Subversion-Resilient Signatures without Random Oracles

Pascal Bemmann[1], Sebastian Berndt[2], Rongmao Chen[3]

[1] Bergische Universität Wuppertal, Wuppertal, Germany.
bemmann@uni-wuppertal.de
[2] Universität zu Lübeck, Lübeck, Germany. s.berndt@uni-luebeck.de
[3] National University of Defense Technology, Changsha, China.
chromao@nudt.edu.cn

Abstract. In the aftermath of the Snowden revelations in 2013, concerns about the integrity and security of cryptographic systems have grown significantly. As adversaries with substantial resources might attempt to subvert cryptographic algorithms and undermine their intended security guarantees, the need for subversion-resilient cryptography has become paramount. Security properties are preserved in subversion-resilient schemes, even if the adversary implements the scheme used in the security experiment. This paper addresses this pressing concern by introducing novel constructions of subversion-resilient signatures and hash functions while proving the subversion-resilience of existing cryptographic primitives. Our main contribution is the first construction of subversion-resilient signatures under *complete* subversion in the offline watchdog model (with trusted amalgamation) *without* relying on random oracles. We demonstrate that one-way permutations naturally yield subversion-resilient one-way functions, thereby enabling us to establish the subversion-resilience of Lamport signatures, assuming a trusted comparison is available. Additionally, we develop subversion-resilient target-collision-resistant hash functions using a trusted XOR. By leveraging this approach, we expand the arsenal of cryptographic tools that can withstand potential subversion attacks. Our research builds upon previous work in the offline watchdog model with trusted amalgamation (Russell et al. ASIACRYPT'16) and subversion-resilient pseudo-random functions (Bemmann et al. ACNS'23), culminating in the formal proof of subversion-resilience for the classical Naor-Yung signature construction.

Keywords: Subversion, Digital Signatures, Public-key Cryptography

1 Introduction

Subversion attacks have garnered increasing attention from the cryptography research community in recent years. In a subversion setting, attackers can tamper with or even take control of the implementation of cryptographic algorithms to leak secrets covertly, thereby weakening or breaking the security of

the cryptosystems. For a long time, subversion attacks, dating back to the notion of kleptography by Young and Yung in the 1990s [31,32], were thought to be unrealistic and far-fetched by some cryptographers. However, the Snowden revelations in 2013 debunked this belief and exposed that subversion attacks were among the primary approaches certain law enforcement agencies employed to achieve mass surveillance. Particularly, it is reported that these agencies could intentionally "insert vulnerabilities into commercial encryption systems, IT systems, networks, and endpoint communications devices used by targets" [25]. As a result, numerous researchers have demonstrated the theoretical feasibility and potential dangers of subversion attacks against various cryptographic primitives [1,4,5,8,9,11,14], and as a specific case against digital signatures [2,3,17,20,30].

Consequently, the research community has devoted extensive efforts exploring effective countermeasures and designing subversion-resilient cryptographic primitives. In recent discussions on the standardization of post-quantum cryptography, specific treatments have been considered to prevent subversion attacks [26], highlighting the practical significance of this area of research. There is no hope for security when attackers can arbitrarily tamper with cryptographic implementations. For instance, a subverted encryption algorithm could always reveal the secret key, regardless of the input plaintext intended to be encrypted. Also, a subverted signature verification algorithm might always return `valid` when it takes as input a specific signature or message (e.g., a certain hard-coded string). Thus, as it turned out in the literature, achieving meaningful security in the subversion setting requires reliance on additional assumptions such as architectural requirements [6,7,13,27,28,29] or trusted components [2,10,12,15,16,22]. We remark that all these approaches for subversion resilience have their plausibility and thus are generally incomparable and may be useful in different application contexts. The details of these models, with associated commentary about their relevance to practice, will be provided in Section 1.4.

The Watchdog Model. In this work, we mainly consider the model introduced by Russell et al. [27] at ASIACRYPT 2016. Precisely, they formalized the so-called *watchdog model*, which has become one of the most prominent approaches for subversion-resilient cryptography in recent years. The rationale behind their consideration is that in the subversion setting, it is highly desirable for the attackers, often referred to as "big brother", to conceal their attacks. The watchdog model allows the adversary to supply potentially subverted implementations of cryptographic algorithms, while an efficient "watchdog" verifies these implementations against their corresponding specifications. Depending on how the watchdog performs testing of cryptographic algorithms, there are *offline watchdogs* which only perform a one-time check of the supplied implementations before deployment, and *online watchdogs* that could fully access the real-time transcripts of cryptographic algorithms. Note that achieving subversion resilience only using offline watchdogs is often preferable from a practical perspective, as only a "one-time" check is required instead of continuous monitoring.

Cryptographic designs in the watchdog model are often based on an architectural assumption called the *split-program* methodology and the *trusted amalgamation* assumption that is required to be as simple as possible. In such a setting, each algorithm is divided into several functional components, which the watchdog could separately test and then amalgamated into the complete algorithm via the trusted (not subverted) amalgamation function (e.g., trusting the computing base). A cryptographic scheme is said to be subversion-resilient in the watchdog model if *either* a watchdog exists that can effectively detect the subverted components *or* the security of the composed cryptographic scheme still holds even if the underlying components are subverted. By leveraging such a model, various subversion-resilient primitives have been proposed, including one-way-permutations [27], pseudorandom generators [27], randomness generators [7,27], public-key encryption schemes [7,28], authenticated encryption [6], random oracles [29], and signature schemes [13,27].

1.1 Subversion-Resilient Signatures with Watchdogs

Our work primarily focuses on digital signature schemes and aims to advance the construction of subversion-resilient signature schemes within the watchdog model. We provide an overview of the current state-of-the-art and present the question that motivates our work.

Online Watchdogs. Russell et al. proposed the first subversion-resilient signature scheme in the online watchdog model with random oracles [27]. In particular, they consider the *complete-subversion* setting where all cryptographic algorithms — including key generation and verification algorithms — are subject to subversion attacks. At the core of their design is a slight variant of a full domain hash scheme where the message is hashed together with the public key. Precisely, such a modification enables provable security by rendering any random oracle queries made before the implementations provided (by the adversary) useless, as the public key is generated freshly after the adversary commits to the implementations. More generally, they pointed out that in their definitional framework, it is impossible to construct a subversion-resistant signature scheme with just an offline watchdog, even if only the signing algorithm is subverted. Note that in the same work [27], Russell et al. also proposed subversion-resilient one-way permutations. They considered a stronger security setting, where the adversary can choose the function index (pp in our definition). This, in turn, makes the use of random oracles necessary. We will see that this stronger notion is not needed to construct subversion-resilient signatures, and we can thus remove the random oracle dependency.

Offline Watchdogs. In [13], Chow et al. improved Russell et al.'s construction by presenting two schemes with offline watchdogs. They bypassed Russell et al.'s impossibility [27] by using a more fine-grained split-program model adopted in [28] for a semantically secure encryption scheme. The first construction is without random oracles and only considers the partial-subversion model where key generation and signing are subverted while the verification algorithm

is not. By adopting the domain-separation technique and the one-time random tag structure, they extended the idea by Russell et al. (originally for an encryption algorithm) to the context of a signature scheme. They also proposed another subversion-resilient signature scheme in the complete-subversion model but with random oracles. Their main idea is inspired by the correction of subverted random oracles due to Russell et al. [29].

Motivation. Despite significant progress made in constructing subversion-resilient signature schemes in the watchdog model, the state-of-the-art constructions (using offline watchdogs) require random oracles for achieving security in the complete-subversion model, and if without random oracles, only security in the partial-subversion model (where the verification algorithm is not subverted) is achieved. This motivates us to ask the following question.

Is it possible to design a subversion-resilient signature without random oracles in the complete-subversion model with an offline watchdog?

While being attractive for building security, the offline watchdog model has inherent limitations. For example, an offline watchdog can not defend against stateful subversions[4] such as time bomb attacks, which only become active when the underlying algorithm is at some specific state. Nevertheless, since it is up to a practitioner to trade off security and performance constraints, we hope our improvements in subversion-resilient signature schemes (using watchdogs) could help practitioners make well-informed decisions regarding their suitability for specific applications.

1.2 Technical Challenges

The difficulty of designing signature schemes secure against complete subversion mainly lies in the fact that it is challenging to restore the security of the subverted verification algorithm. The main reason is twofold. The first reason is the existence of *input trigger attacks*, where an adversary prepares a special signature $\tilde{\sigma}$, which the verification algorithm accepts for all messages. An attacker can randomly choose $\tilde{\sigma}$ and hard-code it into the subverted implementation. A black-box polynomial time watchdog now has only a negligible chance to detect this, while the attacker can trivially break security. Intuitively, this attack is possible as the attacker chooses the *input distribution* to the verification algorithm which is not publicly available to the watchdog (unless we assume some strategy by the adversary, see subversion under random messages attacks for comparison [2]). Thus, similar to previous works [6], more fine-grained access to the verification algorithm seems necessary. However, the situation in a setting with asymmetric keys is substantially more difficult than in the symmetric setting studied by Bemmann et al. [6], as the attacker sees the public verification

[4] Assuming universal watchdogs that do not depend on the adversary, which is the class of watchdogs we aim for in this work. For a deeper discussion on stateful subversion consider [27].

key and can thus use this knowledge. A common technique to develop signature schemes based on symmetric primitives [21] is the use of a collision-resistant hash function. But, due to the structure of the security experiment of collision-resistant hash functions, they are also highly vulnerable to input trigger attacks. Previous constructions thus needed heavy machinery such as the random oracle model [13,29], sophisticated online watchdogs [2,29], or a trusted initialization phase [16]. Approaches using a collision-resistant hash function in the standard model with an offline watchdog thus seem somewhat futile.

