BERGISCHE
UNIVERSITÄT
WUPPERTAL

Doctoral Dissertation

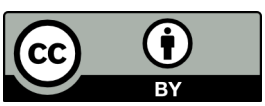
On Practical Subversion-Resilience

Pascal Bemmann, M.Sc.

October 30, 2023

Submitted to the
School of Electrical, Information and Media Engineering
University of Wuppertal

for the degree of
Doktor-Ingenieur (Dr.-Ing.)



Pascal Bemmann
Place of birth: Lübbecke, Germany
Author's contact information:
pasbemcal@gmail.com

Thesis Advisor:	**Prof. Dr.-Ing. Tibor Jager** University of Wuppertal, Wuppertal, Germany
Second Examiner:	**Prof. Dr. Rongmao Chen** College of Computer, National University of Defense Technology, Changsha, China
Thesis submitted:	October 30, 2023
Thesis defense:	January 31, 2024
Last revision:	February 28, 2024

Publication Overview

Publications in this thesis.

[BBCJ23] Pascal Bemmann, Sebastian Berndt, Rongmao Chen, and Tibor Jager. Subversion-resilient public key encryption with practical watchdogs. Cryptology ePrint Archive, Paper 2021/230, 2023. URL: `https://eprint.iacr.org/archive/2021/230/20231011:061958`.
This is the corrected full version of:
Pascal Bemmann, Rongmao Chen, and Tibor Jager. Subversion-resilient public key encryption with practical watchdogs. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 627–658, Virtual Event, May 10–13, 2021. Springer, Heidelberg, Germany. `doi:10.1007/978-3-030-75245-3_23`.

[BBD+23] Pascal Bemmann, Sebastian Berndt, Denis Diemert, Thomas Eisenbarth, and Tibor Jager. Subversion-resilient authenticated encryption without random oracles. In *Applied Cryptography and Network Security - 21st International Conference, ACNS 2023, Kyoto, Japan, June 19-22, 2023, Proceedings, Part II*, pages 460–483, 2023. `doi:10.1007/978-3-031-33491-7\_17`

[BBC24] Pascal Bemmann, Sebastian Berndt, and Rongmao Chen. Subversion-resilient signatures without random oracles. In *Applied Cryptography and Network Security - 22nd International Conference, ACNS 2024, Abu Dhabi, United Arab Emirates*, 2024. To appear.

Other publications.

[BBB+17] Pascal Bemmann, Felix Biermeier, Jan Bürmann, Arne Kemper, Till Knollmann, Steffen Knorr, Nils Kothe, Alexander Mäcker, Manuel Malatyali, Friedhelm Meyer auf der Heide, Sören Riechers, Johannes Schaefer, and Jannik Sundermeier. Monitoring of domain-related problems in distributed data streams. In *Structural Information and Communication Complexity - 24th International Colloquium, SIROCCO 2017, Porquerolles, France, June 19-22, 2017, Revised Selected Papers*, volume 10641 of *Lecture Notes in Computer Science*, pages 212–226. Springer, 2017

Acknowledgements

I want to express my heartfelt appreciation to my supervisor, Tibor Jager. I am so thankful for the opportunity to do my Ph.D. with you and like to thank you for your patience, trust, and support over the last years. Not only is Tibor a great mentor and researcher, but also a fantastic team leader who made my stay with the research group one of the best times of my life. I am genuinely thankful to Tibor for his mentor- and leadership, which have made a significant impact on my academic as well as my personal growth. His contributions have been invaluable, and I could not be more grateful.

My journey certainly would not have been the same without the amazing colleagues in Tibor's group. Thank you, (in alphabetical order) Peter Chvojka, Gareth T. Davies, Denis Diemert, Jan Drees, Amin Faez, Kai Gellert, Tobias Handirk, Raphael Heitjohann, Christian Holler, Máté Horváth, Saqib Kakvi, Rafael Kurek, Lin Lyu, Jutta Maerten, David Niehues, Marloes Venema, and Jonas von der Heyden. I thoroughly enjoyed the atmosphere in the office, whether it was your support in finding some obscure old papers I could not find (nor understand) or "enduring" yet another "pick a card, any card". I would like to especially thank Denis for making our countless train rides more enjoyable and for your overall support, may it be for research or personal matters.

I also had the pleasure of collaborating with exceptionally talented researchers. I would like to thank (in alphabetical order) Sebastian Berndt, Rongmao Chen, Denis Diemert, Thomas Eisenbarth, and Tibor Jager. Rongmao and Sebastian, I would like to thank both of you for your warm welcome and our productive collaboration during my stay in Leuven and Lübeck, respectively. It was great fun working with both of you. Of course, I would like to additionally thank Rongmao Chen for co-reviewing this thesis.

I would also like to thank my close friends, who unfortunately have scattered around Germany, mostly preferring the northern parts. I enjoy that no matter how far apart we are or how long we have not seen each other, it is always like we just met yesterday.

Zu guter Letzt möchte ich meiner Familie danken. Zuerst wäre da Kai, der nicht nur mein Bruder, sondern auch mein bester Freund ist. Danke, dass du für mich da bist und ich auf dich zählen kann, komme was wolle. Besonderer Dank geht auch an meine wundervolle Frau Sonja. Du bist meine beste Freundin sowie mein emotionaler Anker. Du gabst mir die Kraft auch in schweren Zeiten durchzuhalten. Ich könnte mir keine bessere Partnerin an meiner Seite vorstellen. Auch wenn du erst gegen Ende dieser Arbeit in Erscheinung ge-

treten bist, so möchte ich auch meinem Sohn Emil danken, für den Support vor meinem Bauch während des Schreibens diverse Abschnitte dieser Arbeit. Weiterer Dank geht an meine Schwiegermutter Jutta, die mich von Anfang an in der Familie willkommen geheißen hat. Zu guter Letzt möchte natürlich auch noch ganz besondersn meinen Eltern, Sabine und Frank, für ihre endlose und bedingungslose Unterstützung danken. Ihr erst habt diese Arbeit ermöglicht und mich zu dem gemacht, der ich heute bin, wofür ich euch ewig dankbar bleiben werde.

Abstract

With the 2013 Snowden revelations it became evident that state-level adversaries are capable and willing to compromise the integrity of widely employed cryptographic schemes through subversion attacks. These covert intrusions involve tampering with the implementation of cryptographic algorithms, allowing sensitive information to be secretly leaked to adversaries. A concerted effort emerged within the scientific community to formulate formal models capturing these attacks and devise robust countermeasures. Unfortunately, these endeavors soon encountered a sobering reality: generic attacks posed an insurmountable challenge, rendering defense against such attacks impossible without introducing additional assumptions. One countermeasure is the introduction of a trusted guardian, named the *watchdog*, entrusted with testing implementations of cryptographic schemes before their use. In this framework, a scheme is considered secure if *either* the watchdog can successfully detect subversion *or* the adversary fails to breach the scheme's security.

Despite commendable progress in developing subversion-resilient schemes in watchdog models, this thesis aims to enhance practicality by focusing on two key domains while minimizing reliance on trusted operations and avoiding the use of random oracles.

We present the first construction of public-key encryption with practical watchdogs only requiring linear testing time. Previous approaches consider asymptotic running times, which can lead to a huge testing overhead, rendering them somewhat impractical in real-world scenarios. In contrast, our model uses concrete running times, enhancing the watchdog's deployment feasibility.

We also introduce the first construction of authenticated encryption subjecting both encryption and decryption to subversion, and the first construction of digital signatures subjecting all algorithms to subversion, without relying on random oracles. Previous work either omitted critical algorithms from subversion or leaned heavily on random oracles.

While our constructions require further improvement for large-scale deployment, our contributions constitute important steps toward more practical subversion-resilient schemes.

Zusammenfassung

Die sichere Kommunikation über das Internet ist zu einem festen Bestandteil unseres Alltags geworden. Wir verlassen uns darauf, dass unsere Kreditkartendaten nicht kompromittiert werden, wenn wir im Internet einkaufen. Von modernen Instant-Messaging Apps wie WhatsApp und Signal erwarten wir, dass unsere privaten Gespräche mit Freunden und Familie vor dem Zugriff Dritter geschützt sind. All dies ist durch den Einsatz moderner Kryptographie möglich geworden. In den letzten Jahren wurde aufgedeckt, dass mächtige Angreifer diese Sicherheit, an die wir uns so lange gewöhnt haben, im großen Stil untergraben haben. Im Jahr 2013 wurde durch die Snowden-Enthüllungen die massenhafte und verdachtsunabhängige Überwachung der Telekommunikation auf globaler Ebene, vor allem durch die USA, bekannt. Insbesondere wurde festgestellt, dass die Sicherheit gängiger kryptographischer Protokolle durch die Verwendung von sogenannter Backdoors (deutsch „Hintertüren“) untergraben wurde. Diese Backdoors beschreiben Angriffe, bei denen die Implementierung kryptografischer Primitive unbemerkt verändert wird. Dadurch wurde es möglich, dass Geheimdienste unberechtigten Zugriff auf eine Kommunikation erhielten, die zuvor als sicher galt. Diese Enthüllungen stießen einer Reihe von wissenschaftlichen Arbeiten an, die diese Angriffe formal beschrieben sowie geeignete Gegenmaßnahmen entwickelten. Eine dieser Gegenmaßnahmen ist das so genannte „Watchdog“-Modell. Hierbei wird angenommen, dass ein vertrauenswürdiger Wächter (Watchdog) eine Implementierung, die mutmaßlich Backdoors enthält, gegen eine Spezifikation testet, bevor diese tatsächlich zum Einsatz kommt. Ein Angreifer ist nur dann erfolgreich, wenn er es schafft, die Tests des Watchdogs zu umgehen, und es ihm trotzdem gelingt, die Sicherheit einer Primitive zu brechen. Es wurden bereits beeindruckende Fortschritte beim Entwurf von sicheren Primitiven in diesem Modell erzielt. Dennoch gibt es nach wie vor eine Reihe von Aspekten in den bisher zur Verfügung stehenden Primitiven, welche die Anwendbarkeit einschränken.

Diese Arbeit identifiziert zwei Hauptaspekte, welche für die Anwendbarkeit backdoor-resistenter Primitiven essentiell sind. Zum Einen entwickeln wir die erste Konstruktion von Backdoor-resistenter Public-Key Verschlüsselung, die von einem Watchdog effizient getestet werden kann. Vorherige Arbeiten tätigen nur wage asymptotische Aussagen über die Laufzeit des Watchdogs, was potentiell zu einem enormen Mehraufwand beim Testen führen kann. In unserem Modell hingegen können wir für unsere Konstruktion konkrete Laufzeiten sowie Sicherheitsgarantien angeben. Zum Anderen präsentieren wir die ersten Konstruktionen für Authenticated Encryption sowie Digitale Signaturen, bei

denen alle relevanten Algorithmen vom Angreifer bereitgestellt werden, ohne idealisierte Primitiven zu nutzen. Frühere Arbeiten haben entweder kritische Algorithmen wie Entschlüsselung oder Verifikation vom Einfluss des Angreifers ausgenommen oder idealisierte Primitive wie Random Oracles verwendet.

Es bedarf noch weiterer Verbesserungen damit unsere Konstruktionen im großen Maßstab Anwendung finden können. Dennoch stellen unsere Beiträge wichtige Meilensteine auf dem Weg zu wahrlich praktischen, Backdoor-resistenten Primitiven dar.

Contents

Acronyms

ASA Algorithm-Substitution Attack
DS digital signature
HF hash function
KEM key encapsulation mechanism
MAC message authentication code
MPC multi-party computation
NIZK non-interactive zero-knowledge (proof of) knowledge
OWF one-way function
OWP one-way permutation
PKE public-key encryption
PRF pseudo-random function
RF reverse firewall
rTCR random target-collision-resistant
TCR target-collision-resistant

1 Introduction

Modern cryptography is crucial in our current digital age. We expect to do online shopping without our credit card information being compromised. We rely on instant messaging apps like WhatsApp or Signal that utilize end-to-end encryption to keep our online conversations with friends and family private. All of this is possible by means of modern cryptography. Over the last decades, researchers have designed and analyzed schemes that allow provably secure communication even in the presence of powerful adversaries. However, it was recently revealed that state-level adversaries undermined these efforts, trying to covertly access communication previously assumed to be secure.

1.1 Motivating Subversion Resilience

It has been discovered that modern cryptography is vulnerable to attacks known as *subversion* or *backdoor* attacks. In these kinds of attacks, an attacker alters the implementation of a cryptographic scheme. This enables the attacker to leak secret information via seemingly inconspicuous communication covertly. To the best of our knowledge, Petersen and Turn [PT67] were the first to tackle this topic in a scientific work in 1967. They described a class of active infiltration attacks that use "trapdoor" entry points to bypass security measures and permit direct access to data. It is important to note that the term "trapdoor" used in their work corresponds to the term "backdoor" used in cryptography nowadays. Later, Young and Yung [YY97] discussed the notion of kleptography and, thus, the possibility of backdooring cryptographic primitives. Although Young and Yung's work received attention in academic circles, it was not believed to be a feasible or realistic threat at the time. However, this dramatically changed in 2013 with the incident known as the Snowden revelations. In June 2013, documents from Edward Snowden, a former NSA (National Security Agency) contractor, were simultaneously published by The Washington Post and The Guardian [Gre], gaining public attention. These documents revealed efforts to implement a global mass surveillance network driven mainly by the United States and the United Kingdom. Barton Gellman, leading the coverage of the revelations for The Washington Post, summarized the findings as follows [Gel]:

> *"Taken together, the revelations have brought to light a global surveillance system that cast off many of its historical restraints after the attacks of Sept. 11, 2001. Secret legal authorities empowered the NSA to sweep in the telephone, Internet, and location records of whole populations."*

Part of these revelations was "Project Bullrun" which aimed to deliberately weaken the security of widely employed cryptographic schemes. As reported in [PLS13], within this project, resources were 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."*

Internal NSA memos leaked as part of the Snowden revelation indicated that the NSA was involved in the standardization process of the Dual_EC standard. It was later concluded that the Dual_EC standard was indeed backdoored [BG13], which was formally analyzed by Checkoway et al. [CMG+16].

