberg (Dec 2000). https://doi.org/10.1007/3-540-44448-3_41
9. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. Journal of Cryptology **21**(4), 469–491 (Oct 2008). https://doi.org/10.1007/s00145-008-9026-x
10. Bellare, M., Paterson, K.G., Rogaway, P.: Security of symmetric encryption against mass surveillance. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 1–19. Springer, Heidelberg (Aug 2014). https://doi.org/10.1007/978-3-662-44371-2_1
11. Bemmann, P., Chen, R., Jager, T.: Subversion-resilient public key encryption with practical watchdogs. In: Garay, J. (ed.) PKC 2021, Part I. LNCS, vol. 12710, pp. 627–658. Springer, Heidelberg (May 2021). https://doi.org/10.1007/978-3-030-75245-3_23
12. Berndt, S., Liskiewicz, M.: Algorithm substitution attacks from a steganographic perspective. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 1649–1660. ACM Press (Oct / Nov 2017). https://doi.org/10.1145/3133956.3133981
13. Bhattacharyya, R., Nandi, M., Raychaudhuri, A.: Crooked indifferentiability of enveloped XOR revisited. In: INDOCRYPT. Lecture Notes in Computer Science, vol. 13143, pp. 73–92. Springer (2021)
14. Bogdanov, A., Rosen, A.: Pseudorandom functions: Three decades later. In: Tutorials on the Foundations of Cryptography, pp. 79–158. Springer International Publishing (2017)
15. Bossuat, A., Bultel, X., Fouque, P.A., Onete, C., van der Merwe, T.: Designing reverse firewalls for the real world. In: Chen, L., Li, N., Liang, K., Schneider, S.A. (eds.) ESORICS 2020, Part I. LNCS, vol. 12308, pp. 193–213. Springer, Heidelberg (Sep 2020). https://doi.org/10.1007/978-3-030-58951-6_10

16. Brown, D.R.L., Gjøsteen, K.: A security analysis of the NIST SP 800–90 elliptic curve random number generator. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 466–481. Springer, Heidelberg (Aug 2007). https://doi.org/10.1007/978-3-540-74143-5_26
17. Chakraborty, S., Dziembowski, S., Nielsen, J.B.: Reverse firewalls for actively secure MPCs. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 732–762. Springer, Heidelberg (Aug 2020). https://doi.org/10.1007/978-3-030-56880-1_26
18. Chakraborty, S., Magri, B., Nielsen, J.B., Venturi, D.: Universally composable subversion-resilient cryptography. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part I. LNCS, vol. 13275, pp. 272–302. Springer, Heidelberg (May / Jun 2022). https://doi.org/10.1007/978-3-031-06944-4_10
19. Checkoway, S., Maskiewicz, J., Garman, C., Fried, J., Cohney, S., Green, M., Heninger, N., Weinmann, R.P., Rescorla, E., Shacham, H.: A systematic analysis of the juniper dual EC incident. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016. pp. 468–479. ACM Press (Oct 2016). https://doi.org/10.1145/2976749.2978395
20. Chen, R., Mu, Y., Yang, G., Susilo, W., Guo, F., Zhang, M.: Cryptographic reverse firewall via malleable smooth projective hash functions. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part I. LNCS, vol. 10031, pp. 844–876. Springer, Heidelberg (Dec 2016). https://doi.org/10.1007/978-3-662-53887-6_31
21. Chow, S.S.M., Russell, A., Tang, Q., Yung, M., Zhao, Y., Zhou, H.S.: Let a non-barking watchdog bite: Cliptographic signatures with an offline watchdog. In: Lin, D., Sako, K. (eds.) PKC 2019, Part I. LNCS, vol. 11442, pp. 221–251. Springer, Heidelberg (Apr 2019). https://doi.org/10.1007/978-3-030-17253-4_8
22. Degabriele, J.P., Farshim, P., Poettering, B.: A more cautious approach to security against mass surveillance. In: Leander, G. (ed.) FSE 2015. LNCS, vol. 9054, pp. 579–598. Springer, Heidelberg (Mar 2015). https://doi.org/10.1007/978-3-662-48116-5_28
23. Dodis, Y., Farshim, P., Mazaheri, S., Tessaro, S.: Towards defeating backdoored random oracles: Indifferentiability with bounded adaptivity. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part III. LNCS, vol. 12552, pp. 241–273. Springer, Heidelberg (Nov 2020). https://doi.org/10.1007/978-3-030-64381-2_9
24. Dodis, Y., Ganesh, C., Golovnev, A., Juels, A., Ristenpart, T.: A formal treatment of backdoored pseudorandom generators. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 101–126. Springer, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46800-5_5
25. Dodis, Y., Mironov, I., Stephens-Davidowitz, N.: Message transmission with reverse firewalls—secure communication on corrupted machines. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 341–372. Springer, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53018-4_13
26. Fischlin, M., Janson, C., Mazaheri, S.: Backdoored hash functions: Immunizing HMAC and HKDF. In: Chong, S., Delaune, S. (eds.) CSF 2018 Computer Security Foundations Symposium. pp. 105–118. IEEE Computer Society Press (2018). https://doi.org/10.1109/CSF.2018.00015
27. Fischlin, M., Mazaheri, S.: Self-guarding cryptographic protocols against algorithm substitution attacks. In: Chong, S., Delaune, S. (eds.) CSF 2018 Computer Security Foundations Symposium. pp. 76–90. IEEE Computer Society Press (2018). https://doi.org/10.1109/CSF.2018.00013

28. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions (extended abstract). In: 25th FOCS. pp. 464–479. IEEE Computer Society Press (Oct 1984). https://doi.org/10.1109/SFCS.1984.715949
29. Goldreich, O., Goldwasser, S., Micali, S.: On the cryptographic applications of random functions. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO'84. LNCS, vol. 196, pp. 276–288. Springer, Heidelberg (Aug 1984)
30. Maurer, U.M., Sjödin, J.: A fast and key-efficient reduction of chosen-ciphertext to known-plaintext security. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 498–516. Springer, Heidelberg (May 2007). https://doi.org/10.1007/978-3-540-72540-4_29
31. Maurer, U.M., Tessaro, S.: Basing PRFs on constant-query weak PRFs: Minimizing assumptions for efficient symmetric cryptography. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 161–178. Springer, Heidelberg (Dec 2008). https://doi.org/10.1007/978-3-540-89255-7_11
32. Mironov, I., Stephens-Davidowitz, N.: Cryptographic reverse firewalls. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 657–686. Springer, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46803-6_22
33. Naor, M., Reingold, O.: Synthesizers and their application to the parallel construction of pseudo-random functions. J. Comput. Syst. Sci. **58**(2), 336–375 (1999)
34. Perlroth, N., Larson, J., Shane, S.: Secret documents reveal nsa campaign against encryption (2013), `https://archive.nytimes.com/www.nytimes.com/interactive/2013/09/05/us/documents-reveal-nsa-campaign-against-encryption.html`
35. Rogaway, P.: Authenticated-encryption with associated-data. In: Atluri, V. (ed.) ACM CCS 2002. pp. 98–107. ACM Press (Nov 2002). https://doi.org/10.1145/586110.586125
36. Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: A block-cipher mode of operation for efficient authenticated encryption. In: Reiter, M.K., Samarati, P. (eds.) ACM CCS 2001. pp. 196–205. ACM Press (Nov 2001). https://doi.org/10.1145/501983.502011
37. Russell, A., Tang, Q., Yung, M., Zhou, H.S.: Cliptography: Clipping the power of kleptographic attacks. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 34–64. Springer, Heidelberg (Dec 2016). https://doi.org/10.1007/978-3-662-53890-6_2
38. Russell, A., Tang, Q., Yung, M., Zhou, H.S.: Generic semantic security against a kleptographic adversary. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 907–922. ACM Press (Oct / Nov 2017). https://doi.org/10.1145/3133956.3133993
39. Russell, A., Tang, Q., Yung, M., Zhou, H.S.: Correcting subverted random oracles. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 241–271. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/978-3-319-96881-0_9
40. Shumow, D., Ferguson, N.: On the possibility of a back door in the nist sp800-90 dual ec prng (2007), `http://rump2007.cr.yp.to/15-shumow.pdf`, cRYPTO 2007 Rump Session
41. Wegman, M.N., Carter, L.: New hash functions and their use in authentication and set equality. Journal of Computer and System Sciences **22**(3), 265–279 (1981). https://doi.org/10.1016/0022-0000(81)90033-7, `https://doi.org/10.1016/0022-0000(81)90033-7`

42. Young, A., Yung, M.: The dark side of "black-box" cryptography, or: Should we trust capstone? In: Koblitz, N. (ed.) CRYPTO'96. LNCS, vol. 1109, pp. 89–103. Springer, Heidelberg (Aug 1996). https://doi.org/10.1007/3-540-68697-5_8