Both problems around input triggers described above rely on the fact that the adversary can prepare its implementation with input triggers as the challenge goes through the (possibly) subverted implementation. In our constructions, we will utilize primitives, where the adversary needs to commit to its implementation *before* a challenge is chosen. This way, the adversary can freely choose the inputs to the primitives, but since the implementation is set up before the challenge, it is hard for the adversary to adapt. As described above, Russell et al. [29] also used a similar idea but needed the random oracle model. We can circumvent the need for the random oracle model by choosing our primitives carefully and revisiting classical results in a subversion setting. As will be shown later, in this work we use a classic result that target-collision-resistant (TCR) hash functions are sufficient to build digital signatures.To break target-collision-resistance, the adversary first commits to an input, then the hash function is specified, and afterward, the adversary tries to find a collision. The task thus becomes to construct target-collision-resistant hash functions in a subversion-resilient manner.

1.3 Our Contributions

Our work provides an affirmative answer to the above question. The main contributions of this work can be summarized as follows:

- We show that one-way permutations (OWPs) are subversion-resilient one-way functions in the offline watchdog model, by taking advantage of the order of events in the security experiment. Russell et al. also showed how to construct subversion-resilient OWPs with random oracles [27]. Our construction of OWFs does not rely on random oracles; in our case, we only need the standard version of OWPs.
- Lamport one-time signatures (OTS) are subversion-resilient if built from subversion-resilient OWFs and a trusted comparison;
- We prove that a classical construction to obtain TCR hash functions from a rTCR hash function can be used to obtain subversion-resilient TCR hash function making use of a random blinding value, given the XOR in the construction is part of the trusted amalgamation;
- From subversion-resilient OTS, subversion-resilient target-collision resistant hash functions, and subversion-resilient PRFs, we build subversion-resilient signatures via the classical Naor-Yung [24] construction of digital signatures.

Thus, similar to the work of Chow et al. [13], we allow the watchdog more fine-grained access to the verification algorithm by breaking it down into smaller

building blocks. This, in turn, allows for a similar approach as Bemmann et al. [6] for the case of authenticated encryption. We build signatures from symmetric primitives by revisiting classical results and show that we can construct the necessary building blocks in a subversion-resilient manner. This way, during the verification of the signatures, we recompute symmetric primitives, which allows the watchdog to do meaningful testing. A key insight of our work is that the security of some primitives can be guaranteed if we consider an adversary who has to commit to its implementation before a random challenge is computed. We then see that all the ingredients can be combined to prove the classical Naor-Yung construction for digital signatures to be subversion-resilient. However, while achieving subversion-resilience without random oracles, this comes with the price of decreased efficiency (both in signature size and computational costs) if compared to the state-of-the-art. Nevertheless, as mentioned above, we hope our work could help practitioners make well-informed decisions regarding the suitability of different signature schemes for specific applications. See Figure 1b for an illustration of our overall approach.

1.4 Alternative Models

Several works in the field of cryptography have explored different angles of defense against subversion attacks, resulting in the proposal of various subversion-resilient signature schemes. Although these schemes may be generally incomparable due to the different models and assumptions they are built upon, understanding their differences can provide valuable insights into the landscape of subversion-resilient cryptography. Below we present an overview of current subversion-resilient signature schemes in other models.

Subversion-Resilient Signatures via Reverse Firewalls. In [2], Ateniese et al. showed that signature schemes with unique signatures are subversion-resilient against all subversion attacks that meet the so-called "verifiability condition". This condition essentially requires that signatures produced by the subverted signature algorithm should almost always verify correctly under the target verification key[5]. They adopted the cryptographic reverse firewall (RF) for constructing subversion-resilient signature schemes to relax such a strong requirement. Mironov and Stephens-Davidowitz originally introduced the notion of RF [22], which is assumed to be non-subverted and has access to a reliable source of randomness to re-randomize cryptographic transcripts. In the context of signature schemes, a RF is a (possibly stateful) algorithm that takes a message/signature pair as input and produces an updated signature. The main goal of a RF is to prevent potential covert leakage of sensitive information from subverted signatures. As a general result, Ateniese et al. showed that every re-randomizable signature scheme (including unique signatures as a special case) admits a RF that preserves unforgeability against arbitrary subversion attacks. Such a RF must have self-destruct capability, which means that the RF can publicly check the validity

[5] In [2], only subverted signing algorithms are considered while both key generation and verification algorithms are assumed to be trusted.

of each outgoing message/signature pair before updating the signature. If the RF encounters an invalid pair during this process, it stops processing further queries. The self-destruct capability is essential for the RF to maintain functionality and preserve the unforgeability of the signature scheme simultaneously. One could note that a RF could be viewed as an "active" online watchdog with the additional self-destruct capability. Thus, like the online watchdog model, a RF can defend against stateful subversion, which is inherently not captured by the offline watchdog model.

Subversion-Resilient Signatures via Self-guarding. In [16], Fischlin and Mazaheri proposed a novel defense mechanism called "self-guarding", which could counter stateless subversion of deterministic unforgeable signature schemes. The self-guarding signature scheme introduces a trusted initialization phase during which genuine message-signature pairs are generated for randomly chosen messages. More precisely, a random message denoted as $m_\$$ is signed in the initialization phase, resulting in a signature sample $\sigma_\$$. Later, when signing a message m, the (possibly) subverted signing algorithm is executed twice, once with $m_\$$ and once with the bitwise XOR of m and $[m_\$||\sigma_\$]$, where $||$ represents concatenation. The order of signing these two messages is chosen randomly. If the signing algorithm deviates for one of the two signatures, the subversion is detected with a probability of 1/2. This process can be repeated multiple times with independent key pairs to increase the detection probability to an overwhelming level. From the above, we know that unlike in the reverse firewall model, where a good source of randomness and the self-destruct capability are required, self-guarding schemes rely on a temporary trust phase during initialization. Also, one might think that the initialization phase of self-guarding schemes could be executed by a watchdog, where a specified program could immediately provide a detection solution. However, there is a notable difference: self-guarding schemes involve passing states between the initialization and later phases, whereas watchdogs typically do not forward data to individual users. Another significant distinction between self-guarding and the watchdog model is that self-guarding schemes do not require the subverted algorithm to be available from the start.

The diversity of subversion-resilient signature schemes reflects the complexity of defending against subversion attacks. The choice of models and assumptions is crucial in determining the scheme's effectiveness and practicality. Understanding the strengths and limitations of these subversion-resilient signature schemes is essential for designing secure cryptographic systems in the presence of potential subversion attacks.

2 Model and Preliminaries

In this section, we define the notion of subversion-resilience, and we will use the notations and definitions presented by Bemmann, Berndt, Diemert, Eisenbarth, and Jager [6] which in turn are based on the work of Russell, Tang, Yung, and Zhou [27].

2.1 Notation and Model

Notation. In order to distinguish between the *specification* of a primitive Π and the *implementation* of Π provided by the adversary, we will write $\widehat{\Pi}$ to denote the honest specification of the primitive and $\widetilde{\Pi}$ to denote the implementation of that primitive provided by the adversary. In this paper, we focus on game-based security. Hence, we define security for a cryptographic primitive Π with security objective GOAL by defining a *security experiment* Exp. In a usual experiment, a single party, called the *adversary* $\mathcal{A}$, tries to break the security objective against $\widehat{\Pi}$, and the game is managed by a *challenger*. When dealing with subverted implementation in the watchdog model, we consider a *subversion experiment* ExpSR consisting of three phases:

1. First, the adversary $\mathcal{A}$ provides a subverted implementation $\widetilde{\Pi}$.
2. Then, the *watchdog* WD is run and tries to detect the subversion.
3. Finally, the original security experiment Exp is performed by the adversary, but the subverted implementation is used therein.

To simplify notation, 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 in the final phase. As usual, we denote the security parameter by λ.