Following these events, Juniper Networks disclosed several security vulnerabilities [Pri15] that were discovered two years after the Snowden revelations. These vulnerabilities were caused by unauthorized code in ScreenOS, the operating system of VPN routers. The target was a pseudorandom number generator used by Juniper Networks to establish VPN connections. Here, it was discovered that an unauthorized party had changed one parameter of the Dual_EC constructions, specifically altering one point of the used elliptic curve. Further research showed that it was possible to manipulate this point in a way that would enable the recovery of the entire state of the pseudorandom number generator. This would allow the party who generated this point to decrypt all VPN traffic covertly, as they would have access to all random coins produced.

The revelations made by Snowden had a significant impact on the cryptographic community. Clearly, state-level adversaries were capable of and willing to compromise the security of modern communication through backdoors.

Further, in February 2022, the Washington Post and German public broadcaster ZDF published an article on the Swiss company Crypto AG [Milb]. The article revealed that the company, which sold equipment for encryption to over 120 countries, was secretly bought in 1970 by the American and German secret agencies CIA (Central Intelligence Agency) and BND (Bundesnachrichtendienst). Over the last decades, Crypto AG deliberately weakened the security properties of the sold equipment so that the mentioned agencies could efficiently decrypt encrypted data without knowing the secret key used for encryption. This was part of Operation Rubikon, which enabled the CIA and BND to decrypt diplomatic and military transmissions from foreign states. This was used, for example, during the Falkland Wars and the US conflicts in Lybia and Panama.

1.1.1 Legal Backdoors

All examples so far highlight the *secret* rollout of backdoors. However, recently, there has been a public debate surrounding another type of backdoor, namely,

publicly enforcing backdoors through legal means. In 2018, Australia passed a law [Kar] that requires businesses to provide user information to law enforcement agencies, even if the data is protected by cryptography. This poses a particular problem for end-to-end encryption used in modern messaging apps like WhatsApp and Signal. This is because, within these apps, only the two endpoints communicating with each other can access the encrypted messages, while service providers cannot access the users' messages. However, Australia now demands access to end-to-end encrypted data, ultimately forcing businesses to introduce some form of cryptographic backdoor into their systems. This is a concern because the current state of end-to-end encryption explicitly prohibits this access and leads to two main issues:

1. The first issue with this demand is that it must be guaranteed that other parties cannot exploit this backdoor. While a backdoor opens up new criminal investigation possibilities, it is also an attractive attack vector for adversaries. Thus, backdoors deliberately weaken secure end-to-end encryption, and society has to rely on government agencies to keep the backdoor only available to them.
2. The second issue is whether society is willing to give such powerful tools to its government in the first place. This decision must be taken carefully because widely employed backdoors threaten to undermine digital security and individual freedom. This is even more important if non-democratic regimes were given access to backdoored schemes. As seen in the case of biometric data collected before the deduction of American troops in Afghanistan [Wat], this critical data fell into the hands of the Taliban.

Although the goal of law enforcement to use this additional data to fight crime is honorable, the risk of deliberately lowering the security of all communication does not seem to outweigh the benefits. Because even with lawfully acquired data, it is questionable if accessing and storing data on a large scale for some time (called *data retention*, or *Vorratsdatenspeicherung* in German) has a positive effect on crime-fighting statistics. A study [Vor] from 2011 was conducted by the "Arbeitskreis Vorratsdatenspeicherung" in Germany. They concluded that after the EU data retention directive 2006/24 was implemented, data retention "has certainly not led to a larger share of registered crime being cleared than in previous years" [Vor], taking crime statistics up to 2009 into account. A more recent study commissioned by the EU and conducted by the Belgian consulting firm Milieu [Mila] also investigated data retention. This study could not make definitive statements about the effectiveness of data retention due to the lack of data.

As a reaction to the above revelations and political developments, numerous works developed new models to formalize these previously neglected attacks and find appropriate countermeasures to this newly exposed threat. This kind of attack is the main threat this thesis aims to defend against: the *covert* change in implementations with the goal of exfiltrating secret information.

1.1.2 Scope of this Thesis

To capture the aforementioned subversion attacks, this thesis only examines attacks that exploit cryptographic schemes to leak confidential information to an outside entity. This can, for example, include using unreliable randomness during message encryption or revealing the message instead of outputting a ciphertext. We need to acknowledge that other forms of information leakage are not within the scope of this thesis. For instance, such a case could be a computer sending an encrypted message to its receiver and *also* secretly forwarding the message to the NSA encrypted under the NSA's public key without notifying the user. This issue is not solely of a cryptographic nature but should instead be dealt with at the operating system or software level. One possible approach to detect this attack is the use of *Intrusion Detection Systems* [BC10], which could be used to detect abnormal network traffic by comparing the header of a data packet with a list of permissible recipients. It is essential to understand that subversion attacks targeting non-cryptographic algorithms pose a significant security threat and must be addressed through appropriate measures, which also need to guarantee subversion-resilience. While this thesis does not cover all possible attack scenarios, we aim to provide an initial layer of protection by focusing on preventing attacks on the cryptographic building blocks themselves.

1.2 Related Work

To enable a complete assessment of the contributions of this thesis, it is crucial to have a solid understanding of the prior research in the field of subversion. We categorize prior work into three groups: Attacks, countermeasures, and miscellaneous. It is important to note that there may be some overlap, especially for the first two categories, as some works suggest both attacks and countermeasures against the proposed attacks. The miscellaneous group includes works covering various topics, all loosely connected to the topic of subversion.

Terminology. Before discussing prior work, we need to clarify some terminology. Throughout prior work and this thesis, the terms *backdoor* and *subversion* are used interchangeably. Both refer to attacks where the adversary changes the implementation of a cryptographic scheme. As in most scientific works, we also use the term *subversion* in this thesis. Sometimes, a subclass of attacks called Algorithm-Substitution Attack (ASA) are considered. These are subversion attacks where merely a single algorithm of a cryptographic scheme is substituted by the implementation of the adversary.

1.2.1 Attacks

Various works highlight the imminent threat of subversion attacks on cryptographic primitives by describing attacks and emphasizing the need for appro-

priate countermeasures. All of the following attacks have in common that a subversion attack succeeds by leaking confidential information while remaining undetected, with respect to some detection models. Different models make different statements and assumptions about detection, which we discuss in more detail in Section 1.2.2. We begin by describing generic attacks, after which we discuss more specific attacks against compositions of algorithms, harder-to-detect asymmetric subversion attacks, and attacks on Post-Quantum schemes.

Generic attacks. The first work to formally introduce subversion attacks is due to Bellare, Paterson, and Rogaway [BPR14]. They propose ASA, where an adversary substitutes the specification of an algorithm of a cryptographic primitive with its own implementation. Their work focuses on the encryption algorithm of symmetric encryption schemes, but the underlying approach can also be generalized to other schemes. They propose two attacks utilizing the random coins chosen during encryption. The first attack replaces a block cipher's initialization vector (IV) with an encryption of the symmetric encryption key. This encryption is done with a special subversion key only known to the adversary. Thus, only that adversary can decrypt the IV with regards to the special subversion key and can, therefore, obtain the secret key to decrypt other ciphertexts. This attack is undetectable as long as the adversary uses an encryption scheme whose ciphertexts are indistinguishable from random to encrypt the encryption key by the user. All modern encryption schemes used in practice have this property, highlighting the impact of this attack. The second attack called the *biased-ciphertext attack* is more versatile, as it only requires the encryption algorithm to use some random coins and can easily be applied to any other randomized cryptographic primitive. The goal of this attack is to leak the secret key bit by bit via biased ciphertexts, In detail, the adversary again has a subversion key $\tilde{K}$, which is used as a key for a pseudo-random function (PRF) F (for details on PRFs, see Section 12.5). If a message M is encrypted with the subverted encryption scheme, the scheme runs the specification of the encryption algorithm $\mathsf{Enc}(K, M)$ to obtain a ciphertext C. It then checks whether $F(\tilde{K}, C) = K[0]$, i.e., if F is evaluated on the subversion key and ciphertext, the output equals the first bit of the secret key. If not, a new ciphertext is recomputed with new randomness. This process is also often called *rejection sampling*. Thus, any adversary holding the subversion key can learn the first bit of the secret key from the ciphertext. This approach can now easily be extended to leak the secret key bit by bit via several ciphertexts. However, this only holds if the implementation can hold a state between executions to keep track of the currently leaked bit.

As described above, the powerful biased-ciphertext attacks have the downside that they require the implementation to hold state between executions. Thus, in case of state resets or stateless schemes, this attack becomes infeasible. Bellare, Jaeger, and Kane [BJK15] then presented an even stronger variety of this attack, circumventing the requirement of holding a state. Their idea is to leak *random*

secret key bits and their index via the biased-ciphertext attack. They showed that, given enough ciphertexts, the secret key can be recovered while the attack cannot be detected within their model. To summarize, their work showed that powerful subversion attacks can be carried out against *all* randomized schemes *without* the requirement that the implementation needs to hold a state.

All these attacks illustrate that using randomness can be leveraged in a subversion setting. While randomness allows to *defend* against standard adversaries, it *enables* subversion attacks. However, not only randomized schemes are vulnerable to subversion attacks. Degabriele, Farshim, and Poettering [DFP15] introduced *input-trigger attacks* for symmetric encryption schemes, where specially chosen inputs trigger an attack. In detail, the adversary prepares the implementation of the encryption algorithm such that for a special message M^* called the *trigger*, the encryption algorithm outputs the secret key as a ciphertext. Thus, any adversary obtaining encryptions of M^* can trivially break the security of the considered scheme. Note that this attack can directly be applied to other primitives. This attack can only be detected by strong security models, where a (monitoring) party gets access to transcripts of all communication of the adversary. We discuss these different models in Section 1.2.2.

So far, all of the mentioned works considered attacks on a sending party's algorithms, i.e., the party computing a ciphertext or signature. In a series of works, Armour, Farshim, and Poettering [AP22, DFP15, AP19] showed that the receiving parties, i.e., the party decrypting ciphertexts or verifying signatures can be subject to subversion attacks. Their main idea is to alter the behavior of the receiver's algorithm to leak information through artificially induced decryption errors. This means that the implementation either rejects valid ciphertexts or accepts bogus signatures. An adversary observing this behavior of the receiver can use this information to learn information about the user's secret key or simply accept invalid signatures. Third parties cannot detect these attacks in their proposed security model.

With all these powerful subversion attacks introduced in the literature, Berndt and Liśkiewicz [BL17] investigated common properties of *ASA* and showed that successful ASAs correspond to secure *stegosystems* and vice versa. *Steganography* is the concept of hiding secret messages in ordinary communication. The authors also showed lower bounds on the exfiltration rate for *universal* ASAs on symmetric encryption, i.e., an ASA that works for *all* encryption schemes where the exfiltration rate of an ASA on symmetric encryption schemes describes how many bits of secret information can be leaked via a single ciphertext. In [BL17], it was shown that no universal ASA could leak more than $\mathcal{O}(1) \cdot \log(\lambda)$ bits of information, where λ is the security parameter.

Composition and Protocols. So far, all mentioned works mainly considered a single primitive in isolation and subversion attacks against that primitive. However, in practice, usually, many different schemes work together to guarantee security. Thus, Chen, Huang, and Yung [CHY20] considered subversion of the

KEM/DEM paradigm, where secret and public-key encryption schemes work in conjunction. While previous works considered leaking secret keys in a symmetric setting, it is unclear how this translates to a public-key setting. The authors pointed out weaknesses in composed cryptographic primitives, allowing them to bypass the known logarithmic upper bound from [BL17].

Berndt, Wichelmann, Pott, Traving, and Eisenbarth [BWP+22] proposed subversion attacks on cryptographic protocols rather than individual cryptographic schemes. They showed that the highly desirable security property of forward security implies the applicability of algorithm-substitution attacks. They analyzed subversion attacks on TLS, WireGuard, and Signal and showed that for TLS and WireGuard, long-term secrets can be leaked within a few messages. In the case of Signal, the double-ratchet protocol turns out to be highly immune against ASAs. Unfortunately, choices in the implementation of Signal's multi-device support again allowed for subversion attacks. But Signal is not the only protocol susceptible to subversion attacks. Recently, Cogliati, Ethan, and Jha [CEJ23] proposed an algorithm-substitution attack on MTProto2.0, the underlying authenticated encryption scheme of Telegram's end-to-end encryption protocol. Their attacks exploit the random padding in MTProto2.0 while also showing that only minor changes can make the protocol subversion-resilient, assuming all ciphertexts can be correctly decrypted.

Asymmetric subversion attacks. So far, all mentioned attacks can be described as *symmetric* subversion attacks. The adversary uses a secret (symmetric) subversion key that is also used in the implementation. However, this approach has a downside: if a third party can access this subversion key, this party could also make use of the backdoor. With this observation, Lie, Chen, Wang, and Wang [LCWW18] and Wang, Chen, Liu, Wang and Wang [WCL+22] proposed *asymmetric* attacks on signatures and identification schemes. These attacks use asymmetric keys, just as in public-key encryption. The implementation uses the public key, while the adversary has the corresponding secret key. This way, if a third party gets access to the public key used in the implementation, this party cannot make use of the backdoor. The authors showed that this approach still allows the adversary to recover the signing key efficiently, further emphasizing the impact of subversion attacks.

Post-Quantum Schemes. With the rise of post-quantum schemes over the last years, the natural question arises if these schemes are susceptible to more sophisticated subversion attacks beyond the generic attacks presented above. Several works show that this is indeed also the case. Yang, Chen, Li, Qu, and Yang [YCL+20] proposed a practical subversion attack against the well-known lattice-based encryption scheme proposed by Regev [Reg05]. Thus, their work can be seen as a possibility result of the vulnerability of LWE (Learning With Errors)-based cryptosystem. Also, post-quantum signature schemes can also be targeted by subversion attacks, as shown by Galteland and Gjøsteen [GG19]. They analyzed all digital signature schemes submitted to NIST's

post-quantum cryptography standardization Project and showed that subliminal channels could be embedded in all submissions. Further, Jiang, Han, Zhang, Ma, and Wang [JHZ+23] showed that current post-quantum cryptosystems could also be targets of subversion attacks. They present a practical attack for general randomized algorithms with a certain structure that this randomness can be revealed to an adversary. This allows them to present a series of ASAs on public-key primitives such as public-key encryption, key-encapsulation mechanisms, and digital signatures. Their attacks highlight the threat of ASAs, as their attacks can be used to attack the ongoing NIST post-quantum standards.

Summary. Consequently, subversion attacks are possible against virtually all kinds of modern cryptographic schemes.

1.2.2 Countermeasures and Models

As the above attacks threaten the security and privacy of modern applications, different countermeasures arise to defend against these extremely powerful attacks. Before we dive into the different approaches, we emphasize that these make inherently different assumptions and are sometimes hard to compare. Each model has advantages and disadvantages, and we are convinced that a careful evaluation of a fitting model must be done for each application. We hope that this overview can help us make an adequate choice to defend against subversion attacks while also allowing us to assess the contributions of this thesis better.

In Section 1.2.1, we saw various generic attacks against cryptographic primitives. We emphasize that due to their generic nature, it is impossible to defend against these attacks without additional assumptions. In the following, we give an overview of different assumptions that enable us to defend against subversion attacks. These assumptions vary from introducing trusted parties, requiring a certain architectural design of the considered primitive, or simply introducing trusted operations. Note that while these different approaches share common ideas, it can be hard to compare them in general. However, all aim to minimize the necessary assumptions as best as possible. Unsurprisingly, there is a tendency for stronger security properties to require stronger security assumptions. Thus, in the following, we provide an overview of the approaches found in the literature to defend against these powerful subversion attacks.

Watchdogs. The *watchdog model* was introduced by Bellare, Patterson, and Rogaway [BPR14].[1] This model introduces a trusted party called the *watchdog*. This party has access to the subverted implementation of a scheme as well as to an honest specification. It is then allowed to query the subverted implementation for inputs of its choice. Usually, this means that the watchdog samples inputs uniformly at random from the input domain, asks for the subverted output, and compares it to the output of the honest specification under the same input.

[1] Although the authors do explicitly reference a watchdog model, they introduce the concept of a detection-based defense against subversion attacks.

Security, i.e., *subversion-resilience*, is then informally captured by the following two requirements:

- *either* the watchdog has a "high" chance of detecting the attack
- *or* some security guarantee for the underlying scheme holds.

This approach follows a detection-based strategy, which is motivated by the presence of a *proud-but-malicious* adversaries. This means that they would *not* engage in an attack with a high chance of getting detected. For example, there could be political repercussions if a state-level adversary is caught carrying out subversion attacks. The above notion does not capture adversaries that do not care about being detected.

Various different watchdog types were classified by Russell, Tang, Yung, and Zhou [RTYZ16]. As the different types greatly impact the meaning of security in the corresponding model, we also give an overview and highlight why we chose to use a universal offline watchdog in this thesis.

Online/Offline: A watchdog can either be offline or online. An *offline* watchdog engages in testing *before* an adversary interacts with the implementation and tries to break the security of a scheme. On the other hand, an *online* watchdog (first introduced by Degabriele, Farshim, and Poettering [DFP15]) is granted direct access to transcripts of interactions of the adversary with the subverted scheme.

Access to information: An important aspect to consider is which information the watchdog is given access to. While this is not relevant for offline watchdogs because the watchdog can, for example, choose secret keys by itself, it is essential for online watchdogs. *Omniscient* watchdogs get access to all secrets and even access to the internal state of the considered construction. Here, giving the online watchdog as little information as possible is desirable to minimize the necessary trust put in the watchdog.

Order of quantification: The order in which the watchdog and the adversary are quantified in the definition is subtle but important. There are two possibilities:

- For all adversaries, there exists a watchdog
- There exists one watchdog that defends against all adversaries

The former class of watchdogs is often used to argue that specific subversion attacks could potentially be detected by a watchdog. While useful from a theoretical perspective, this type of watchdog is of limited use in practice. The latter class of watchdogs is often called *universal* watchdogs. From a practical perspective, universal watchdogs are the desirable class of watchdogs. This is because we can guarantee security against all possible adversaries after running such a watchdog. Otherwise, one would need to run a watchdog for every possible adversary (which seems infeasible) or hope that the fear of detection by a watchdog specific to its attack prevents

an adversary from engaging in an attack. Obviously, designing a scheme that can be proven secure with a universal watchdog is significantly more complex than designing one with a non-universal watchdog.

There are many ways to combine the three properties mentioned above. We can use a non-universal omniscient online watchdog to limit an adversary's options by employing a powerful watchdog. Alternatively, we can use a universal offline watchdog and limit the necessary trust put in it as much as possible.

The offline watchdog model, which we consider in this thesis and is illustrated in Fig. 1.1, can be structured into two phases. In the first phase, the detection phase, the watchdog is aware of a specification. After the adversary provides its implementation of the scheme, the watchdog either approves or rejects the implementation by comparing the implementation to the specification. In case of approval, a second phase, called the surveillance phase, starts. The adversary tries to break the security of the scheme while interacting with the subverted implementation.

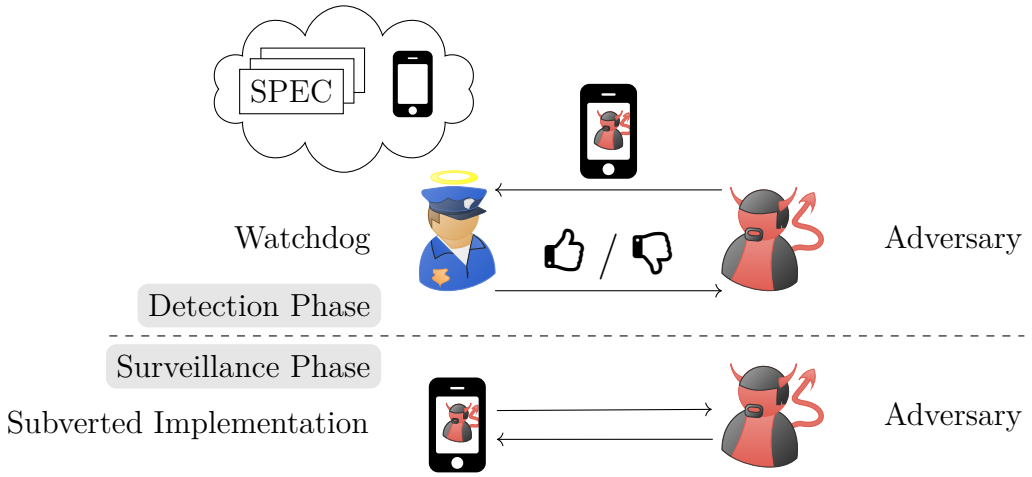


Figure 1.1: Illustration of the offline Watchdog model. In the detection phase, the adversary provides its implementation, which the watchdog tests against the specification and either accepts or rejects the implementation. Afterwards, in the surveillance phase, the adversary tries to break the security of the subverted implementation.

We now provide a brief overview of the available constructions within watchdog models of previous works. Note that we do not delve into much detail here and discuss the works mentioned in later chapters (Section 3.4, Section 12.4) more thoroughly and compare them to our approach. The first formal construction within an online watchdog model (although not called this way) was by Bellare, Paterson, and Rogaway [BPR14]. They showed that symmetric encryption with unique ciphertexts can be proven subversion-resilient if we abandon randomized schemes. Ateniese, Magri, and Venturi showed similar results for signatures [AMV15], proving that unique signatures are subversion-resilient against

subversion attacks meeting basic undetectability requirements. As a downside, both works basically do not consider subversion of the decryption or verification algorithm. Additionally, only considering deterministic schemes severely limits the possibilities for cryptographic primitives. In the face of generic attacks against randomized schemes, Russell et al. [RTYZ16, RTYZ17] proposed the *trusted amalgamation* and *split-program model*, which allows more fine-grained access to the building blocks of a primitive as well as dedicated testing of randomness generations. This new architectural assumption allowed for more primitives to be proven subversion-resilient, including (trapdoor) one-way permutations [RTYZ16], encryption schemes [RTYZ17], random oracles [RTYZ18], and signatures [CRT+19]. All of these works utilize asymptotic security definitions, where attacks are deemed unsuccessful if a watchdog can detect subversion with some tiny yet non-negligible probability. Additionally, no scheme could achieve subversion-resilience where *all* algorithms are subject to subversion *without* using idealized cryptographic primitives such as random oracles (see Section 2.3 for a deeper discussion).

Cryptographic Reverse Firewalls. A different approach uses *cryptographic reverse firewalls*, or shortly reverse firewall (RF), which was first introduced by Mironov and Stephens-Davidowitz [MS15]. RFs are proxies located between the corrupted machine and an adversary. A simple example of where a RF could be located would be a home router, which relays the communication of a desktop computer over the internet. While ordinary firewalls protect a system from outside sources, *reverse* firewalls protect from attacks launched from the implemented system itself. The main idea of RFs is that this proxy applies some operation to potentially subverted outputs (and possibly inputs from outside parties), thus removing possible biases. In contrast to watchdogs, reverse firewalls can be seen as a preventive approach rather than one for detection. For an illustration of this approach consider Fig. 1.2, where in- and outgoing messages of a subverted implementation are manipulated by the reverse firewall.

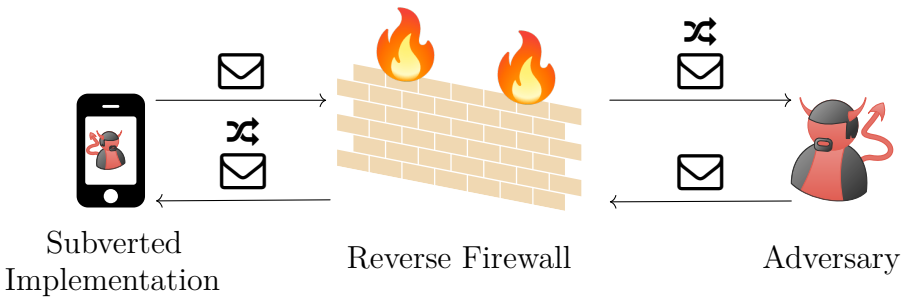


Figure 1.2: Illustration of the reverse firewall model. The reverse firewall manipulates all traffic routed through it.

In order to obtain meaningful security guarantees, it is required from RFs to

preserve security rather than *create* it. A protocol guarded by RF should thus be secure in a standard setting without a RF, while the RF provides additional security guarantees in a subversion setting. Otherwise, one could shift all the logic into the RF and thus basically remove all attack capabilities of the adversary. The reverse firewalls are not trusted as they cannot access the user's secret keys. That being said, it is assumed that the reverse firewall has access to "good" randomness, i.e., it can use uniformly random coins. Previous works showed reverse firewalls can be constructed for a variety of (re-randomizeable) cryptographic primitives such as key exchange and public-key encryption [DMS16], multiparty computation [MS15, CMY+16, CMNV22], digital signatures [AMV15] and interactive proof systems [GMV20]. RFs can easily defend against stateful subversion due to their online approach. As the RF is *always* sanitizing in- and outputs, it does not matter if the implementation changes its behavior after some specific time passed.

Bossuat, Bultel, Fouque, Onete, and van der Merwe [BBF+20] showed that reverse firewalls can also be applied to TLS-like protocols and thus be applied to secure channels (called the record layer). This is surprising since all previous works only applied to rerandomizeable primitives such as public-key encryption or signatures. As the record layer is not rerandomizeable, the authors broke with one assumption made throughout previous works. The reverse firewall is allowed to share a key with both participating parties. However, this is set up in a way such that the reverse firewall cannot access the encrypted messages.

Another work that applies reverse firewalls to non-rerandomizeable primitives is due to Dauterman, Corrigan-Gibbs, Marières, Boneh, and Rizzo [DCM+19]. The idea is that by setting up keys in a certain way and deriving randomness in a deterministic way, a party can "rerandomize" ECDSA signatures such that the output is uniformly random in the set of all valid signatures for a given message.

Another downside apart from requiring access to uniformly random coins is that RFs need and must *self-destruct* on inputs that the RF cannot handle. The notion of self-destruction was first put forward by Ateniese, Magri, and Venturi [AMV15] for the case of digital signatures. Prior works required signatures to be verifiable, i.e. the adversary would only get access to valid signatures. Ateniese, Magri, and Venturi extended the security definition and allowed invalid signatures to be handled by the RF. The problem is that an invalid signature cannot be randomized in general. The question arises of how the RF handles this situation. Unfortunately, as shown in [AMV15], the only thing the RF can do is to self-destruct, i.e., not output anything when given an invalid signature and not respond to any further queries to avoid attacks. Basically, given an invalid signature, the scheme stops working altogether. If the reverse firewall drops invalid signatures, the adversary could use this to establish a covert channel to leak secret key material. Depending on whether or not the RF responses could correlate to bits of the secret key. Additionally, the adversary could use the self-destructing behavior to carry out extremely simple Denial-of-Service attacks.

One could argue that it is possible to detect this behavior. However, one would again engage in an (online) detection approach and not solely rely on reverse firewalls for prevention.

Since the reverse firewall continuously sanitizes communication, a sudden change in the implementation's behavior does not make a difference as long as the reverse firewall can sanitize communication in the first place. Thus, reverse firewalls can defend even against stateful subversion attacks.

Self-Guarding Schemes. While most subversion-resilient schemes were proven secure in one of the two above models, another model was introduced by Fischlin and Mazaheri [FM18]. Their *self-guarding schemes* use an honest sampling phase where the scheme acts according to a specification. During this phase, samples of outputs of the primitives are collected, which can then later be used in the subversion phase, where the implementation is altered. Thus, this approach actively prevents attacks but does not need an online external party in contrast to reverse firewalls. The authors provide constructions for self-guarding CPA-secure public-key encryption from any homomorphic encryption scheme (e.g., ElGamal [ElG84]) and self-guarding symmetric-key encryption scheme from any CPA-secure scheme. In both cases, the number of messages one can encrypt is limited by the number of samples collected in the first phase. Additionally, they provide a construction for self-guarding signatures, which can be used to sign arbitrary many messages but can only defend against stateless subversion.

Immunization. Dodis, Ganesh, Golovnev, Juels and Ristenpart [DGG+15] 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 choosing 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.

1.2.3 Miscellaneous

Parameter Subversion. In this thesis, we consider subversion of the algorithms used in a cryptographic scheme. Another important aspect of cryptographic schemes is parameters. These could be, for instance, a description of a group or a common reference string or, in the case of Dual_EC , points on an elliptic curve. There is a line of research (including [BFS16, Fuc18, ALSZ21, AS21, Bag19]) investigating parameter subversion. Note that these works do not try to make it impossible to subvert parameters but rather investigate and construct schemes where all other algorithms, besides parameter generation, behave according to the specification. The question arises as to how these honestly behaving algorithms can prevent the subverted parameters from impacting the security of the considered scheme. Most works in this area focus on the subversion in the context of non-interactive zero-knowledge (proof of) knowledge

(NIZK), where the schemes use a common reference string. This random string, which all parties can access during protocol runs, is usually set up in a trusted manner.