Amalgamation. As discussed earlier, there is no black-box way to prevent subversion attacks in the watchdog model, as universal undetectable attacks are known, e.g., by Berndt and Liśkiewicz [8]. To still give security guarantees against subverted implementations, different non-black-box models were presented in the literature. In this work, we follow Russell, Tang, Yung, and Zhou [27] who introduced the *trusted amalgamation model*. Intuitively, this model splits all its components into *subroutines* with a more fine-granular resolution than the usual division into different algorithms. For example, a signature scheme consists of the three algorithms (KGen, Sign, Ver), but each might again consist of several subroutines (which might even be shared among the algorithms). We denote the list of subroutines by $\pi = (\pi_1, \ldots, \pi_n)$. The idea behind the trusted amalgamation model is each of these subroutines π_i might be subverted by the attacker, but the composition of them is performed by a *trusted amalgamation function* Am that is not subverted. Hence, Am is given the list π, producing all the needed algorithms for the primitive. The security experiment is then played on $\widetilde{\Pi} = \mathsf{Am}(\widetilde{\pi})$, where $\widetilde{\pi}$ denotes the list of subverted subroutines provided by the attacker. To provide meaningful security guarantees, one thus aims to make the amalgamation functions as simple as possible to allow automatic or manual verification. Typically, these amalgamation functions only consist of a few XOR operations and equality checks [7,28,6]. To formalize this scenario, we always represent the specification $\widehat{\Pi}$ of a primitive as $\widehat{\Pi} = (\mathsf{Am}, \pi)$. Sometimes, we consider the amalgamation function for a *single* algorithm Π_i of a primitive, denoted by Am_i.

Split-program model. In addition to trusted amalgamation, Russell, Tang, Yung, and Zhou [27] also used the *split-program* methodology. Like modern pro-

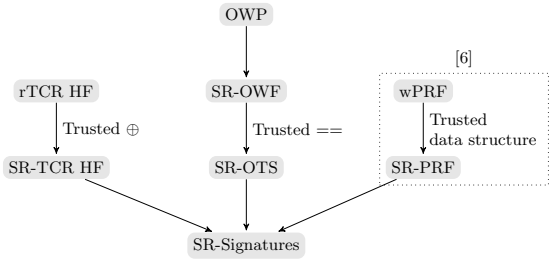
$\mathsf{ExpSR}^{\mathrm{GOAL},\widehat{\Pi}}_{\mathsf{WD},\mathcal{A}}(1^\lambda)$

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

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

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

(a) The experiment for GOAL-security under subversion.



(b) Overview of our construction. Here, SR denotes subversion-resilience, HF denotes hash function, OWP/OWF denotes one-way permutation/-function, PRF denotes pseudo-random function, TCR denotes target collision-resistance, and OTS denotes one-time signature.

Fig. 1

gramming techniques, randomness generation is assumed to be split from a randomized algorithm. The watchdog can then test the randomness generator and the deterministic algorithm individually. We also use this methodology.

2.2 Subversion-Resilience

Now, we describe the notion of a subversion-resilience experiment more formally. As described above, such an experiment consists of three phases, illustrated in Figure 1a. In this paper, we will use both decision and search experiments and thus need to associate a naive win probability $\delta \in [0, 1]$ to each experiment Exp, which will be 0 for search experiments (such as forgery experiments) and 1/2 for decision experiments (such as real-or-random experiments). First, $\mathcal{A}_0$ provides a subverted implementation $\widetilde{\pi}$. Then, the watchdog WD can run the implementation to detect the subversion. Finally, the usual security experiment Exp is run by the adversary $\mathcal{A}_1$ on the subverted implementation $\widetilde{\Pi} = \mathsf{Am}(\widetilde{\pi})$. The watchdog outputs 1 if it detects a subversion. To formalize subversion-resilience, consider the next definition and the corresponding security experiment shown in Figure 1a.

Definition 1. *A specification of a primitive $\widehat{\Pi} = (\mathsf{Am}, \pi)$ is* GOAL-subversion-resilient in the offline watchdog model with trusted amalgamation *if one can efficiently construct a ppt watchdog algorithm* WD *such that for any ppt adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ it holds that $\mathsf{AdvSR}^{\mathrm{GOAL},\widehat{\Pi}}_{\mathcal{A}}(1^\lambda, \delta)$ is negligible or $\mathsf{Det}_{\mathsf{WD},\mathcal{A}}(1^\lambda)$ 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 Figure 1a, with $\delta \in \{0, \frac{1}{2}\}$ indicating whether a search or a decision problem is considered.*

Note that GOAL-subversion-resilience implies GOAL security as the above definition also has to hold for the adversary which outputs the specification as its implementation. For the sake of readability, we will call primitives simply *subversion-resilient*-GOAL with the understanding that they fulfill Definition 1.

2.3 Achieving Subversion-Resilience

A quite simple but instrumental observation was formalized by Russell, Tang, Yung, and Zhou [27]: If the inputs to a deterministic algorithm made by the adversary follow a *public* distribution, the watchdog can make queries also following this distribution. Hence, if the subverted implementation deviates from the specification with probability δ on such input distributions, a polynomial time watchdog can detect the presence of the subversion with probability at least δ. Hence, to have a negligible detection rate by the watchdog, the probability of deviating from the specification must also be very low.

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* $X_\lambda^1, \ldots, X_\lambda^k$ *are defined respectively. If there exists a* $j \in [k]$ *such that* $\Pr[\widetilde{\pi}_j(x) \neq \widehat{\pi}_j(x) : x \xleftarrow{\$} X_\lambda^j] = \delta$*, this can be detected by a ppt offline watchdog with probability at least* δ*.*

This lemma will be used to argue for the subversion-resilience of one-way functions (Section 3) and hash functions (Section 4).

2.4 Assumptions

We make several assumptions throughout this work which we shortly summarize here to enable a quick overview. First, we only consider stateless subversion to rule out the aforementioned time bomb attacks. Second, our results are in the split-program model with trusted amalgamation [27]. Third, since subversion-resilient randomness generators have been already shown to be achievable without using random oracles [6,7], we will simply assume uniform random coins are available in our constructions to simplify notation and point to the mentioned prior work for more details. In particular, in our constructions, all key generation algorithms are assumed to be subversion-resilient, as we could decouple them (in the split-program model) into a randomness generation algorithm and a deterministic algorithm that takes as input the random coins and outputs the key. Note that an offline watchdog could effectively detect the later component as its input is drawn from a public distribution. In addition to these architectural assumptions, our amalgamation will use a trusted XOR and a trusted comparison.

2.5 Pseudorandom Functions

For our signature scheme to be stateless, we also utilize PRFs. Bemmann et al. [6] showed how to construct subversion-resilient PRFs from weak PRFs based

on the classical Naor-Reingold construction for PRFs [23]. We thus only state their definitions and results in a summarized form, point to their work for more details, and assume subversion-resilient PRFs can be constructed and used.

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 $\mathcal{R}$. Let us recall the standard definition of PRFs.

Definition 2 ([6]). *Let $\mathsf{Exp}^{PR}_{\mathcal{A},F}$ be defined as shown in Figure 2. We define*

$$\mathsf{Adv}^{PR}_{\mathcal{A},F}(1^\lambda) := |\Pr[\mathsf{Exp}^{PR}_{\mathcal{A},F}(1^\lambda) = 1] - 1/2|.$$

We say that F is pseudorandom *if $\mathsf{Adv}^{\mathsf{PR}}_{\mathcal{A},F}(1^\lambda)$ is negligible for all ppt adversaries $\mathcal{A}$.*

$\mathsf{Exp}^{\mathsf{PR}}_{\mathcal{A},F}(1^\lambda)$

$b \xleftarrow{\$} \{0,1\};\ K \xleftarrow{\$} \mathcal{K}_\lambda$

if $b = 1$ **then** $b' \leftarrow \mathcal{A}^{F(K,\cdot)}(1^\lambda)$

else $g \xleftarrow{\$} \mathsf{Func}(\mathcal{D}_\lambda, \mathcal{R}_\lambda);\ b' \leftarrow \mathcal{A}^{g(\cdot)}(1^\lambda)$

return $b' == b$

Fig. 2: The security experiment for PRFs.

Theorem 1 ([6]). *Let F be a weak PRF. Then one can construct subversion-resilient PRFs in the trusted amalgamation model.*

Note that their construction does not need any trusted operation like an XOR, but rather that the trusted amalgamation parses the input as a bit string and then forwards one of two possible keys to a weak PRF.

3 Subversion-Resilient One-Way Functions

One of the major observations Bemmann et al. [6] make is that certain primitives are *inherently* subversion-resilient. One such primitive is weak PRFs, which do not allow the attacker to input any value during the security game. Thus, one could hope a similar result also holds for one-way functions. Unfortunately, it seems that solely relying on one-wayness is insufficient, as we will need the outputs of the considered function also to be (pseudo-) random and sufficiently large. By guaranteeing these properties, we can argue that an adversary has a low chance that a prepared trigger matches a random challenge and thus wins

the security experiment. Therefore, instead of one-way functions, we consider one-way permutations, which we will later use to construct one-time signatures, particularly Lamport signatures [19]. This way, the challenge given to the adversary in the security game is a uniformly random element from the domain of the permutation. The critical point in our security proof will be that an adversary cannot access enough entropy to "hit" a *random* output of the one-way function/permutation while avoiding detection. In general, we can not assess the output distribution of a one-way function, even if the input is random. However, this is different for a one-way *permutation*, as the uniform input distribution implies a uniform output distribution. Thus, since the input distribution of the one-way permutation is public, Lemma 1 implies that there is only a negligible probability that the one-way permutation deviates from the specification. Then we can argue that the adversary cannot access enough entropy to develop an input that matches its challenge after being evaluated.

3.1 One-Way Permutations

We recall the standard definition of one-way functions and permutations.

Definition 3. *A (family of)* one-way functions Π *consists of two ppt algorithms* Gen *and* Eval*. On input* 1^λ*, the randomized algorithm* Gen *returns the* public parameters pp*. The deterministic algorithm* Eval *takes the public parameters* pp *and an element* $x \in \{0,1\}^\lambda$ *and returns an element* $y \in \{0,1\}^\lambda$*. If* $\mathsf{Eval}(pp, \cdot)$ *is a permutation on* $\{0,1\}^\lambda$*, we call* Π *a family of* one-way permutations.

Definition 4. *We say that* $\Pi = (\mathsf{KGen}, \mathsf{Eval})$ *is* secure*, if there is a negligible function* negl *such that for all ppt attackers* $\mathcal{A}$*, the probability* $\Pr[\mathsf{ExpInv}_\Pi^{\mathcal{A}}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$ *with* $\mathsf{ExpInv}_\Pi^{\mathcal{A}}(1^\lambda)$ *displayed in Figure 3.*

$\mathsf{ExpInv}_\Pi^{\mathcal{A}}(1^\lambda)$

$(pp) \leftarrow \mathsf{Gen}(1^\lambda)$

if Π is permutation : $y^* \xleftarrow{\$} \{0,1\}^\lambda$

else : $x \xleftarrow{\$} \{0,1\}^\lambda, y^* = \mathsf{Eval}(pp, x)$

$x^* = \mathcal{A}(pp, y^*)$

if $\mathsf{Eval}(pp, x^*) == y^*$: **return** 1

return 0

$\mathsf{ExpSig}_\Sigma^{\mathcal{A}}(1^\lambda)$

$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda)$

$(m, \sigma) = \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(1^\lambda, \mathsf{vk})$

if $\mathsf{Vf}(pk, (m, \sigma)) = 1$ and $m \notin Q$:

  return 1

return 0

Fig. 3: Left: One-way function/ -permutation security experiment. Right: Unforgeability experiment for digital signatures.

Note that other definitions exist in which x^* is chosen uniformly at random from the domain and $y^* = \mathsf{Eval}(pp, x^*)$ is given to the adversary. In the classical,

i.e., non-subversion setting, both definitions are equivalent for permutations. In our case, there is also little difference in our results. Since all inputs to Eval are known, i.e., public, an adversary can guarantee that Eval follows the specification but with negligible probability. Thus, in our proof in the next section, we could introduce an additional game hop and replace $\widetilde{\mathsf{Eval}}$ with $\widehat{\mathsf{Eval}}$, but instead, we will use this more straightforward way of defining security.

3.2 Subversion-Resilient One-Way Functions

We will see that starting from an "ordinary" one-way permutation, we directly obtain a subversion-resilient one-way function *without* the need of any further amalgamation, assuming that we already have a subversion-resilient key generation algorithm as stated in Section 2.4. The idea is that by applying Lemma 1 and using a permutation, the challenge handed to the adversary is a random element. Then we can use that the adversary has to provide its implementation *before* the execution of the inverting experiment, i.e., the challenge is *independent* from the subverted implementation. Since the subversion can only utilize negligible many triggers to avoid detection by the watchdog, the probability that a trigger can be used to break security is also negligible. Thus, it needs to find an input that can then be used to break the one-wayness of the specification *without* making use of an input trigger, which contradicts the usual non-subversion security.

Note that it is impossible (with polynomial testing time) to ensure that the implementation provided by an adversary is still a permutation. Even changing the output under a single input leads to the function not being a permutation anymore, which can only be detected with negligible probability by a polynomial time watchdog. Fortunately, we will only utilize the permutation property of the specification to guarantee a uniform output distribution of honest evaluations. Thus, we lose the permutation property in exchange for subversion-resilience.

Theorem 2. *Let $\Pi = (\mathsf{Gen}, \mathsf{Eval})$ be a one-way permutation. Then the trivial specification $\widehat{\Pi} = (\mathsf{Gen}, \mathsf{Eval})$ is a subversion-resilient one-way function in the split-program model with trusted amalgamation.*

Proof. Let $\widehat{\Pi} = (\mathsf{Gen}, \mathsf{Eval})$ be the specification of a permutation, and $\widetilde{\Pi}$ be the implementation of $\widehat{\Pi}$ provided by $\mathcal{A}$. First, the watchdog simply runs KGen for pp, samples x and compares $\widetilde{\mathsf{Eval}}(pp, x)$ to $\widehat{\mathsf{Eval}}(pp, x)$. Whenever a mismatch between these values is found, the watchdog returns 1. To prove the subversion-resilience, let $\mathcal{T} \subseteq \mathcal{PP} \times \{0,1\}^\lambda$ denote the *trigger set* such that $(pp, x) \in \mathcal{T} \Leftrightarrow \widetilde{\mathsf{Eval}}(pp, x) \neq \widehat{\mathsf{Eval}}(pp, x)$ where $\mathcal{PP}$ denotes the public parameter space. Thus, $\mathcal{T}$ contains *all* inputs for which the implementation deviates from the specification. To avoid the detection by a watchdog, we know that the density of $\mathcal{T}$ needs to be negligible, i.e., we have $|\mathcal{T}|/(|\mathcal{PP}| \cdot 2^\lambda) \in \mathsf{negl}(\lambda)$. Due to the flow of the subversion experiment, the attacker needs to provide the implementation $\widetilde{\Pi}$ *before* the parameters pp and the challenge y^* are chosen in the security game. Hence, the set $\mathcal{T}$ is *independent* of pp and y^* and so is the

image of $\mathcal{T}$, i.e., $\mathsf{img}(\mathcal{T})$. Now, whenever the attacker is successful (as in 'wins the security experiment') on input y^*, they will output a value x^* such that $\widetilde{\mathsf{Eval}}(pp, x^*) = y^*$. We now distinguish whether the adversary uses a trigger, i.e., whether $(pp, x^*) \in \mathcal{T}$ or $(pp, x^*) \notin \mathcal{T}$ holds. If $(pp, x^*) \notin \mathcal{T}$, we know that $y^* = \widetilde{\mathsf{Eval}}(pp, x^*) = \widehat{\mathsf{Eval}}(pp, x^*)$. Thus, the attacker on the subverted implementation can be transformed into an attacker on the non-subverted specification, breaking the one-wayness of Π. If $(pp, x^*) \in \mathcal{T}$, we can't predict $\widetilde{\mathsf{Eval}}(pp, x^*)$, however it can only redistribute weight within $\mathcal{T}$, as $\mathsf{Eval}(pp, \cdot)$ is a deterministic mapping on $\{0,1\}^\lambda \setminus \mathcal{T}$. Now, $\mathcal{T}$ is independent from y^* and y^* is uniformly drawn from $\{0,1\}^\lambda$. Together, this implies that the expected probability (upon the random choice of y^*) of a subversion attacker to win when submitting any trigger x^* (and setting up its implementation accordingly beforehand) is at most $|\mathcal{T}|/(|\mathcal{PP}| \cdot 2^\lambda)$, i.e., negligible. Hence, the probability that a trigger x^* with $\widetilde{\mathsf{Eval}}(pp, x^*) = y^*$ exists is negligible. □

4 Subversion-Resilient Hash Functions

Another crucial building block we will use is hash functions. Since we build one-time signatures from one-way functions, we need a way to hash two public keys (of the one-time signature) down to the size of one public key to make the signature construction of [24] work. Unfortunately, subversion-resilient collision-resistant hash functions seem impossible (without any further assumptions), as discussed in Section 6. On the positive side, just like in the case of ordinary signatures, subversion-resilient target collision-resistant hash functions are sufficient for our case, and we will see that they can be constructed by using a trusted XOR. So, let us begin by providing the necessary (security) definitions.

Definition 5. *A family of hash functions $\mathcal{H}$ is a pair of ppt algorithms* $(\mathsf{Gen}, \mathsf{H})$ *where* Gen *takes as input the security parameter* 1^λ *and outputs a (non-secret) key* s *and* H *takes as input a key* s *and a string* $x \in \{0,1\}^*$ *and outputs* $\mathsf{H}_s(x)$.

Note that we only consider keyed hash functions that take a fixed-length input. We will assume that inputs have length 2λ. Our approach can also handle inputs with lengths up to 2λ, but this would imply more encoding and notation overhead as inputs would need to be interpreted as 2λ long items with leading zeros.

In the following, we consider two different but very related security notions concerning hash functions.