Note that in this thesis, parameter generation can be seen as part of the key generation, as we use methods to generically sanitize key generation from the work of Russell, Tang, Yung, and Zhou [RTYZ17]. As we assume that key generation can be further split into smaller building blocks, our constructions can avoid parameter subversion. Thus, if it is known that the method used for parameter computations uses that method proposed by Russell et al. [RTYZ16] (and also used in Part II of this thesis), protection against parameter subversion is guaranteed.

Hardware Trojans. While we focus on backdoors in cryptographic primitives, prior work also considered backdoors in hardware, also known as *hardware trojans*. There is a variety of works on this topic [ABK+07, WS10, WS11, HFK+10] covering various classes of attacks as well as coming up with adequate countermeasures. Since these works operate on an entirely different abstraction level, comparing their results to ours is hard. However, similar techniques, such as more fine-grained access to building blocks and trusted operations like an XOR, seem common in both worlds.

Discussion in the Scientific Community. While the endeavors to achieve subversion-resilient were mainly theoretical, the worry of backdoors opened discussions in the scientific community. As the first example, there was a discussion on the standardization of post-quantum cryptography asking for specific treatments to prevent subversion attacks[2]. Thus, the threat of backdoors already impacts current proposals for cryptographic constructions. As a second example, an "elliptic curves seeds bounty" was recently announced[3]. In 1997, NIST standardized various elliptic curves. These curves are specified by a coefficient and a random seed value. These values are then used to derive keys transparently and verifiable. However, it is unclear how the random seeds were generated. Rumors say that the seeds were obtained by applying SHA-1 to English sentences. To outrule that these values were also chosen in a way that enables to break security (as it was in the case of Dual_EC), members of the cryptographic community announced a 12.000$ bounty for anyone who can provide the (supposedly) used English sentences for deriving the seeds.

[2]https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/WFRDl8DqYQ4?pli=1 , last accessed 25.10.2023

[3]https://words.filippo.io/dispatches/seeds-bounty/ , last accessed 25.10.2023

1.3 Contributions of this Thesis

From our perspective, an offline watchdog best captures what one intuitively expects when considering subversion-resilience:

Test an implementation once before use, and one can be sure that either the subversion attack is detected or the adversary can not break the security of the scheme.

Offline watchdogs only require one-time testing before a scheme is used and do not require access to the keys used during the actual communication. Thus, all constructions in this thesis are proven secure in an *universal offline* watchdog model, as we deem these properties essential for a watchdog to be actually usable in practice. While significant progress has been made in developing subversion-resilient schemes in the offline watchdog model, several critical properties still need further improvement. To the best of our knowledge, no construction of subversion-resilient cryptography is yet used in practice. In order to make steps towards changing this, our main research question of this thesis is:

How can we improve subversion-resilient schemes in a universal offline watchdog model to make them more practical?

While the approaches of prior work differ in their details, we identified two main aspects all works mentioned in Section 1.2.2 that achieve subversion-resilience in an offline watchdog model have in common:

1. Firstly, all prior work utilizing offline watchdogs used abstract asymptotic running times to describe the watchdog's runtime. While theoretically sound, this gives little guidance for practitioners on how long the watchdog would need to be run and allows schemes to be considered secure while enduring a potentially huge testing overhead.
2. Secondly, all prior work either excludes critical algorithms from subversion, only considers deterministic schemes, or utilizes idealized random functions (called *random oracles*, see Section 2.3) to obtain subversion-resilience when *all* algorithms are implemented by the adversary.

In this thesis, we address both these issues and summarize our contributions as follows:

Practical Watchdogs. We propose the first watchdog model with a concrete security model providing specific security guarantees for fixed runtimes of the watchdog. Within this model, we provide constructions for subversion-resilient randomness generators and subversion-resilient public-key encryption using watchdogs with linear or even constant running time. As our main contribution, we construct a subversion-resilient key encapsulation mechanism, enabling subversion-resilient public-key encryption given only a trusted XOR operation.

Complete Subversion without Random Oracles. We propose the first construction for authenticated encryption where both the encryption and decryption algorithm are provided by the adversary and the first construction of digital signatures where *all* algorithms are subject to subversion. Both constructions achieve subversion-resilience without relying on random oracles. While we do not formally include subversion of key generation for our construction of authenticated encryption, techniques from this and prior work can be applied in a straightforward fashion to also include sanitizing key generation. In both constructions, we revisit classical results from cryptography and observe that certain cryptographic schemes we use as building blocks are naturally subversion-resilient. Our results show that subversion-resilience can be achieved *without* random oracles but at the cost of larger ciphertext and signature sizes.

All of our constructions use the *trusted amalgamation* and*split-program model* proposed by Russell et al. [RTYZ16, RTYZ17], which allows the watchdog more fine-grained access to the building blocks of a primitive and decouple randomness generation from a randomized algorithm. Although this thesis does not provide constructions suitable for large-scale deployment, we believe that our contributions are important steps toward making subversion-resilient schemes more practical.

1.4 Thesis Outline

We briefly introduce some notation and foundational concepts used in this thesis in Chapter 2. Then, in Part I, we discuss the topic of subversion-resilient public-key encryption with practical watchdogs in a concrete security setting. We explore how to construct subversion-resilient one-way public-key encryption while also discussing the limitations in achieving stronger security notions. In Part II, we investigate ways to achieve subversion-resilience in an asymptotic setting under complete subversion without using random oracles. Specifically, we provide constructions of subversion-resilient authenticated encryption and digital signatures. Finally, we conclude this thesis in Part III by discussing our results and open problems.

2 Preliminaries

2.1 Notation

In this section, we shortly introduce the notation used throughout this thesis. We use $s \xleftarrow{\$} \mathcal{S}$ to denote that s is sampled uniformly at random from the set S. Let $\mathcal{A}$ be a randomized algorithm. Then we use $y \leftarrow \mathcal{A}(x)$ to denote $\mathcal{A}$ outputting y on input x. Sometimes, we explicitly reference the random coins used by $\mathcal{A}$. Then, we use $\mathcal{A}(x; r)$ to denote $\mathcal{A}$ running with input x and using the random coins r and use $\mathcal{A}(; r)$ if $\mathcal{A}$ is called without input. Let $\mathbb{N}$ be the set of the natural numbers. Then we use $[n]$ to denote the set of natural numbers of 1 to n, i.e. $\{1, 2, \ldots, n-1, n\}$. We use $\oplus$ to denote the bitwise exclusive OR (XOR) operation on two bitstrings.

In this thesis, it's important to differentiate between the *specification* and the *implementation* of a primitive Π that is being used. To distinguish between the two, we use $\widehat{\Pi}$ when referring to the primitive's specification and $\widetilde{\Pi}$ when referring to the implementation provided by the adversary.

2.2 Provable Security

As stated earlier, having a solid theoretical foundation for our security proofs is essential. Therefore, we briefly explain cryptographic reductions and the two approaches used to define security in this thesis. Our reference material for this section is primarily the book written by Katz and Lindell [KL14]. The modern approach of provable security wants to capture the notion that a primitive is secure if

- security is preserved against *efficient* adversaries, and
- efficient adversaries can break security only with a *very small* probability.

To make this approach sound, it is crucial to provide formal definitions of what *efficient* and *very small* mean in this context. We continue describing two common approaches: the *concrete approach* and the *asymptotic approach*.

The Concrete Approach. The concrete approach introduces two constants t and ε, where t denotes the running time of an adversary and ε with $0 \leq \varepsilon \leq 1$. Then, the security of a scheme Π can be (informally) stated as found in [KL14]:

> Π *is* (t, ε)*-secure if every adversary running for time at most* t *breaks the security of* Π *with probability at most* ε.

This approach of defining security (while additionally requiring a formalization of "breaking the security of Π") is useful in practice, as it captures what users are usually interested in. However, one must be careful in interpreting the results stated in the above form. For instance, it is unclear how results translate to some t' vastly greater or smaller than t. Also, with the above approach, no scheme is ever simply *secure*. While this approach allows discussion for certain choices of t and ε, it depends on the application when a (t, ε)-secure scheme can be considered *secure*.

The Asymptotic Approach. In contrast to the concrete approach, the asymptotic approach describes the running time of the adversary of the adversary as well as its success probability as functions of a security parameter $\lambda \in \mathbb{N}$. A scheme is instantiated based on this security parameter. A bigger λ usually corresponds to longer keys used within a scheme. The adversary's running time and success probability are then expressed as functions of λ. We use the abbreviation ppt (probabilistic, polynomial-time) for probabilistic algorithms algorithms with running time polynomial in λ. Additionally, we call algorithms *efficient* if their running time is polynomial in λ. For describing the success probability of adversaries, we use the notion of *negligible* functions, which is defined as follows.

Definition 1. A function f is negligible if for every polynomial $p(\cdot)$ there exists $N \in \mathbb{N}$ such that for all integers $n > N$ it holds that $f(N) < \frac{1}{p(n)}$.

Informally speaking, a function being negligible means that it vanishes faster than any inverse polynomial in λ. A simple example for a negligible function is $f(x) = \frac{1}{2^x}$. We can now put the above together and describe the notion of asymptotic security as follows, again closely following [KL14]:

Π *is secure if every ppt adversary* $\mathcal{A}$ *breaks the security of* Π *only with negligible probability.*

Note in contrast to the concrete approach, this way of phrasing security only makes statements about security for "large enough" values of λ since we consider asymptotic behavior.

Note on a Unifying Approach. We want to note that there is the possibility to unify the two above approaches by accessing the two values (t, ε) of the concrete approach as functions in the security parameter. Thus, the concrete approach could be considered more general. However, we use the two classical approaches presented above to keep the distinction clear and not overload the notation.

Cryptographic Reductions. We have explored two different methods for formalizing security. In addition, we must define what it means for a specific scheme to be secure. This is accomplished through *security experiments*, also known as *security games*. A security game outlines an adversary's capabilities and what it takes to break security. A trusted challenger performs necessary

actions like computing keys and passing them to the adversary. There are essentially two kinds of problems in security games: search and indistinguishability. A search problem involves finding specific information or a solution that meets certain criteria in a big search space. Indistinguishability problems arise when an adversary is presented with a challenge that could have been computed in one of two possible ways. The adversary's task is determining which method was used to compute the challenge with a probability much more significant than mere chance. As we now understand what security for a specific scheme means, the next step is to establish a way to formally prove that a given scheme attains the desired security property. For this, we use *cryptographic reductions*. In cryptography, we want to prove statements of the form

If some assumed to be hard problem X holds, then construction Y is secure.

Here, the hard problem X could be, for example, a mathematical problem for which no efficient algorithms are known or that some building block has a certain security property (in this thesis, we always have the latter case). The security of construction Y (which we want to prove) can be expressed in either the asymptotic or concrete setting. However, it is unclear how statements of the above form would be formally proven directly. Instead, we can make use of the contraposition of the above statement:

If Y is not secure, there exists an algorithm that can solve X.

This is far more manageable, as this is equivalent to asking to construct an adversary for "X is not true", i.e., solve a mathematical problem or break the security of a building block while using an adversary against Y. Almost all proofs in this thesis follow this generic template.

2.3 The Random Oracle Model

We refer to the random oracle model throughout this thesis, although we do not use it for our constructions. Therefore, we briefly explain this model. A common approach in modern cryptography is to construct and prove cryptographic schemes in the *random oracle model*. A random oracle is an idealized random function often used to replace a cryptographic hash function (see Section 13.7). A random oracle responds to a query with a uniformly random element of its range after being given an element from its domain. Since a random oracle is still a function, it always responds with the same output if queried multiple times with the same input. To formally capture this approach, Bellare and Rogaway [BR93] introduced the *random oracle methodology* to design practical schemes with random oracles. Their paradigm can be summarized as follows:

1. Devise a scheme in the random oracle model.
2. Prove the security of the scheme in the random oracle model.
3. Replace the oracle access to the random oracle by computations of a hash function.

Thus, the random oracle model is a *heuristic*, which can provide secure and practical schemes if a "good enough" hash function is used as a replacement. Many modern constructions with security proofs in the random oracle are used in practice, often using SHA-256 as an instantiation for the random oracle. As we discuss shortly, there are theoretical limitations to the above approach.

Limitations. Unfortunately, Canetti, Goldreich, and Halevi [CGH04] have shown that the random oracle methodology does not grant secure schemes *in general*. To show this, they build (artificial) constructions of signature and asymmetric encryption schemes, which are secure if used with a random oracle but insecure if *any* implementation of the random oracle is used. This means that a security proof in the random oracle model does not necessarily imply that the considered construction is also secure in the standard model. However, to the best of our knowledge, there is *not a single* practical scheme that has been shown to be insecure due to the use of random oracles. Additionally, a description of a truly random function is incredibly huge. This is because uniformly random values can not be compressed significantly more than simply storing all values in a list, which can be derived from Shannon's source coding theorem [Sha49a]. Therefore, even storing a local copy of a random oracle would be impractical.

From a theoretical perspective, avoiding random oracles is preferred as it minimizes the number of assumptions necessary to achieve security and uses sound assumptions for proofs instead of heuristics. Unfortunately, this often induces decreased efficiency, such as in signature or ciphertext size, compared to constructions with random oracles.

Part I

Subversion-Resilient Public-Key Encryption with Practical Watchdogs

---

In the upcoming chapters, we examine the protection of public-key encryption against subversion attacks using *practical* offline watchdogs and investigate their limitations.

Author's Contributions. In collaboration with Rongmao Chen and Tibor Jager, the author of this thesis developed a subversion-resilience model with practical watchdogs and a construction for subversion-resilient one-way public-key encryption, originally published in [BCJ21]. The key contribution of the author of this thesis was the idea to use combiners to achieve subversion-resilience, which was supported by a corresponding security proof. The aforementioned authors later discovered a subtle flaw in the proof and, with the support of Sebastian Berndt, revisited the result [BCJ21] and proved that the initially stated claims are valid for a weaker security notion and provided an impossibility result for the originally considered security notion. These new results were published in [BBCJ23].

3 Practical Watchdogs

As mentioned in Section 1.3, we aim to improve the practicality of the offline watchdog model. As a reminder, the watchdog is a trusted entity that tests a possibly subverted primitive before actual use. An adversary tries to break the security of the considered scheme while avoiding detection by the watchdog. Informally speaking, a cryptographic scheme is considered subversion-resilient if either the watchdog detects the subversion with some "high" probability or the adversary can not break the security. With this general setting in mind, our main concern is identifying the aspects that make a subversion-resilient scheme *practical*. In this part of this thesis, we focus on the watchdog's practical considerations rather than the details and properties of the considered primitives.