Definition 6. *Let $\mathcal{H} = (\mathsf{Gen}, \mathsf{H})$ be a family of hash functions. Then we say that $\mathcal{H}$ is* target collision resistant (TCR) *iff* $\Pr[\mathsf{ExpTCR}^{\mathcal{H}}_{\mathcal{A}}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$ *where* $\mathsf{ExpTCR}^{\mathcal{H}}_{\mathcal{A}}(1^\lambda)$ *is depicted in Figure 4.*

Definition 7. *Let $\mathcal{H} = (\mathsf{Gen}, \mathsf{H})$ be a family of hash functions. Then we say that $\mathcal{H}$ is* random target collision resistant (rTCR) *iff* $\Pr[\mathsf{ExpRTCR}^{\mathcal{H}}_{\mathcal{A}}(1^\lambda) = 1] \leq \mathsf{negl}(secpar)$ *where* $\mathsf{ExpRTCR}^{\mathcal{H}}_{\mathcal{A}}(1^\lambda)$ *is depicted in Figure 4.*

$\mathsf{ExpTCR}^{\mathcal{H}}_{\mathcal{A}}(1^\lambda)$	$\mathsf{ExpRTCR}^{\mathcal{A}}_{\mathcal{H}}(1^\lambda)$
$(x, \mathsf{st}) = \mathcal{A}_1(1^\lambda)$	$x \xleftarrow{\$} \mathcal{D}$
$s \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$	$s \leftarrow \mathsf{Gen}(1^\lambda)$
$y = \mathcal{A}_2(s, x, \mathsf{st})$	$y = \mathcal{A}(s, x)$
if $\mathsf{H}_s(x) == \mathsf{H}_s(y)$ and $x \neq y$:	**if** $\mathsf{H}_s(x) == \mathsf{H}_s(y)$ and $x \neq y$:
return 1	**return** 1
return 0	**return** 0

Fig. 4: (random) Target-collision resistance security experiment for hash functions. In the left experiment, we use that $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and use st to denote the state passed between the subroutines of the adversary.

Big Domains. Before we present our construction, let us quickly illustrate a powerful subversion attack against hash function families with big domains, which is inspired by the attack on one-way permutations by Russel et al. in [27]. Let $\mathcal{H} = (\mathsf{Gen}, \mathsf{H})$ with $\mathsf{H}\colon \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$ be a family of hash functions, which hashes inputs to outputs half the input size. Then an adversary could prepare its implementation such that $\tilde{\mathsf{H}}_s(\mathsf{k} \parallel y) := y$ for some randomly sampled (or simply chosen by the adversary) string k. With this construction, an input trigger exists for *every* element in the range of H, enabling the adversary to win the security experiment trivially. Additionally, detecting this attack is very hard for an offline watchdog without knowledge of k. Assuming the watchdog samples random inputs for the hash function, the probability for a random input to match y is $(\frac{1}{2})^\lambda$, which is negligible in λ. Since the watchdog only has a polynomial running time, it has a negligible probability of detecting this attack. Thus, we only use hash functions where the domain and range of the hash functions are of similar size to rule out this otherwise unpreventable attack. More concretely, we only consider hash functions where the output is one bit shorter than the input. Larger input sizes are then handled by constructing hash functions for different input sizes and hashing the input down through these different hash functions. To guarantee subversion-resilience, we need to run a watchdog for each input length individually. However, this seems unavoidable to prevent the above attack.

Construction. Similar to our construction of subversion-resilient one-way functions, we make use of the fact that rTCR hash functions have a *random* challenge. More formally, let $\mathcal{H} = (\mathsf{Gen}, \mathsf{H})$ be a family of rTCR hash functions. Then we construct a TCR hash function $\mathcal{H}' = (\mathsf{Gen}', \mathsf{H}')$ as follows: To sample a key s, the algorithm Gen' first executes Gen and then additionally samples a uniformly random element r from the domain of the hash function and finally outputs $s' = (s, r)$ as the key. Now, H' evaluates inputs as $\mathsf{H}'_{s'}(x) := \mathsf{H}_s(x \oplus r)$. Thus, this construction has an additional blinding value as part of its key, which is XORed to the input before evaluating the hash function. In order to sanitize key generation, our watchdog will test Gen for uniformly random coins. Thus,

just as in [28] we can guarantee that either s is computed in accordance with the specification or the watchdog detects subversion. To compute the blinding value, we can use any construction from [28] or [7] to produce random coins in a subversion-resilient manner that does not use random oracles.

We note that the $\oplus$ operation will be part of the trusted amalgamation when we prove subversion-resilience. This is essential to the construction and prevents the attacker from feeding adversarially chosen inputs directly into subverted components, similar to Russell et al. [28]. Just like in the section about one-way functions, the order of events is critical for our analysis. Our security proof again uses that the hash function (and especially the random value provided along) is provided to the adversary *after* the adversary provides its implementation. Note that in a non-subverted setting, our construction is the folklore[6] construction to obtain a TCR hash function from a rTCR hash function.

In the proof, we will use that target collision resistant hash functions with small input domain need to distribute their inputs somewhat 'equally' into the range of the hash functions. Otherwise, this would contradict its target collision resistance property.

Lemma 2. *Let $\mathcal{H} = (\mathsf{Gen}, \mathsf{H})$ with $\mathsf{H} : \{0,1\}^\lambda \to \{0,1\}^{\lambda-1}$ be a rTCR family of hash functions. Then, the set $\{x \in \{0,1\}^\lambda \mid \mathsf{H}_s(x) = z\}$ is negligible in λ with probability $1 - \mathsf{negl}(\lambda)$ upon random choice of z and s.*

Theorem 3. *Let $\mathcal{H} = (\mathsf{Gen}, \mathsf{H})$ with $\mathsf{H} : \{0,1\}^\lambda \to \{0,1\}^{\lambda-1}$ be a rTCR family of hash functions. Then $\mathcal{H}' = (\mathsf{Gen}', \mathsf{H}')$ with $\mathsf{H}' : \{0,1\}^\lambda \to \{0,1\}^{\lambda-1}$ as described above is a subversion-resilient TCR family of hash functions in the split-program model with trusted amalgamation where the $\oplus$ is part of the amalgamation.*

Proof. Let $\mathcal{H}$ be a rTCR hash function family, and let $\mathcal{T}$ be the trigger set of H, i.e., $(s,x) \in \mathcal{T} \Leftrightarrow \widetilde{\mathsf{H}}_s(x) \neq \widehat{\mathsf{H}}_s(x)$. Just as in [28], we can use Lemma 1 to argue that either the keys s' output by Gen' are computed according to the specification or the watchdog detects subversion. Further, due to Lemma 1, we know that $|\mathcal{T}| \in \mathsf{negl}(\lambda)$. Hence, our watchdog for H will query Gen and H' on random inputs. Due to the trusted XOR used in H', Lemma 1 implies that with high probability the value $\mathsf{H}_s(x \oplus r)$ is a non-subverted output, as $s' = (s, r)$ is chosen *after* the adversary provides its implementation. Now, let $\mathcal{A}$ be an adversary against the subversion-resilience of H', i.e., $\mathcal{A}$ first outputs x, is then handed $s' = (s, r)$ and then outputs a value $y \neq x$ and succeeds if $\mathsf{H}_s(x \oplus r) = \mathsf{H}_s(y \oplus r)$. We now distinguish two cases. In the first case, we have $(s, y) \notin \mathcal{T}$. If $\mathcal{A}$ can output $y \neq x$ such that $\mathsf{H}'_{s'}(x) = \mathsf{H}'_{s'}(y)$ (where both inputs do *not* lead to input trigger), we can construct an adversary $\mathcal{B}$ which breaks the rTCR of $\mathcal{H}$ as follows. After $\mathcal{A}$ outputs some value x, the adversary $\mathcal{B}$ obtains (s, x') from its challenger. Now, $\mathcal{B}$ forwards $s' = (s, r)$ with $r = x \oplus x'$ to $\mathcal{A}$ which answers with some y. Finally, $\mathcal{B}$ forwards $y \oplus r$ to its challenger. We observe that in the case that $\mathcal{A}$ finds a collision such that $\mathsf{H}'_{s'}(x) = \mathsf{H}'_{s'}(y)$, it holds that $\mathsf{H}'_{s'}(x) = \mathsf{H}_s(x' \oplus x \oplus x) = \mathsf{H}_s(x')$ and $\mathsf{H}'_{s'}(y) = \mathsf{H}_s(y \oplus x \oplus x')$. Since $x \neq y$, it

[6] Unfortunately, we were not able to find an explicit reference for this construction.

also holds that $x' \neq y \oplus x' \oplus x$. Thus, if $\mathcal{A}$ finds a collision, so does $\mathcal{B}$, at least if H does not deviate from its specification with regard to (s, y)[7].

The other case is $(s, y) \in \mathcal{T}$. But, as we will now argue, this can only happen with negligible probability. Remember that H maps λ-bit string to $(\lambda - 1)$-bit strings. Now, let $\mathsf{H}_s^{-1}(z) \subseteq \{0,1\}^\lambda$ denote the set of preimages of an element $z \in \{0,1\}^{\lambda-1}$. By Lemma 2, the size of $\mathsf{H}_s^{-1}(z)$ must be negligible for all but negligible many pairs (s, z). Hence, the probability that there is some $y \in \mathsf{H}_s^{-1}(\mathsf{H}s(x))$ with $(s, y) \in \mathcal{T}$ is negligible, since $\mathcal{A}$ commits to its implementation before H and its associated blinding value is chosen and $\mathcal{A}$ has only negligible many input trigger. Thus, any adversary which breaks the subversion-resilience of $\mathcal{H}'$ can also be used to break the security of $\mathcal{H}$. □

As stated before, the above construction only reduces the input size by a single bit. Hence, to hash a string of length 2λ down to length λ, we will need a hash family $\mathcal{H}_\ell \colon \{0,1\}^\ell \to \{0,1\}^{\ell-1}$ for each $\ell = 2\lambda, 2\lambda - 1, \ldots, \lambda + 1$.

5 Subversion-resilient Signatures

Finally, we have all the ingredients to prove the signature scheme based on the Naor-Yung construction [24] subversion-resilient. As a necessary stepping stone, we will see that the classical Lamport signatures are subversion-resilient if instantiated with a subversion-resilient one-way function. Then, all the previous sections' building blocks can be combined to obtain a subversion-resilient signature, where even the verification algorithm is subject to subversion.

5.1 Digital Signatures

We continue by recalling the standard definition of digital signatures.

Definition 8. *A digital signature scheme Σ consists of three ppt algorithms $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$. On input 1^λ the key generation algorithm KGen outputs a pair of keys $(\mathsf{sk}, \mathsf{vk})$. The signing algorithm Sign takes as input the secret signing key sk and a message m from the message space and outs a signature σ. The verification algorithm Vf takes as input the public verification key vk, a message m, and a signature σ. It outputs a bit b where $b = 1$ indicates a valid signature, while $b = 0$ means that the signature cannot be verified. We say a signature scheme is* correct *if for every key pair $(\mathsf{sk}, \mathsf{vk})$ generated by $\mathsf{KGen}(1^\lambda)$ and every message $m \in M$ it holds that $\mathsf{Vf}(\mathsf{vk}, (m, \mathsf{Sign}(\mathsf{sk}, m)) = 1$ but with negligible probability.*

Next, we recall the standard definition of existential unforgeability.

Definition 9. *We say that a signature scheme Σ is* existentially unforgeable *if for all ppt adversaries $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(1^\lambda)$ such that $\Pr[\mathsf{ExpSig}_\Sigma^{\mathcal{A}}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$ where $\mathsf{ExpSig}_\Sigma^{\mathcal{A}}(1^\lambda)$ is displayed in Figure 3 and $\mathcal{A}$ has access to an oracle returning $\sigma_i = \mathsf{Sign}(\mathsf{sk}, m_i)$ on input m_i and where Q denotes the set of all queries that $\mathcal{A}$ issued to its signing oracle.*

[7] This resembles the 'classical' security proof of the construction.

Definition 10. *We say a signature is a* one-time *signature if the above holds and the attacker can only issue a single query to its signing oracle.*

5.2 Lamport Signatures

Using the results of Section 3, we have access to subversion-resilient one-way functions and can directly obtain Lamport signatures [19] given a trusted comparison. So let us quickly recall the definition of the aforementioned Lamport signatures for messages of length ℓ, which uses a family of one-way functions $(\mathsf{Gen}, \mathsf{Eval})$.

The key generation algorithm chooses ℓ many values $x_{i,0}, x_{i,1} \in \{0,1\}^\lambda$ uniformly at random as well as $pp = \mathsf{Gen}(1^\lambda)$. Then compute $y_{i,0} = \mathsf{Eval}(pp, x_{i,0})$ and $y_{i,1} = \mathsf{Eval}(pp, x_{i,1})$. The verification key vk consists of all y values and the signing key of all x values. On input a message $m \in \{0,1\}^\ell$ with $m = m_1 \dots m_\ell$, the signing algorithm simply outputs the signature $\sigma = (x_{1,m}, \dots, x_{\ell,m_\ell})$. On input a verification key vk, a message $m \in \{0,1\}^\ell$ with $m = (m_1 \dots m_\ell)$, and a signature $\sigma = (x_1, \dots, x_\ell)$, the verification algorithm outputs 1 iff $\mathsf{Eval}(pp, x_i) = y_{i,m_i}$ for all $1 \leq i \leq \ell$.

Then it is not hard to see that the security of the Lamport signatures scheme follows directly from the security of the used one-way function. Similarly, the Lamport signature's subversion-resilience follows from the subversion-resilience of the used one-way function. However, additionally, we need a trusted comparison for the above construction to be secure. As discussed in [6] for the context of MACs, a trusted comparison seems unavoidable. Otherwise, the subverted implementation could ignore the output of Eval and output 1 for a value chosen by the adversary and embedded into the implementation. Thus, the subversion-resilience of Lamport signatures directly boils down to the subversion-resilience of the one-way function.

Theorem 4. *Let Π be a subversion-resilient one-way function. Then Lamport Signatures using Π as the one-way function are subversion-resilient one-time signatures where the trusted amalgamation makes a trusted comparison.*

5.3 The Naor-Yung Construction

Before we dive into the classical Naor-Yung construction, let us provide some intuition on the approach. The main idea is to follow a tree-based approach and heavily use one-time signatures, which sign pairs of verification keys to form an authenticated path in a tree based on the message to be signed. Since the Lamport signature can not sign messages bigger than its public key, a hash function is used to allow the signing of two verification keys. Here a target-collision-resistant hash function is sufficient to guarantee security. While the original construction is stateful, it is known that it can be extended via PRFs and deterministically recomputing keys to make the construction stateless. Note that the PRFs are only needed to sign messages and not for signature verification. We continue with the construction and are given a one-time signature scheme

$(\mathsf{KGen}_{\mathrm{OTS}}, \mathsf{Sign}_{\mathrm{OTS}}, \mathsf{Vf}_{\mathrm{OTS}})$, a target-collision resistant hash function family $\mathcal{H} = (\mathsf{Gen}, \mathsf{H})$ with $\mathsf{H} = \{\mathsf{H}_s \colon \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}\}$, and a pseudorandom function $(\mathsf{KGen}_{\mathsf{PRF}}, F)$. Furthermore, for a string $w \in \{0,1\}^*$, we define $\mathsf{Pre}(w) \subseteq \{0,1\}^*$ as the set of prefixes of w, including the empty string ϵ and w itself. For technical reasons, we assume that for $w \in \{0,1\}^{\lambda}$, we have $\mathsf{Pre}(w) \subseteq \{0,1\}^{\lambda+\lceil \log(\lambda) \rceil}$ and $|\mathsf{Pre}(w)| = |w| + 1$ to guarantee that all prefixes have the same length and to differentiate them uniquely.[8] We also assume that the verification key vk corresponding to a secret key sk can easily be derived from sk. Now, we define our signature scheme $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ as follows:

$\mathsf{KGen}(1^{\lambda})$

$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}_{\mathrm{OTS}}(1^{\lambda})$

$\mathsf{k}_{\mathsf{keys}} \leftarrow \mathsf{KGen}_{\mathsf{PRF}}(1^{\lambda})$

$\mathsf{k}_{\mathsf{sigs}} \leftarrow \mathsf{KGen}_{\mathsf{PRF}}(1^{\lambda})$

$\mathsf{k}_{\mathsf{hashs}} \leftarrow \mathsf{KGen}_{\mathsf{PRF}}(1^{\lambda})$

return $((\mathsf{sk}, \mathsf{k}_{\mathsf{keys}}, \mathsf{k}_{\mathsf{sigs}}, \mathsf{k}_{\mathsf{hashs}}), \mathsf{vk})$

$\mathsf{Vf}(\mathsf{vk}, m, \sigma)$

Parse σ as $(\sigma_m, \mathsf{vk}_m, (\sigma_w, s_w, \mathsf{vk}_{w\|0}, \mathsf{vk}_{w\|1})_{w \in \mathsf{Pre}(m) \setminus \{m\}})$

$\mathsf{vk}_{\epsilon} = \mathsf{vk}$

for $w \in \mathsf{Pre}(m) \setminus \{m\}$:

$b_w = \mathsf{Vf}(\mathsf{vk}_w, \mathsf{H}_{s_w}(\mathsf{vk}_{w\|0} \| \mathsf{vk}_{w\|1}), \sigma_w)$

$b_m = \mathsf{Vf}(\mathsf{vk}_m, m, \sigma_m)$

return $\bigwedge_{w \in \mathsf{Pre}(m)} b_w$

$\mathsf{Sign}((\mathsf{sk}, \mathsf{k}_{\mathsf{keys}}, \mathsf{k}_{\mathsf{sigs}}, \mathsf{k}_{\mathsf{hashs}}), m)$

$\mathsf{sk}_{\epsilon} = \mathsf{sk}$

$\mathsf{vk}_{\epsilon} = \mathsf{vk}$

for $w \in \mathsf{Pre}(m) \setminus \{\epsilon\}$:

$r_w = F(\mathsf{k}_{\mathsf{keys}}, w)$

$(\mathsf{sk}_w, \mathsf{vk}_w) = \mathsf{KGen}_{\mathrm{OTS}}(1^{\lambda}; r_w)$

for $w \in \mathsf{Pre}(m) \setminus \{m\}$:

$r_{w,h} = F(\mathsf{k}_{\mathsf{hashs}}, w)$

$s_w = \mathsf{Gen}(1^{\lambda}; r_{w,h})$

$r_w = F(\mathsf{k}_{\mathsf{sigs}}, w)$

$\sigma_w = \mathsf{Sign}_{\mathrm{OTS}}(\mathsf{sk}_w, \mathsf{H}_{s_w}(\mathsf{vk}_{w\|0} \| \mathsf{vk}_{w\|1}); r_w)$

$r_m = F(\mathsf{k}_{\mathsf{sigs}}, m)$

$\sigma_m = \mathsf{Sign}_{\mathrm{OTS}}(\mathsf{sk}_m, m; r_m)$

return $(\sigma_m, (\sigma_w, s_w, \mathsf{vk}_{w\|0}, \mathsf{vk}_{w\|1})_{w \in \mathsf{Pre}(m) \setminus \{m\}})$

Fig. 5: Our proposed signature scheme.