Powerful Watchdogs. Generally, a watchdog's capability to prevent subversion attacks increases as it becomes more powerful. To illustrate, consider a simple example. If a watchdog had unlimited time to run, it could test a cryptographic algorithm on every possible input, including those algorithms with exponentially big input spaces. As a result, it could quickly identify any subverted implementations that differ from the specified behavior, ensuring security in the subversion setting. However, the exponential running time of such a strong watchdog renders it practically useless, as secure schemes must have exponentially big input spaces (including the key and randomness space). This is because otherwise, an adversary could engage in a simple brute-force attack to, for example, obtain a secret key and trivially break security. From a practical standpoint, the offline watchdog method appears appealing since it only requires a single check rather than the continuous monitoring that online watchdogs demand. Nonetheless, as demonstrated by our previous example, even a one-time check may result in excessive testing overhead.

In particular, we are interested in techniques to construct cryptographic schemes that utilize offline watchdogs with a *concretely bounded, low running time* while achieving *strong security guarantees*. As we illustrate in Section 6.1 in more technical detail, using a weaker watchdog for testing makes it more challenging for the designer, as cryptographic components may deviate more often from their specifications. Thus, special treatments are required to achieve subversion resilience in this stronger setting. All the above points lead us to our main research question in this chapter.

What security notions can be achieved with practical offline watchdogs doing minimal testing?

Prior work by Russell, Tang, Yung, and Zhou [RTYZ16, RTYZ17] utilizes asymptotic security definitions and considers a watchdog already successful if it has a *non-negligible* probability of detecting a malicious subversion. Although theoretically sound, these asymptotic definitions do not provide clear guidance on the practical duration of the watchdog's testing, indicating a need for improvement in practical applicability. To obtain some *concrete* acceptable detection probability, this could require the offline watchdog to do a *polynomial* amount of testing, where in the worst case, the running time of the watchdog might even have to exceed the running time of the adversary, which seems not very practical. As pointed out by Russell, Tang, Yung, and Zhou [RTYZ17] and also discussed in Chapter 8, while the detection probability can be amplified via repetition, this cannot be adapted to a particular non-negligible function as long as the watchdog's running time is fixed and independent of the adversary.

Additionally, it's unclear how long the watchdog needs to run *concretely* before confidently determining whether a component is secure in an asymptotic setting. Moreover, if there are device switches or software updates and the watchdog needs to test the implementation repeatedly, the testing overhead will further increase.

Detection Probability. In addition to the issue of testing overhead, asserting a scheme secure if there is a non-negligible chance of detecting subversion might not be sufficient for some use cases. For example, consider investigative journalists or whistleblowers who could face severe consequences if their work or identity is exposed through subverted cryptography. In this scenario, it is desired that one can be *overwhelmingly confident* that the security of the considered scheme cannot be broken if the watchdog test passes. Therefore, schemes with efficient watchdogs are crucial for making subversion-resilient schemes relevant for practical use. In the theory of self-testing/correcting programs by Blum, Luby, and Rubinfeld [BLR90], limited testing by the tester/corrector is also considered a central design goal. Notably, it is required that the running time of the tester/corrector should be at most within *a constant multiplicative factor* of that of the underlying program. Thus, we believe it is also meaningful to reduce the running time of the watchdog when establishing security if we view the (possibly) subverted algorithm as a to-be-corrected program that executes in a limited time.

Our Goal. In this part, we address the practicality of subversion resilience by examining the testing overhead of the watchdog. We aim to construct subversion-resilient cryptosystems in an offline watchdog model with concrete security definitions, such that we can construct highly efficient watchdogs that are guaranteed to run significantly faster than an adversary, ideally in *linear* or even in *constant* time. We believe this is a necessary step towards making Russell et al.'s [RTYZ16, RTYZ17] watchdog model applicable in real-world applications.

3.1 Our Results

We provide the first constructions of a subversion-resilient random number generator, key encapsulation mechanism (KEM), and public-key encryption (PKE) schemes with offline watchdogs that perform only limited testing while still achieving meaningful security guarantees:

- We provide a parameterized refinement of the asymptotic watchdog model by Russell et al. [RTYZ16, RTYZ17]. More precisely, we present a generic model to capture the goal of subversion-resilience with a universal offline watchdog and trusted amalgamation for any cryptographic primitive. The model is defined with concrete security parameters so that specific bounds on the runtime of the watchdog are explicitly given. This requires a conceptual modification of the joint security experiment involving the watchdog and the adversary.
- Within this new model, we construct a simple randomness generator that is guaranteed to output *uniformly* random bits, even if the watchdog only tests the underlying component for a *constant* time. While an essential building block for our key encapsulation and public-key encryption constructions, it is also of independent interest. This randomness generator can also be used in our constructions in Part II.
- Based on this randomness generator as well as an additional trusted XOR operation, we design a subversion-resilient one-way key encapsulation mechanism using watchdogs running in *linear* time. This means that an adversary cannot compute the key encapsulated in a given ciphertext, even if it provides the implementation used for computing said ciphertext. This KEM implies a subversion-resilient one-way public-key encryption scheme with a watchdog with practical running time. The size of public keys and ciphertexts of this scheme is linear in the security parameter to achieve standard security guarantees.
- We show that prior results claiming that our KEM construction is indistinguishable are indeed impossible to achieve for some subclass of reductions.

One might wonder why exactly we chose to design subversion-resilient KEM and PKE schemes over any other primitives. The reason for this is twofold.

1. KEMs are an essential building block of modern cryptographic protocols. Having access to subversion-resilient KEM schemes hopefully enables the construction of more advanced subversion-resilient protocols. Also, KEMs play an essential role in the field of post-quantum cryptography, as demonstrated by the recent decision of NIST to standardize CRYSTALS-KYBER as the primary algorithm used for key-establishment.[1]

[1] `https://www.nist.gov/news-events/news/2022/07/pqc-standardization-process-announcing-four-candidates-be-standardized-plus`, last accessed on 17.10.2023

2. Achieving subversion-resilience at all with practical watchdogs is a challenging task. In our security model, the adversary is not given any oracle access but rather a single challenge ciphertext. This allows us to use a combiner approach to obtain security as long as one of many different building blocks is secure. Other primitives, like symmetric encryption, allow the adversary to access an oracle repeatedly. This makes the development of constructions with efficient watchdogs even harder, as we can not guarantee that oracle queries are simulated correctly. Consider Chapter 9 for a more detailed discussion on this subject.

Thus, we focus on KEMs and PKE while leaving the development of efficient watchdogs for other primitives as an interesting open problem.

3.2 Main Technical Challenge

As we use KEMs as our primary building block to construct subversion-resilient public-key encryption, let us briefly explain why constructing subversion-resilient KEMs with practical watchdogs is a challenging task. Due to the inherently randomized nature of KEMs, it is hard to guard them against subversion attacks. Given any (black-box) construction of a KEM, it can be shown that an adversary can always prepare an implementation of a KEM in such a way that the adversary obtains subverted outputs of the KEM with "high" probability. At the same time, a watchdog only has a "low" chance of detecting this attack. We refer to Section 6.1 for more technical details. Therefore, the main challenge is to design a countermeasure that prevents this generic attack while simultaneously enabling the watchdog to engage only in limited testing while still achieving security.

3.3 Our Approach

We adapt a model proposed in the work of Russell et al. [RTYZ17] called the trusted amalgamation model, which allows schemes to be split into smaller building blocks. Within our new model, we propose the following approach. We construct a subversion-resilient randomness generator, which can be tested in *constant* time to obtain random coins for our scheme. We use the classical von Neumann extractor [von51] to obtain uniformly random coins in a subversion-resilient manner. With this as a building block, we use *cryptographic combiners* [Sha49b, GHP18]. A cryptographic combiner takes as input several different candidates of a cryptographic scheme. It combines them so that the resulting scheme is secure if *at least one* of the candidates is secure. We adapt this approach and use n instantiations of a KEM in parallel, where n is a parameter that can be chosen appropriately depending on the application. We run the key generation algorithm n times in parallel to obtain n key pairs. To encapsulate

a key, we also run the Encaps algorithm n times in parallel, using a previously generated public key. This gives us n ciphertext/key pairs. While all ciphertexts are just output as the final ciphertext, the amalgamation executes an XOR function on all keys. As we see in the security analysis in Section 6.2, as long as one public key and the ciphertext under that public key are computed according to the specification, this is sufficient for our reduction to be successful.

On the KEM combiner. Giacon, Heuer, and Poettering [GHP18] proposed several KEM combiners for various security properties. However, they consider a different setting from ours. In their work, it is assumed that there are several different KEMs are available, and at least one KEM (although it is not known which) is secure by assumption. Their approach is useful, for example, to combine several different KEM based on different hardness assumptions. This way, the combined scheme is still secure, even of some hardness assumptions can be broken. Our primary approach based on the XOR combiner from [GHP18] is to use the watchdog doing limited testing to ensure that the subverted KEM is consistent with its specification at some points, which can be used to argue for the security of our construction. Thus, the testing only guarantees that at least one instance is output according to the specification, unlike in a traditional combiner setting where the security of one complete building block is given by assumption. One may wonder whether we could use combiners for public-key encryption directly to obtain a subversion-resilient PKE scheme. Unfortunately, this is not the case. From a syntactical level, most of the currently known combiners would be considered part of the trusted amalgamation in our model. However, since we consider a runtime-constrained watchdog, we can not rule out the possibility that the PKE scheme would, for instance, output the message instead of a ciphertext for a specified message m. Such an attack is known as an *input trigger attack* [DFP15] and prevents us from directly using most existing combiners or amplifiers in our setting.

Russell et al. [RTYZ16, RTYZ17] argued that subverted encryption algorithms cannot directly access the input message to rule out input trigger attacks. Thus, they blind the message with a random coin, which is output as part of the ciphertext. This approach does not work in our setting since this still requires (at least) polynomial time testing by the watchdog to obtain an overwhelming detection probability. In our construction, we bypass this type of attack by directly performing the XOR operation on the message with the output key of our designed subversion-resilient KEM.

3.4 Comparison with Prior Work

Existing Constructions in the Watchdog Model. There have been various constructions in the typical watchdog model by Russell et al. [RTYZ17]. Based on the split-program methodology [RTYZ16], several cryptographic schemes were proposed and proved to be subversion-resilient in the complete subver-

sion setting [RTYZ16, RTYZ17, RTYZ18, CRT+19]. Notably, Russell et al. [RTYZ16] constructed subversion-resilient (trapdoor) one-way permutations in the offline watchdog model (assuming fixed public input distributions). The main idea is to use a standard hash function (modeled as a random oracle) to remove the ability of an adversary to embed any potential backdoors in the function. Further, based on this general sanitizing strategy, they built a subversion-resilient signature scheme with online watchdogs and subversion-resilient pseudorandom generators (PRG) with offline watchdogs. To generically eliminate subliminal channels in randomized algorithms, Russell et al. [RTYZ17] proposed a *double-splitting* strategy where the randomness generation is carried out by mixing the output of two independent components with an immunization function. Based on this, they showed how to immunize each algorithm of an encryption scheme, including symmetric-key encryption and public-key encryption, with offline watchdogs. Chen, Huang, and Yung [CHY20] also discussed how to construct subversion-resilient key encapsulation mechanisms using this helpful strategy. Russell, Tang, Yung, and Zhou [RTYZ18] considered how to correct subverted random oracles and Chow, Russell, Tang, Yung, Zhao, and Zhou [CRT+19] further extended their results to construct subversion-resilient signature schemes in the offline watchdog model. Ateniese, Francati, Magri, and Venturi [AFMV19] relied on an additional independent (untamperable) source of public randomness and proposed a subversion-secure immunizer in the plain model for a broad class of deterministic primitives.

Some other constructions also implicitly rely on the (polynomial-testing) watchdog to achieve subversion-resilience. Bellare, Patterson, and Rogaway [BPR14] showed that symmetric encryption producing unique ciphertexts could resist subversion attacks, assuming all subverted ciphertexts are decryptable. This decryptability condition was further relaxed by Degabriele, Farshim, and Poettering [DFP15] for considering the possibility of input-trigger attacks. Note that both require an omniscient watchdog that needs to access the decryption key for verifying the ciphertext decryptability produced by the supplied implementation of the encryption algorithm. Similarly, Ateniese, Magri, and Venturi [AMV15] showed that unique signatures are subversion-resilient on the condition that all subverted signatures are valid. They also proved the unforgeability of unique signatures still holds against random message attacks when the verifiability condition is relaxed in a way similar to what is considered by [DFP15]. Their constructions require an online watchdog as the signature scheme is publicly verifiable.

All of these constructions have in common that they utilize *polynomial-time* watchdogs while defining attacks as *unsuccessful* if they can detected with *some non-negligible* probability.

Combiner and Amplification. As mentioned earlier, Giacon et al. [GHP18] proposed a variety of combiners for KEMs. There are several works constructing combiners for other primitives such as authenticated encryption with associated

data (AEAD) by Poettering and Rösler [PR20] and functional encryption by Jain, Manohar, and Sahai [JMS20]. Our approach relies on trusted amplification to build fully functional schemes from individual "secure enough" components to achieve security. Amplification is strongly related to cryptographic combiners. Given a cryptographic scheme with some "weak" security guarantee, an amplifier can construct a scheme with stronger security guarantees (for example, simply by repeated executions). Amplification has been applied to different primitives like functional encryption [JKMS20], interactive cryptographic protocols [DIJK09], and CCA-secure public-key encryption [HR05].

Subversion-Resilient Random Oracles. Although set in an asymptotic model, the results of Russell et al. [RTYZ18] with regards to sanitizing subverted random oracles share a similar underlying idea to our approach. That is, their approach also utilizes the rough idea of basing the security of their construction on the event that one of many instances of the random oracle is computed according to the specification. To prove security, they also utilize a trusted XOR. However, they also heavily utilize that only a negligible fraction of outputs of the random oracle deviate from the specification. One of their main challenges is correct simulation, as the adversary can adaptively query the subverted random oracle. As the security models considered for our subversion-resilient constructions do not include repeated oracle access, we can avoid this issue.

Outline. In Chapter 4 we describe our formal model for subversion in a concrete security setting and efficient watchdogs. In Chapter 5 we provide definitions for subversion-resilient randomness generators and prove a simple construction secure in that model. We use this as a foundation to construct subversion-resilient key encapsulation mechanisms with efficient watchdogs, assuming a trusted XOR operation is available in Chapter 6. In Chapter 7, this gives us the leverage to construct subversion-resilient public-key encryption. Finally, we show that our KEM construction can not directly be used to obtain (previously claimed) stronger security notions before discussing our results in Chapter 9.

4 Concrete Subversion Model

After getting an intuition for our approach, this chapter presents a formal model for capturing the notion of *practical* watchdogs in a concrete security setting. Our model is a variant of the security definitions from [RTYZ16, RTYZ17] and is similar in spirit to theirs but better captures our security goals for watchdogs with bounded running time.

4.1 A General Security Definition for Cryptographic Schemes

We define a *cryptographic scheme* Π as a tuple of n algorithms

$$\Pi = (\Pi_1, \ldots, \Pi_n).$$

Note that for a specific primitive, n is a fixed and usually small number. For example, for a public-key encryption scheme, which is typically defined as a tuple of algorithms $(\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$, we have $n = 3$ and

$$(\Pi_1, \Pi_2, \Pi_3) = (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt}).$$

The (abstract) security notion GOAL of Π is defined via a *security experiment* $\mathsf{Exp}^{\mathrm{GOAL}}_{\mathcal{A},\Pi}$ which involves Π and an adversary $\mathcal{A}$ and outputs a bit $b \in \{0, 1\}$. In this experiment, the adversary is asked to break the security of the Π. In the sequel, we consider security experiments based on the *indistinguishability* of two distributions or *search problems*.[1] In the former category, the adversary is given a challenge, which was computed in one of two possible ways. The adversary is then asked to distinguish these two cases. In the latter category, the adversary must compute a solution to a given problem instead of distinguishing between two possibilities.

To capture both cases formally, we generically define the *advantage function* of $\mathcal{A}$ with respect to Π and experiment Exp and a value $\delta \in \{0, 1/2\}$ indicating whether a search or indistinguishability experiment is considered, respectively, as

$$\mathsf{Adv}^{\mathrm{GOAL}}_{\mathcal{A},\Pi}(\delta) := \left|\Pr\left[\mathsf{Exp}^{\mathrm{GOAL}}_{\mathcal{A},\Pi} = 1\right] - \delta\right|.$$

[1] We consider cases where the adversary wins if it can cause an abort/fail to occur a subcategory of search problems.

When $\delta = 0$, the advantage captures search problems. This is because it equals the probability that the adversary calculates a solution to the search problem. If $\delta = 1/2$, the advantage captures indistinguishability since it is the difference to $1/2$, i.e., simply guessing. In the concrete security setting, we say the scheme Π is (t, ϵ)-*secure*, if

$$\mathsf{Adv}^{\mathrm{GOAL}}_{\mathcal{A},\Pi}(\delta) \leq \epsilon$$

for all $\mathcal{A}$ running in time at most t. Note that in later chapters, δ is clear from context, and we thus take the liberty to drop it to ease notation.

4.2 Subversion-Resilience with an Offline Watchdog

Usually, in modern cryptography, the security experiment is executed with an honest specification of the considered scheme. That is, all algorithms act in accordance with some specification. To model subversion attacks where the adversary can alter the implementation of a scheme, we thus need to choose a different approach. We follow Russell et al.'s works [RTYZ16, RTYZ17] and consider a setting where *the adversary itself* provides the implementation used in the security experiment of the considered scheme. More precisely, we consider an adversary $\mathcal{A}$ that consists of two parts $(\mathcal{A}_0, \mathcal{A}_1)$, where $\mathcal{A}_0$ produces a (possibly subverted) *implementation* and a state st

$$(\widetilde{\Pi}_1, \ldots, \widetilde{\Pi}_n, \mathsf{st}) \leftarrow \mathcal{A}_0().$$

Then $\mathcal{A}_1(st)$ engages in the security experiment Exp, which uses $\widetilde{\Pi} = (\widetilde{\Pi}_1, \ldots, \widetilde{\Pi}_n)$. The state st is used here to model that $\mathcal{A}_0$ can pass information to $\mathcal{A}_1$. In such a strong adversarial setting, achieving meaningful security without further assumptions is impossible (see Section 1.2 for an overview of generic attacks). Therefore, based on the fact that in the real world, implementations of algorithms can be tested before deployment in an application, Russell et al. [RTYZ16, RTYZ17] used an additional party called the *watchdog*. The watchdog aims to detect a possible subversion in the implementation supplied by the adversary. The watchdog WD is aware of an "honest" (not subverted) *specification* of the scheme, denoted by

$$\widehat{\Pi} = (\Pi_1, \ldots, \Pi_n),$$

and has oracle access to the implementation $\widetilde{\Pi} = (\widetilde{\Pi}_1, \ldots, \widetilde{\Pi}_n)$ produced by $\mathcal{A}_0$. The adversary $\mathcal{A}$ is only considered successful if it breaks the security of the scheme and the subverted implementation *evades detection* by the watchdog. Hence, we consider the *subversion-resilience* security experiment $\overline{\mathsf{ExpSR}}^{\mathrm{GOAL}}_{\mathsf{WD},\mathcal{A},\widehat{\Pi}}$ from Fig. 4.1, which involves a subversion adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, a watchdog WD, a protocol specification $\widehat{\Pi}$ and an underlying experiment Exp .

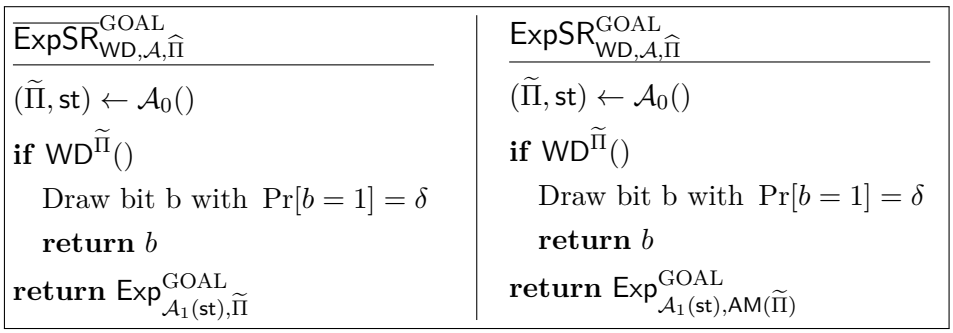


Figure 4.1: Security experiments with offline watchdog and subverted implementation. Left: Without trusted amalgamation and with $\widehat{\Pi} = (\Pi_1, \ldots, \Pi_n)$. Right: With trusted amalgamation and $\widehat{\Pi} = (\mathsf{AM}, \Pi_1, \ldots, \Pi_n)$

At the beginning of the experiment, adversary $\mathcal{A}_0$ produces a (subverted) implementation $\widetilde{\Pi}$ and a state st. The watchdog is provided oracle access to $\widetilde{\Pi}$. If WD outputs true, which means that the watchdog has detected a subverted implementation, the experiment outputs either 0 or a random bit, depending on the value δ referring to search or indistinguishability experiments. This implies that $\mathcal{A}$ has zero advantage if it outputs an implementation that WD recognizes as subverted. If $\mathsf{WD}^{\widetilde{\Pi}}() = \mathsf{false}$, then the security experiment Exp is executed, using the *adversarially-provided* implementation $\widetilde{\Pi}$ of $\widehat{\Pi}$ and with an adversary $\mathcal{A}_1(st)$ that may depend on the state produced by $\mathcal{A}_0()$. Note that the watchdog can be completely unaware of the internal workings of each algorithm as long as it has access to the in- and output behavior of the specification for all values it wants to test the implementation on.

To avoid trivial watchdogs that always output true, such that any scheme would be provably secure, we require that the watchdog is *correct* in the sense that it *always* outputs false when provided with oracle access to the actual protocol. Formally:

Definition 2. We say that WD is a *correct* watchdog for protocol specification $\widehat{\Pi}$, if

$$\Pr\left[\mathsf{WD}^{\widehat{\Pi}}() = \mathsf{false}\right] = 1.$$

All watchdogs in this thesis trivially fulfill this property. As one would intuitively expect, our watchdogs sample input according to some distribution and then compare the implementation with the outputs of the specification under the same inputs. Thus, our watchdogs never reject the protocol specification.

Note that in the above security experiment (Fig. 4.1), WD verifies $\widetilde{\Pi}$ *prior* to the experiment Exp. That is, our definition only considers offline watchdogs that simply check the supplied implementations by the adversary with no access to the full transcript of the experiment Exp. As discussed in [RTYZ16, RTYZ17],

such a watchdog is preferable over online watchdogs because it only carries out a one-time check on the implementation and does not require constant monitoring of all communication. In this thesis, we only consider a *universal* watchdog, which means it is quantified before the adversary in the security definition. Thus, for a secure scheme, there exists a *single* watchdog that defends against *all* considered adversaries.

4.3 The Split-Program Model and Trusted Amalgamation

The above offline watchdog model is meaningful and was used to establish security for some specific cryptographic primitives [RTYZ16]. However, it turns out that it is still not generic enough to achieve subversion-resilience for many other primitives. Particularly, it is known that if the user makes only black-box use of the subverted implementation of randomized algorithms, it is hopeless to eliminate a steganographic channel built on the output of algorithms [RTYZ17]. Therefore, a non-black-box model is required for general feasibility results.

Motivated by the above, Russell et al. [RTYZ17] proposed the *trusted amalgamation model*, where the specification of each algorithm is split into a constant number of components. Precisely, a scheme Π is still represented by a tuple of algorithms $(\Pi_1, \ldots, \Pi_n)$, but n may be larger than the actual number of algorithms of a protocol. The actual algorithms are then *amalgamated* by combining these underlying building blocks in a trusted way that the adversary cannot influence. Using such a somewhat relaxed model, Russell et al. [RTYZ17] showed how to generically design stego-free specifications for randomized algorithms, which are algorithms for which the adversary cannot distinguish its implementation from the specification. These play a crucial role in the construction of subversion-resilient encryption schemes in their offline watchdog model.

Note that for a meaningful construction of a subversion-resilient cryptosystem, the amalgamation should be *as-simple-as-possible*, such that most of the complexity of the involved algorithms is contained in the algorithms $\Pi = (\Pi_1, \ldots, \Pi_n)$ that may be subject to subversion. Otherwise, one could simply shift all logic into the trusted amalgamation and obtain a "secure" scheme, as we took away all attack possibilities of the adversary. To make this approach more precise, we explicitly define an *amalgamation function* AM and include it in the specification of the scheme. For instance, for a public-key encryption scheme, we would have the specification

$$\widehat{\Pi} = (\mathsf{AM}, \Pi) = (\mathsf{AM}, (\Pi_1, \ldots, \Pi_n))$$

for which

$$\mathsf{AM}(\Pi_1, \ldots, \Pi_n) = (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$$

holds.

As depicted in Fig. 4.1, the subversion-resilience security experiment with trusted amalgamation proceeds precisely as the basic experiment described above, except that the watchdog has access to all procedures

$$(\widetilde{\Pi}_1, \ldots, \widetilde{\Pi}_n, \mathsf{st}) \leftarrow \mathcal{A}_0()$$

produced by $\mathcal{A}_0$ *individually*. The security experiment is then executed with the amalgamated primitive

$$\mathsf{AM}(\widetilde{\Pi}_1, \ldots, \widetilde{\Pi}_n).$$

Following our example for public-key encryption, this would correspond to

$$\mathsf{AM}(\widetilde{\Pi}_1, \ldots, \widetilde{\Pi}_n) = (\widetilde{\mathsf{Gen}}, \widetilde{\mathsf{Encrypt}}, \widetilde{\mathsf{Decrypt}}).$$

With our security model set up, we can now define the advantage of a subversion adversary in the split program model with offline watchdog as

$$\mathsf{AdvSR}^{\mathrm{GOAL}}_{\mathsf{WD},\mathcal{A},\widehat{\Pi}}(\delta) := \Big|\Pr\Big[\mathsf{ExpSR}^{\mathrm{GOAL}}_{\mathsf{WD},\mathcal{A},\widehat{\Pi}}\Big] - \delta\Big|.$$

We now provide our full security definition, which is inspired by the works of Russell et al. [RTYZ16, RTYZ17], in the context of our general model. We aim to find a reasonable balance between allowing strong subversion adversaries while still being realistic and achievable for randomized cryptographic schemes with exponential-sized input spaces. We also slightly changed the wording in our definition compared to [RTYZ17, RTYZ16]. Instead of requiring that a watchdog *exists* that the below statement holds, we require that it is possible to *efficiently construct* a watchdog. Admittedly, all the watchdogs proposed in [RTYZ17, RTYZ16] would also fulfill a definition using our wording. However, this subtlety outrules watchdogs that in theory exist, but cannot be efficiently constructed and are therefore unfit for practical use.

Definition 3. A specification of a cryptographic protocol $\widehat{\Pi} = (\mathsf{AM}, \Pi)$ is $(t_{\mathsf{WD}}, t_{\mathcal{A}}, \varepsilon)$-*subversion-resilient*-GOAL *in the offline watchdog model with trusted amalgamation*, if one can efficiently construct a *correct* watchdog algorithm WD running in time at most t_{WD} such that for any adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ running in time at most $t_{\mathcal{A}}$ it holds that

$$\mathsf{AdvSR}^{\mathrm{GOAL}}_{\mathsf{WD},\mathcal{A},\widehat{\Pi}}(\delta) \leq \varepsilon$$

using the experiment shown in Fig. 4.1.

For convenience, we also refer to schemes as simply *subversion-resilient* in this chapter, with the understanding that they fulfill Definition 3. It might seem counterintuitive to define security with regards to the *specification* if the underlying security experiment is executed with the *subverted* implementation.

Our definition can be interpreted in the following way. While the security experiment is executed with the subverted implementation provided by the adversary, the watchdog (which is also executed in the experiment) tests the implementation with respect to that specification, and the adversary outputs subverted algorithms that syntactically follow the specification of the trusted amalgamation.

Measuring the Watchdog's Runtime. While we assign a value to the runtime of the watchdog, we still need to give this value meaning. In the following, we will measure the watchdog's runtime in terms of queries issued to the specification. Thus, our results can be interpreted in terms of testing queries with additional testing overhead due to the recomputation of the queried algorithm.