Theorem 5. *Given subversion-resilient one-time signatures, subversion-resilient target-collision-resistant hash functions, and subversion-resilient PRFs, then the above construction is a stateless, subversion-resilient digital signature scheme in the split-program model with trusted amalgamation where* all *algorithms are subject to subversion.*

In the following proof, we follow the proof sketch by Naor and Yung [24], but need to adapt the proof somewhat. First, Naor and Yung only considered a stateful signature while our use of the PRF makes the complete construction stateless. Furthermore, we need to make sure that we reduce the security to the

[8] This prevents complications and allows us to identify each prefix uniquely.

subversion-resilience of the building blocks rather than their original security properties, as we only work with the (possibly) subverted implementation here and not with the specification.

Proof. As a first step, the watchdog for the signature scheme simply runs the watchdog of the one-time signature, the watchdog of the hash function, and the watchdog of the PRF. If none of these watchdogs detect a subversion, we follow an adaption of the proof by Naor and Yung [24].

Now, we replace the values generated by the PRF with completely random strings, i.e., all strings r_w, $r_{w,h}$, and r_m are now independent random strings that are stored by the system for reuse in case that the values are needed again. If this would be distinguishable from the setting where the PRF is used, we can easily build an attacker against the subversion-resilience of the PRF by simulating all other parts of the construction. We will also ignore the cases which some of the randomly chosen values (random strings or keys) collide, as this will only happen with negligible probability.

Now, let $\mathcal{A}_{\mathsf{sigs}}$ be an attacker against the subversion-resilience of the signature scheme that is successful with non-negligible probability $1/p(\lambda)$ for some non-negligible function p. In the following, we will now show that such an attacker implies the existence of an attacker $\mathcal{A}_{\mathrm{OTS}}$ against the one-time signature and an attacker $\mathcal{A}_{\mathsf{hashs}}$ against the hash function such that at least one of these attackers is also successful with non-negligible probability. As $\mathcal{A}_{\mathsf{SIG}}$ wins the subversion-resilience game with non-negligible probability, it outputs a valid message-signature pair (m^*, σ^*) with $m^* \notin Q_M$ with non-negligible probability. Here, Q_M is the set of messages for which $\mathcal{A}$ queried its signing oracle. Let Q_S be the set of signatures returned by the signing oracles. By definition, for each $m \in Q_M$ and each corresponding answer $\sigma \in Q_S$, we have $\sigma = (\sigma_m, (\sigma_w, s_w, \mathsf{vk}_{w\|0}, \mathsf{vk}_{w\|1})_{w \in \mathsf{Pre}(m)\setminus\{m\}})$. Similarly, we also have $\sigma^* = (\sigma^*_{m^*}, (\sigma^*_{w^*}, s_{w^*}, \mathsf{vk}_{w^*\|0}, \mathsf{vk}_{w^*\|1})_{w^* \in \mathsf{Pre}(m^*)\setminus\{m^*\}})$. By construction of the verification algorithm, a successfully forged signature σ^* must contain a tuple $(\sigma^*_{w^*}, s_{w^*}, \mathsf{vk}_{w^*\|0}, \mathsf{vk}_{w^*\|1})$ that is not contained in any signature in Q_S. Now, we need to distinguish two cases.

If $\mathsf{H}_{s_{w^*}}(\mathsf{vk}_{w^*\|0}\|\mathsf{vk}_{w^*\|1}) \neq \mathsf{H}_{s_{w^*}}(\mathsf{vk}_{w'\|0}\|\mathsf{vk}_{w'\|1})$ for all $\mathsf{vk}_{w'\|0}$ and $\mathsf{vk}_{w'\|1}$ contained in the signatures in Q_S, we can construct an attacker $\mathcal{A}_{\mathrm{OTS}}$ against the one-time signature. The attacker $\mathcal{A}_{\mathrm{OTS}}$ is given some verification key vk' from the one-time signature and simulates the complete security experiment, but instead of sampling the key pair $(\mathsf{sk}_{w^*}, \mathsf{vk}_{w^*})$, it sets $\mathsf{vk}_{w^*} = \mathsf{vk}'$. To sign a message with sk_{w^*}, it uses its oracle to the signing algorithm of the one-time signature. Finally, $\mathcal{A}_{\mathrm{OTS}}$ outputs the message-signature pair $(m', \sigma') = (\mathsf{H}_{s_{w^*}}(\mathsf{vk}_{w^*\|0}\|\mathsf{vk}_{w^*\|1}), \sigma^*_{w^*})$, which is a valid pair as (m^*, σ^*) was a valid pair for the signature scheme. Furthermore, as $\mathsf{H}_{s_{w^*}}(\mathsf{vk}_{w^*\|0}\|\mathsf{vk}_{w^*\|1}) \neq \mathsf{H}_{s_{w^*}}(\mathsf{vk}_{w'\|0}\|\mathsf{vk}_{w'\|1})$ holds for all verification keys $\mathsf{vk}_{w'\|0}$ and $\mathsf{vk}_{w'\|1}$ contained in Q_S, the one-time signing oracle was never queried on the value $\mathsf{H}_{s_{w^*}}(\mathsf{vk}_{w^*\|0}\|\mathsf{vk}_{w^*\|1})$. Hence, (m', σ') is a successful forgery of the one-time signature.

If some signature in Q_S contains a tuple $(\sigma^*_{w^*}, s_{w^*}, \mathsf{vk}_{w'\|0}||\mathsf{vk}_{w'\|1})$ with

$$\mathsf{H}_{s_{w^*}}(\mathsf{vk}_{w^*\|0}||\mathsf{vk}_{w^*\|1}) = \mathsf{H}_{s_{w^*}}(\mathsf{vk}_{w'\|0}\|\mathsf{vk}_{w'\|1}),$$

which was created by signing a message m', we can build the attacker $\mathcal{A}_{\mathsf{hashs}}$ against the hash function as follows: The attacker $\mathcal{A}_{\mathsf{hashs}}$ simulates the complete experiment but does not sample a hash function $\mathsf{H}_{s_{w^*}}$. Instead, before $\mathsf{H}_{s_{w^*}}$ is evaluated during a signing operation of m', the attacker returns the value $\mathsf{vk}_{w'\|0}\|\mathsf{vk}_{w'\|1}$ to the hash function challenge and then obtains a hash function h, which will be used as $\mathsf{H}_{s_{w^*}}$. Finally, the attacker $\mathcal{A}_{\mathsf{H}}$ outputs $\mathsf{vk}_{w^*\|0}||\mathsf{vk}_{w^*\|1}$. As $\mathsf{H}_{s_{w^*}}(\mathsf{vk}_{w^*\|0}||\mathsf{vk}_{w^*\|1}) = \mathsf{H}_{s_{w^*}}(\mathsf{vk}_{w'\|0}\|\mathsf{vk}_{w'\|1})$, this is a valid collision of the hash function keyed with s_{w^*}.

If $\mathcal{A}$ wins the security experiment with probability $p(\lambda)$ for some non-negligible function $p(\lambda)$, the attacker $\mathcal{A}_{\mathrm{OTS}}$ wins with probability $p_{\mathrm{OTS}}(\lambda)$, and the attacker $\mathcal{A}_{\mathsf{hashs}}$ wins with probability $p_{\mathsf{hashs}}(\lambda)$, we have $p(\lambda) \leq p_{\mathrm{OTS}}(\lambda) + p_{\mathsf{hashs}}(\lambda)$. Hence, either $\mathcal{A}_{\mathrm{OTS}}$ or $\mathcal{A}_{\mathsf{hashs}}$ is successful if $\mathcal{A}$ is successful. □

6 Discussion

Efficiency. To better assess our results, in Table 1 we provide an overview of the available constructions of subversion-resilient signatures found in the literature. The table shows that while our construction grants the strongest security in the watchdog model, i.e. no random oracle and complete subversion, it also has the biggest signature size. Note that for the reverse firewall (RF) model and the self-guarding (SG) model, additional/other assumptions are applied (verifiability, honest sample phase).

Table 1: Comparison of different approaches for subversion-resilient signature schemes. Here σ denotes the size of an underlying signature scheme, m denotes the length of the messages to be signed, and s is the size of the key of our hash function.

	Model	RO	Complete Subv.	Signature size	Stateful subv.
[2]	RF	✗	✗	σ	✓
[16]	SG	✗	✗	$\approx \lambda \cdot m + 2\lambda\sigma$	✗
[27]	online WD	✓	✓	m	✓
[13]	offline WD	✓	✗	m	✗
[13]	offline WD	✗	✗	$2(m + \sigma)$	✗
This work	offline WD	✗	✓	$m(\sigma + \lvert s\rvert + 2\lvert vk\rvert) + \sigma$	✗

It is well known that digital signatures can be constructed from one-way and collision-resistant hash functions. Thus, we now focus on constructing collision-resistant hash functions and explain why this seems impossible if the hash function is not idealized as a random oracle.