Split-Program Model. Following Russell et al. [RTYZ16], we use the *split-program model* and allow randomized algorithms to be split into a probabilistic part (the randomness generation) and a deterministic part, where all parts can be tested individually. The trusted amalgamation then feeds the generated randomness into the deterministic algorithms. For simplicity, we do not explicitly reference the split-program model in our theorems but consider it part of the trusted amalgamation model.

Stateless and Deterministic Subversion. We assume that a subverted implementation of a deterministic primitive is also deterministic. Otherwise, a subverted implementation could probabilistically deviate from the specification with some probability, where this probability could be chosen depending on the watchdog and its bounded running time so that the watchdog might fail to detect the subversion. We also remark here that we only consider *stateless* subversion in this thesis. Thus, the subverted implementation does not hold any state between different executions. Note that we are mainly interested in offline watchdogs which run in bounded time for testing. A trivial attack to evade such detection is called the *time-bomb* attack, which only becomes active when the underlying algorithm is at some specific state. Specifically, the implementations would behave honestly when they are under testing by the offline watchdog, and at a later point in time, the malicious behavior would be triggered to wake up. It is clear that such a tricky attack is impossible to detect by our considered (i.e., bounded running time) watchdog. To prevent such an attack in hardware tokens, some previous work requires a semi-online watchdog to perform testing regularly [DFS16]. Therefore, we insist that our considered model does not capture stateful subversion. Another approach to consider stateful subversion is reverse firewalls, as discussed in Section 1.2.2.

Comparison with Previous Watchdog Models The security model presented in this section is a refinement of the security models from [RTYZ17, RTYZ16]. We follow the concrete security approach instead of considering asymptotic definitions and assume the specification could be divided into an arbitrary number of components. Similar to prior work, we consider a single security experiment that can be separated into a "detection phase" and a

"surveillance phase". Note that in [RTYZ17, RTYZ16], two advantage functions are defined: one for the watchdog and another for the adversary. Security then holds if either the detection advantage of the watchdog is non-negligible or the adversaries' advantage is negligible. We change the model in the regard that we enforce that the adversary "loses" the security experiment in case the watchdog detects subversion by outputting either a random or zero bit (depending on whether a search or indistinguishability problem is considered) instead of executing the security experiment with the subverted implementation. In this way, we simplify the security definition by using a single advantage function. Our choice for this model has primarily notational advantages for achieving subversion-resilience with efficient watchdogs.

5 Subversion-Resilient Randomness Generators

After presenting our security model, we now show how to generate randomness in a way that allows a watchdog to guarantee in *constant* time that the outputs of our construction are *uniformly random*. Thus, the produced outputs are not just indistinguishable from random, but *truly* random, while the watchdog's runtime is also independent of an adversary's runtime. The following construction is then later used to generate the random coins for our subversion-resilient KEM and PKE scheme and can also be used as a building block for our constructions of Part II. Additionally, this construction is of independent interest as it is a general and efficient tool to provide uniformly random coins to any cryptographic primitives in our model.

A *randomness generator* is a randomized algorithm that outputs some strings. It is an abstraction of any means of obtaining *random* coins, even though these may be severely biased. We consider a randomness generator secure if its outputs are indistinguishable from uniformly random strings.

Definition 4. We say that a randomness generator RG is (t, ε)-indistinguishable if for any adversary $\mathcal{A}$ running in time t it holds that

$$\mathsf{Adv}^{\mathsf{IND}}_{\mathcal{A},\mathsf{RG}} = |\Pr[\mathsf{Exp}^{\mathsf{IND}}_{\mathcal{A},\mathsf{RG}} - 1/2]| \leq \varepsilon$$

with $\mathsf{Exp}^{\mathsf{IND}}_{\mathcal{A},\mathsf{RG}}$ displayed in Fig. 5.1.

Following Definition 3, we say that a randomness generator is *subversion-resilient under trusted amalgamation* if the randomness generator produces outputs that are indistinguishable from random, even in a security experiment which uses a trusted amalgamation of a subverted implementation.

Definition 5. We say the specification of a randomness generator $\widehat{\mathsf{RGSR}} = (\mathsf{AM}_{\mathsf{RG}}, \mathsf{RG}_{\mathsf{SR}})$ is $(t_{\mathsf{WD}}, t_{\mathcal{A}}, \varepsilon)$-*subversion-resilient in the offline watchdog model with trusted amalgamation*, if one can efficiently construct a *correct* watchdog WD running in time at most t_{WD}, such that for any adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ running in time at most $t_{\mathcal{A}}$ it holds that:

$$\mathsf{AdvSR}^{\mathsf{IND}}_{\mathsf{WD},\mathcal{A},\widehat{\mathsf{RGSR}}} \leq \varepsilon$$

with the used experiments shown in Fig. 5.1.

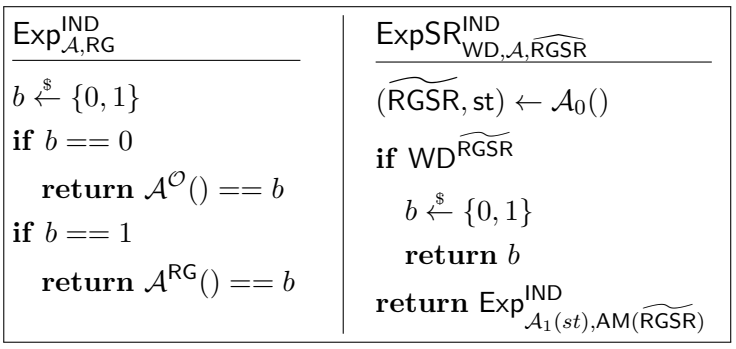


Figure 5.1: Left: Experiment for a randomness generator RG. Oracle $\mathcal{O}$ returns uniformly random strings. Right: Subversion-resilience experiment for randomness generators.

Known impossibilities. Russell et al. [RTYZ17] showed that immunizing a *single* randomness generator against subversion with an immunizing function is impossible. Essentially, they adopt the approach of subverting algorithms from [BPR14, BJK15] to randomness generators, showing that one can easily introduce a bias into a single source via rejection sampling. A subverted immunizing function can then maintain this bias. This bias may furthermore be "hidden" because detecting it requires knowledge of a secret key only known to the adversary to compute some predicate, such that a watchdog cannot detect it efficiently. In contrast, a subverting adversary may easily distinguish the subverted RG from random.

Russel et al. [RTYZ17] introduces the *double splitting* approach to overcome this general impossibility. Here, two RGs are run independently in parallel. The outputs of these two RGs are then fed into an immunization function, which may also be subverted. Russel et al. showed that if the immunization function is modeled as a random oracle, then this yields a randomness generator whose output is indistinguishable from the outputs of a non-subverted randomness generator, even for the subverting adversary. They also provide a standard-model construction of a randomness generator that outputs a single bit and a watchdog that tests whether one bit appears significantly more often than the other. Using the Chernoff bound, they argue that the watchdog will notice a bias after gathering enough samples. The randomness generator would then be run n times independently to obtain a n bit output.

5.1 Construction

We describe a new construction, which applies the "two independent RNG" approach of [RTYZ17] differently. Our construction is extremely simple and efficient to test, yet provides *perfect* random bits and does not require a random oracle.

$\mathsf{AM}_{\mathsf{RG}}(\mathsf{RG}, \mathsf{VN})$

$b_0 = b_1 := \bot$
while $b_0 == b_1$
  $b_0 \leftarrow \mathsf{RG}()$
  $b_1 \leftarrow \mathsf{RG}()$
$b := \mathsf{VN}(b_0, b_1)$
return b

Figure 5.2: The trusted amalgamation function $\mathsf{AM}_{\mathsf{RG}}$ for a randomness generator RG and a von Neumann extractor VN.

The specification $\widehat{\mathsf{RGSR}} = (\mathsf{AM}, \mathsf{RG}, \mathsf{VN})$ of our randomness generator consists of the following building blocks:

- A probabilistic algorithm RG outputs a bit $\mathsf{RG}() \in \{0, 1\}$.
- A simple binary and deterministic immunization function $\mathsf{VN} : \{0, 1\} \times \{0, 1\} \to \{0, 1\}$, which is defined as follows:

$$\mathsf{VN}(b_0, b_1) := \begin{cases} 0 \text{ if } b_0 < b_1, \\ 1 \text{ if } b_0 > b_1, \\ \bot \text{ else.} \end{cases}$$

Note that this function is the classical von Neumann extractor [von51].

The von Neumann extractor takes as input two bits with some (arbitrary and unknown, but fixed) bias and outputs a uniformly random bit as long as the two input bits are distinct from each other.

Using these two building blocks, we construct an algorithm that outputs a single bit using a trusted amalgamation. The amalgamation is essentially a simple while-loop, given in Fig. 5.2. It can be easily generalized to output n bits by calling it n times.

The amalgamation function $\mathsf{AM}_{\mathsf{RG}}$ is extremely simple. It runs RG twice independently and applies VN to the output. This is repeated in a while-loop until the output of VN is not the error symbol $\bot$, but a bit $b \in \{0, 1\}$.

Producing Outputs. Since we have $\mathsf{VN}(b_0, b_1) = \bot \iff b_0 = b_1$, whether the above algorithm produces any output at all depends on the probability that the two executions of RG computing b_0 and b_1 in the while-loop yield $b_0 \neq b_1$. Let $p := \Pr[\mathsf{RG} = 1]$, then we have

$$\Pr[b_0 \neq b_1] = 2p(1 - p).$$

Hence, it takes an expected $(p(1 - p))^{-1}$ executions of RG and $(2p(1 - p))^{-1}$ executions of VN to generate an output bit. For instance, if RG is truly random, we would have 4 expected executions of RG and 2 of VN. Even if RG is a rather

bad random number generator, say with $p = 1/4$, then one would expect about $5 + 1/3$ executions of RG and $2 + 2/3$ of VN. Although not relevant for security, the case that $p = 0$ or $p = 1$ would lead to our construction never producing any outputs, as the RG only outputs either 0 or 1. Additional testing can circumvent this case by checking if RG outputs both 0 and 1 at all.

5.2 Security

The proof that $\widehat{\mathsf{RGSR}}$ is a subversion-resilient randomness generator uses the fact that the adversary provides a *single* implementation $\tilde{\mathsf{RG}}$, which is then queried *twice* by the trusted amalgamation. Therefore, the two bits b_0, b_1 are computed *independently* in every while loop and are identically distributed. The watchdog only has to test the implementation of the von Neumann extractor on all four possible inputs. It is not necessary to test the randomness generator's implementation at all to achieve security.

Theorem 1. *The specification* $\widehat{\mathsf{RGSR}}$ *as defined above is* $(\mathcal{O}(1), t_{\mathcal{A}}, 0)$*-subversion-resilient in the offline watchdog model with trusted amalgamation.*

Note that the theorem asserts that we can construct a *constant-time* watchdog. It is independent of the runtime of the adversary or any bias possibly embedded in the subverted implementation of RG. Also, as our construction always outputs uniformly random bits, our construction is even information-theoretically secure, i.e. secure even against adversaries with unlimited runtime.

Proof. The only component we test is the implementation of $\mathsf{VN} : \{0,1\} \times \{0,1\} \to \{0,1\}$, a very simple function with only four possible inputs. Therefore, it can be checked on all possible inputs in constant time. The watchdog WD runs a given implementation $\widetilde{\mathsf{VN}}$ on all four possible inputs and checks the correctness of the output with the specification.

If $\widetilde{\mathsf{VN}}$ deviates from the specification on any input, the watchdog will detect this with probability 1. Thus, this would immediately lead to an advantage of 0 for the adversary.

Provided that $\widetilde{\mathsf{VN}}$ implements VN correctly and using the fact that the two bits $b_0, b_1 \leftarrow \tilde{\mathsf{RG}}()$ are computed *independently* in every while loop, we obtain that when $\tilde{\mathsf{RG}}$ outputs a bit b then we have

$$\Pr[b = 0] = \Pr[b_0 = 0 \wedge b_1 = 1] = \Pr[b_0 = 1 \wedge b_1 = 0] = \Pr[b = 1]$$

and thus $\Pr[b = 0] = \Pr[b = 1] = 1/2$, again leading to an advantage of 0 for the adversary.

□

The main advantage of our proposed RG is that it achieves perfect security using a *constant*-time watchdog. Russell et al. [RTYZ17] also described several

alternative approaches to purify randomness in the standard model. Below, we provide more discussion about their approaches. It is worth mentioning that in [AFMV19], Ateniese et al. proposed a different approach to eliminating the requirement of random oracles by essentially relying on an additional independent and untamperable source of public randomness.

Simple Multi-Splitting. The first approach proposed in [RTYZ17] is simple multi-splitting, which means that n copies of RG (each outputting a single bit) are run, and all outputs are concatenated and output. The main problem with this approach is that RG has to be tested many times. Otherwise, the watchdog is unable to notice a small but non-negligible bias.

More efficient Construction using Randomness Extractors. The second approach uses randomness extractors but in a totally different way. Precisely, it is observed that a watchdog making $\mathcal{O}(n^{2c})$ queries can verify that the output of each RG has at least $c \log n$ bits of entropy (for some constant c). Thus, Russell et al. [RTYZ17] proposed to run RG for $\log n$ times to obtain a random string of length $\log n$, which is then used as a seed for a randomness extractor. This extractor can then obtain more random bits from RG, which can be expanded with a pseudorandom generator (PRG) afterwards. In this case, a watchdog would have to check RG for entropy and recompute all calculations of the extractor and the PRG to check if these components follow their specifications.

A disadvantage of our construction of RG is that we do not have a strict upper bound on its running time, but only expected bounds. However, this is a minor disadvantage since the lack of functionality will be an automated way to detect subversion that introduces a too-heavy bias in RG. The efficiency can be improved by conducting additional testing. For instance, the watchdog could also test RG and check for a too-heavy bias p that would significantly harm performance. Note, however, that even a particularly bad underlying randomness generator RG, which always outputs a constant 0, would only harm correctness but not the security of RG.

6 Subversion-Resilient Key Encapsulation Mechanisms

Key encapsulation mechanisms are techniques to transport symmetric cryptographic key material using public-key cryptography securely. Keys are uniformly generated and then encapsulated under a public key. Only parties possessing the secret key can decapsulate ciphertexts to retrieve the encapsulated key. The idea of KEMs is that it is hard for an adversary to obtain information on the encapsulated key without access to the secret key. KEMs are an essential building block for our construction of public-key encryption, as messages can be encrypted by applying an XOR to the message and the encapsulated key. We use the following standard definitions for key encapsulation mechanisms and their security.