Subversion-resilient Collision Resistance via Black-box Testing. Similar to the case of weak PRFs [6] and one-way permutations (see Section 3), one may hope that simply taking any hash function and testing it sufficiently may already grant positive results. Unfortunately, this seems impossible. Consider an adversary which provides an implementation $\tilde{H}$ of H, which only differ for two values m_0, m_1 from H in the sense that $\tilde{H}(m_0) = 0 = \tilde{H}(m_1)$. Any watchdog that samples messages from the (finite) domain[9] of the hash function uniformly at random only has negligible probability in testing for m_0 or m_1. Conversely, the adversary can trivially output a collision by outputting m_0, m_1. While this observation is not very involved, to the best of our knowledge, it was not yet formally written down in previous works.

Implications for Signatures. In some textbooks for modern cryptography, such as [18], the construction of Naor-Yung is often displayed by utilizing collision-resistant hash functions instead of target-collision-resistant hash functions. This is useful from a teaching perspective, as collision resistance is introduced in courses, and there is little benefit in introducing target-collision resistance if only the Naor-Yung construction is considered. While the classical setting makes little difference in which notion is used, the distinction between these two notions is crucial in the subversion setting. As the stronger notion seems impossible to achieve, the weaker and sufficient property allows for the subversion-resilient construction.

Correctness. Note that both of our signature construction satisfies our correctness definition, even under subversion. This is because due to the testing of the watchdog Lemma 1 can be used to argue that only for negligible many inputs correctness is violated. Unfortunately, our approach cannot achieve perfect correctness (as achieved by the symmetric encryption construction in [6]). Note that no work achieves perfect correctness other than assuming verifiability in the reverse firewall model [2], thus assuming correctness.

Acknowledgements. The authors would like to thank all anonymous reviewers for their valuable comments. The work of Rongmao Chen is supported by the National Natural Science Foundation of China (Grant No.62122092, No.62032005).

References

1. Armour, M., Poettering, B.: Algorithm substitution attacks against receivers. Int. J. Inf. Sec. 21(5), 1027–1050 (2022)
2. Ateniese, G., Magri, B., Venturi, D.: Subversion-resilient signature schemes. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 2015: 22nd Conference on Computer and Communications Security. pp. 364–375. ACM Press (Oct 2015)
3. Baek, J., Susilo, W., Kim, J., Chow, Y.W.: Subversion in practice: How to efficiently undermine signatures. Cryptology ePrint Archive, Report 2018/1201 (2018), `https://eprint.iacr.org/2018/1201`

[9] As is the case for tree-based signatures

4. 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: 22nd Conference on Computer and Communications Security. pp. 1431–1440. ACM Press (Oct 2015)
5. Bellare, M., Paterson, K.G., Rogaway, P.: Security of symmetric encryption against mass surveillance. In: Garay, J.A., Gennaro, R. (eds.) Advances in Cryptology – CRYPTO 2014, Part I. Lecture Notes in Computer Science, vol. 8616, pp. 1–19. Springer, Heidelberg (Aug 2014)
6. Bemmann, P., Berndt, S., Diemert, D., Eisenbarth, T., Jager, T.: Subversion-resilient authenticated encryption without random oracles. In: ACNS. Lecture Notes in Computer Science, vol. 13906, pp. 460–483. Springer (2023)
7. Bemmann, P., Chen, R., Jager, T.: Subversion-resilient public key encryption with practical watchdogs. In: Garay, J. (ed.) PKC 2021: 24th International Conference on Theory and Practice of Public Key Cryptography, Part I. Lecture Notes in Computer Science, vol. 12710, pp. 627–658. Springer, Heidelberg (May 2021)
8. 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: 24th Conference on Computer and Communications Security. pp. 1649–1660. ACM Press (Oct / Nov 2017)
9. Berndt, S., Wichelmann, J., Pott, C., Traving, T.H., Eisenbarth, T.: ASAP: Algorithm substitution attacks on cryptographic protocols. In: Suga, Y., Sakurai, K., Ding, X., Sako, K. (eds.) ASIACCS 22: 17th ACM Symposium on Information, Computer and Communications Security. pp. 712–726. ACM Press (May / Jun 2022)
10. Chakraborty, S., Dziembowski, S., Nielsen, J.B.: Reverse firewalls for actively secure MPCs. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020, Part II. Lecture Notes in Computer Science, vol. 12171, pp. 732–762. Springer, Heidelberg (Aug 2020)
11. Chen, R., Huang, X., Yung, M.: Subvert KEM to break DEM: Practical algorithm-substitution attacks on public-key encryption. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2020, Part II. Lecture Notes in Computer Science, vol. 12492, pp. 98–128. Springer, Heidelberg (Dec 2020)
12. 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.) Advances in Cryptology – ASIACRYPT 2016, Part I. Lecture Notes in Computer Science, vol. 10031, pp. 844–876. Springer, Heidelberg (Dec 2016)
13. 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: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part I. Lecture Notes in Computer Science, vol. 11442, pp. 221–251. Springer, Heidelberg (Apr 2019)
14. Degabriele, J.P., Farshim, P., Poettering, B.: A more cautious approach to security against mass surveillance. In: Leander, G. (ed.) Fast Software Encryption – FSE 2015. Lecture Notes in Computer Science, vol. 9054, pp. 579–598. Springer, Heidelberg (Mar 2015)
15. Dodis, Y., Mironov, I., Stephens-Davidowitz, N.: Message transmission with reverse firewalls—secure communication on corrupted machines. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology – CRYPTO 2016, Part I. Lecture Notes in Computer Science, vol. 9814, pp. 341–372. Springer, Heidelberg (Aug 2016)

16. Fischlin, M., Mazaheri, S.: Self-guarding cryptographic protocols against algorithm substitution attacks. In: Chong, S., Delaune, S. (eds.) CSF 2018: IEEE 31st Computer Security Foundations Symposium. pp. 76–90. IEEE Computer Society Press (2018)
17. Galteland, H., Gjøsteen, K.: Subliminal channels in post-quantum digital signature schemes. Cryptology ePrint Archive, Report 2019/574 (2019), https://eprint.iacr.org/2019/574
18. Katz, J., Lindell, Y.: Introduction to Modern Cryptography, Second Edition. CRC Press (2014)
19. Lamport, L.: Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory (Oct 1979)
20. Liu, C., Chen, R., Wang, Y., Wang, Y.: Asymmetric subversion attacks on signature schemes. In: Susilo, W., Yang, G. (eds.) ACISP 18: 23rd Australasian Conference on Information Security and Privacy. Lecture Notes in Computer Science, vol. 10946, pp. 376–395. Springer, Heidelberg (Jul 2018)
21. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) Advances in Cryptology – CRYPTO'89. Lecture Notes in Computer Science, vol. 435, pp. 218–238. Springer, Heidelberg (Aug 1990)
22. Mironov, I., Stephens-Davidowitz, N.: Cryptographic reverse firewalls. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015, Part II. Lecture Notes in Computer Science, vol. 9057, pp. 657–686. Springer, Heidelberg (Apr 2015)
23. Naor, M., Reingold, O.: Synthesizers and their application to the parallel construction of pseudo-random functions. Journal of Computer and System Sciences 58(2), 336–375 (1999)
24. Naor, M., Yung, M.: Universal one-way hash functions and their cryptographic applications. In: 21st Annual ACM Symposium on Theory of Computing. pp. 33–43. ACM Press (May 1989)
25. 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
26. Discussion about kyber's tweaked fo transform (2023), https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/WFRDl8DqYQ4, discussion Thread on the PQC mailing list
27. Russell, A., Tang, Q., Yung, M., Zhou, H.S.: Cliptography: Clipping the power of kleptographic attacks. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology – ASIACRYPT 2016, Part II. Lecture Notes in Computer Science, vol. 10032, pp. 34–64. Springer, Heidelberg (Dec 2016)
28. 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: 24th Conference on Computer and Communications Security. pp. 907–922. ACM Press (Oct / Nov 2017)
29. Russell, A., Tang, Q., Yung, M., Zhou, H.S.: Correcting subverted random oracles. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018, Part II. Lecture Notes in Computer Science, vol. 10992, pp. 241–271. Springer, Heidelberg (Aug 2018)
30. Teseleanu, G.: Threshold kleptographic attacks on discrete logarithm based signatures. In: Lange, T., Dunkelman, O. (eds.) Progress in Cryptology - LATINCRYPT 2017: 5th International Conference on Cryptology and Information Secu-

rity in Latin America. Lecture Notes in Computer Science, vol. 11368, pp. 401–414. Springer, Heidelberg (Sep 2019)
31. Young, A., Yung, M.: The dark side of "black-box" cryptography, or: Should we trust capstone? In: Koblitz, N. (ed.) Advances in Cryptology – CRYPTO'96. Lecture Notes in Computer Science, vol. 1109, pp. 89–103. Springer, Heidelberg (Aug 1996)
32. Young, A., Yung, M.: Kleptography: Using cryptography against cryptography. In: Fumy, W. (ed.) Advances in Cryptology – EUROCRYPT'97. Lecture Notes in Computer Science, vol. 1233, pp. 62–74. Springer, Heidelberg (May 1997)