Definition 6. A *key encapsulation mechanism* $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ consists of three algorithms with the following syntax.

- $\mathsf{Gen}()$: The randomized *key generation algorithm* outputs a key pair $(\mathsf{sk}, \mathsf{pk})$.
- $\mathsf{Encaps}(\mathsf{pk})$: The randomized *encapsulation algorithm* takes as input a public key pk. It outputs a key $K \in \mathcal{KS}$, where $\mathcal{KS}$ is called the *key space* defined by pk (either implicitly or explicitly), and a ciphertext C.
- $\mathsf{Decaps}(\mathsf{sk}, C)$: The deterministic *decapsulation algorithm* takes a secret key sk and a ciphertext C. It outputs a key $K \in \mathcal{KS}$ or a distinguished error symbol $\perp$.

A KEM is *indistinguishable* or IND-secure if no adversary can *distinguish* whether it is given a public key, a ciphertext and the key encapsulated in that ciphertext, or a public key, a ciphertext and a random key from the key space. Intuitively speaking, this property captures that encapsulated keys look random to adversaries.

Definition 7. We say that $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ is $(t_{\mathcal{A}}, \varepsilon)$-*indistinguishable* if for any adversary $\mathcal{A}$ running in time at most $t_{\mathcal{A}}$ it holds that

$$\mathsf{Adv}^{\mathsf{IND}}_{\mathcal{A},\mathsf{KEM}} := |\Pr[\mathsf{Exp}^{\mathsf{IND}}_{\mathcal{A},\mathsf{KEM}}] - 1/2| \leq \varepsilon$$

with $\mathsf{Exp}^{\mathsf{IND}}_{\mathcal{A},\mathsf{KEM}}$ as defined in Figure 6.1.

A weaker notion is *one-way* security. In some scenarios it may be sufficient that an adversary could potentially distinguish between encapsulated and random keys, as long it can not *compute* encapsulated keys with probability significantly better than guessing. Thus, the corresponding security experiments

$\mathsf{Exp}^{\mathsf{IND}}_{\mathcal{A},\mathsf{KEM}}$

$(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{Gen}()$
$K_0 \xleftarrow{\$} \mathcal{KS}$
$(C^*, K_1) \leftarrow \mathsf{Encaps}(\mathsf{pk})$
$b \xleftarrow{\$} \{0,1\}$
$b_{\mathcal{A}} \leftarrow \mathcal{A}(\mathsf{pk}, K_b, C^*)$
if $b_{\mathcal{A}} == b$
  return 1
else
  return 0

$\mathsf{Exp}^{\mathsf{OW}}_{\mathcal{A},\mathsf{KEM}}$

$(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{Gen}()$
$(C^*, K^*) \leftarrow \mathsf{Encaps}(\mathsf{pk})$
$(K) \leftarrow \mathcal{A}(\mathsf{pk}, C^*)$
if $K^* == K$
  return 1
else
  return 0

Figure 6.1: Left: Indistinguishability experiment for key encapsulation mechanisms. Right: One-way experiment for key encapsulation mechanisms.

$\mathsf{ExpSR}^{\mathrm{GOAL}}_{\mathsf{WD},\mathcal{A},\widehat{\mathsf{KEMSR}}}$

$(\widetilde{\mathsf{KEMSR}}, \mathsf{st}) \leftarrow \mathcal{A}_0()$
if $\mathsf{WD}^{\widetilde{\mathsf{KEMSR}}}()$
  Draw bit b with $\Pr[b = 1] = \delta$
  return b
return $\mathsf{Exp}^{\mathrm{GOAL}}_{\mathcal{A}_1(st),\mathsf{AM}(\widetilde{\mathsf{KEMSR}})}$

Figure 6.2: Subversion-resilience GOAL experiment for key encapsulation mechanisms where $\mathrm{GOAL} \in \{\mathsf{IND}, \mathsf{OW}\}$ and $\delta \in \{0, 1/2\}$, accordingly.

hand the adversary a public key and ciphertext, which wins if it responds with the encapsulated key.

Definition 8. We say that $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ is $(t_{\mathcal{A}}, \varepsilon)$-*one-way* if for any adversary $\mathcal{A}$ running in time at most $t_{\mathcal{A}}$ it holds that

$$\mathsf{Adv}^{\mathsf{OW}}_{\mathcal{A},\mathsf{KEM}} := |\Pr[\mathsf{Exp}^{\mathsf{OW}}_{\mathcal{A},\mathsf{KEM}}]| \leq \varepsilon$$

with $\mathsf{Exp}^{\mathsf{OW}}_{\mathcal{A},\mathsf{KEM}}$ as defined in Figure 6.1.

We define *subversion-resilient* KEMs by adapting Definition 3 to KEMs.

Definition 9. We say that the specification $\widehat{\mathsf{KEMSR}} = (\mathsf{AM}_{\mathsf{KEM}}, \mathsf{KEMSR})$ of a key encapsulation mechanism is $(t_{\mathsf{WD}}, t_{\mathcal{A}}, \varepsilon)$ *subversion-resilient one-way, or subversion-resilient indistinguishable, respectively, in the offline watchdog model with trusted amalgamation*, if one can efficiently construct a correct watchdog

WD running in time at most t_{WD} such that for any adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ running in time at most $t_{\mathcal{A}}$ it holds that:

$$\mathsf{AdvSR}^{\mathrm{GOAL}}_{\mathsf{WD},\mathcal{A},\widetilde{\mathsf{KEMSR}}}(\delta) \leq \varepsilon$$

with $\mathrm{GOAL} \in \{\mathsf{OW}, \mathsf{IND}\}$ and $\delta \in \{0, 1/2\}$, accordingly.

6.1 Main Technical Challenge

As our formal definitions are set up, let us illustrate our main technical challenges in achieving subversion-resilient KEMs in more detail.

The Difficulty of Recognizing a Subverted KEM with a Watchdog. Let $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ be the specification of a legitimate (i.e., not subverted) KEM. Let $R_{\mathsf{gen}}, R_{\mathsf{enc}}$ be the randomness spaces of Gen and Encaps. Let F be the deterministic function parameterized by a KEM which takes as input randomness $(r, s) \in R_{\mathsf{gen}} \times R_{\mathsf{enc}}$ for Gen and Encaps, respectively, and then computes

$$(\mathsf{pk}, \mathsf{sk}, C, K) = F_{\mathsf{KEM}}(r, s)$$

with

$$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(; r) \quad \text{and} \quad (C, K) \leftarrow \mathsf{Encaps}(\mathsf{pk}; s).$$

Now let $\widetilde{\mathsf{KEM}} = (\widetilde{\mathsf{Gen}}, \widetilde{\mathsf{Encaps}}, \widetilde{\mathsf{Decaps}})$ be a (possibly subverted) implementation of the algorithms of KEM. We observe that if

$$F_{\mathsf{KEM}}(r, s) = F_{\widetilde{\mathsf{KEM}}}(r, s) \tag{6.1}$$

on all $(r, s) \in R_{\mathsf{gen}} \times R_{\mathsf{enc}}$, then the security of the originally specified scheme KEM implies security of $\widetilde{\mathsf{KEM}}$, if (r, s) are indeed chosen randomly. If Equation (6.1) holds, then the implementations $\widetilde{\mathsf{Gen}}$ and $\widetilde{\mathsf{Encaps}}$ agree with the specification on all inputs, and these are the only algorithms of $\widetilde{\mathsf{KEM}}$ used in the security experiment. The same holds if Equation (6.1) holds for all but a *negligible* fraction of all r, s, since the probability that a security experiment chose (r, s) such that Equation (6.1) does *not* hold would be negligible. Unfortunately, we are not able to test efficiently whether Equation (6.1) holds for all but a negligible fraction of all r, s. Let

$$\mathsf{Neq} := \left\{(r, s) : F_{\mathsf{KEM}}(r, s) \neq F_{\widetilde{\mathsf{KEM}}}(r, s)\right\} \tag{6.2}$$

and

$$\mathsf{Eq} := \left\{(r, s) : F_{\mathsf{KEM}}(r, s) = F_{\widetilde{\mathsf{KEM}}}(r, s)\right\}. \tag{6.3}$$

That is, Neq contains all "bad" randomness values such that Equation (6.1) does *not* hold with respect to $(\mathsf{KEM}, \widetilde{\mathsf{KEM}})$. Analogously, Eq contains all "good"

randomness values for which Equation (6.1) does hold. Since Neq and Eq are disjoint sets, we have

$$\mathsf{Neq} \cup \mathsf{Eq} = R_{\mathsf{gen}} \times R_{\mathsf{enc}} \quad \text{and} \quad \mathsf{Neq} \cap \mathsf{Eq} = \emptyset.$$

Note that testing whether $|\mathsf{Neq}|/2^\lambda$ is negligible by repeatedly running F on different inputs takes exponential time. Even if we granted the watchdog a very large running time by allowing it to evaluate F a polynomial $P(\lambda)$ of times, where $P(\lambda)$ is large, this watchdog could still fail to recognize that $|\mathsf{Neq}|$ is non-negligible with very high probability.

For instance, suppose that $\mathsf{Neq} \subset R_{\mathsf{gen}} \times R_{\mathsf{enc}}$ were a random subset of size $|\mathsf{Neq}| = |R_{\mathsf{gen}} \times R_{\mathsf{enc}}|/P^2(\lambda)$. Then a watchdog running in time P would detect the subversion only with probability

$$\Pr[\mathsf{WD} \text{ detects subversion}] = \Pr[\mathsf{WD} \text{ samples from } \mathsf{Neq} \text{ at least once}] \leq \frac{1}{P(\lambda)}$$

where the inequality results from applying the union bound [Boo47](also known as Boole's inequality) as well as using that sampling from Neq occurs with probability $1/P^2(\lambda)$ when sampling once. Simultaneously, the scheme would be insecure since we have

$$\frac{|R_{\mathsf{gen}} \times R_{\mathsf{enc}}| \cdot \frac{1}{P^2(\lambda)}}{|R_{\mathsf{gen}} \times R_{\mathsf{enc}}|} = \frac{1}{P^2(\lambda)}$$

and thus the security experiment would choose "bad" randomness $(r, s) \in \mathsf{Neq}$ with significant probability $\frac{1}{P^2(\lambda)}$. We want to emphasize that this attack allows the adversary to adjust the probabilities freely, increasing their success probability or probability of evading detection while decreasing the other.

Our Approach to Overcoming the Technical Difficulties. We build a subversion-resilient one-way KEM $\mathsf{KEMSR} = (\mathsf{GenSR}, \mathsf{EncapsSR}, \mathsf{DecapsSR})$ based on a regular $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ in the following way. For ease of exposition, we describe a less efficient scheme here. Our actual scheme can be instantiated much more efficiently by trading the size of keys and ciphertexts for a reasonably increased running time of the watchdog. For more details, see Section 6.3.

A key pair $(\mathsf{pk}, \mathsf{sk}) = ((\mathsf{pk}_i)_{i\in[\lambda]}, (\mathsf{sk}_i)_{i\in[\lambda]})$ of KEMSR consists of λ many public keys

$$(\mathsf{pk}_1, \mathsf{sk}_1), \ldots, (\mathsf{pk}_\lambda, \mathsf{sk}_\lambda) \leftarrow \mathsf{Gen}()$$

of KEM. In order to generate an encapsulated key, we run $(C_i, K_i) \leftarrow \mathsf{Encaps}(\mathsf{pk}_i)$ for all $i \in [\lambda]$, and return

$$(C, K) := ((C_1, \ldots, C_\lambda), K_1 \oplus \cdots \oplus K_\lambda).$$

Here, the $\oplus$ is part of the trusted amalgamation function.

The security proof against subversion in the watchdog model of this construction is based on the following idea. and let Neq be as in Equation (6.2). We construct a reduction to the security of the underlying KEM that goes through if the simulated security experiment generates *at least one* pair (pk_i, C_i) using "good" randomness, that is, choosing $(r, s) \xleftarrow{\$} R_{\mathsf{gen}} \times R_{\mathsf{enc}}$ such that $(r, s) \in \mathsf{Eq}$.

Note that even if the underlying KEM is heavily subverted, for instance, such that half of all randomness tuples (r, s) are "bad" and we have

$$|\mathsf{Neq}| = \frac{|R_{\mathsf{gen}} \times R_{\mathsf{enc}}|}{2} \tag{6.4}$$

we would still have

$$\Pr\Big[(r, s) \notin \mathsf{Neq} : (r, s) \xleftarrow{\$} R_{\mathsf{gen}} \times R_{\mathsf{enc}}\Big] = 1/2.$$

Therefore, the probability that the experiment generates at least one pair (pk_i, C_i) using "good" randomness is $1 - \frac{1}{2}^{\lambda}$, which is almost certain, up to a negligibly small probability. If the adversary produces a subverted implementation where $|\mathsf{Neq}|$ is larger than in Equation (6.4), then, of course, it becomes less likely that the experiment chooses "good" randomness. However, already for $|\mathsf{Neq}|$ as in Equation (6.4) we are able to detect the subversion almost certainly, and with a very efficient watchdog. Concretely, a watchdog running F λ times and comparing the output to the specification is able to detect the subversion with overwhelming probability $1-\frac{1}{2}^{\lambda}$. This detection probability increases with larger $|\mathsf{Neq}|$.

In order to ease the notation and make our approach clear, the above sketch of our construction uses λ many executions of the KEM procedures and a watchdog that tests each algorithm λ many times. As we see in our construction, these parameters can be adjusted to allow for tradeoffs between ciphertext (and key) size and runtime of the watchdog.

6.2 Our Proposed KEM

Thus, armed with the necessary definitions and intuition of the upcoming challenge, we can finally provide our construction of subversion-resilient KEMs.

Construction. We are now ready to describe our construction with these definitions in place. Let $\widehat{\mathsf{RGSR}} = (\mathsf{AM}_{\mathsf{RG}}, \mathsf{RG}_{\mathsf{SR}})$ be the specification of a subversion-resilient randomness generator. Further, let $n > 0$ be an arbitrary constant, allowing us to adjust the construction. Since we focus on the key encapsulation mechanism in this section, we use

$$\mathsf{RG}() := \mathsf{AM}_{\mathsf{RG}}(\mathsf{RG}_{\mathsf{SR}})$$

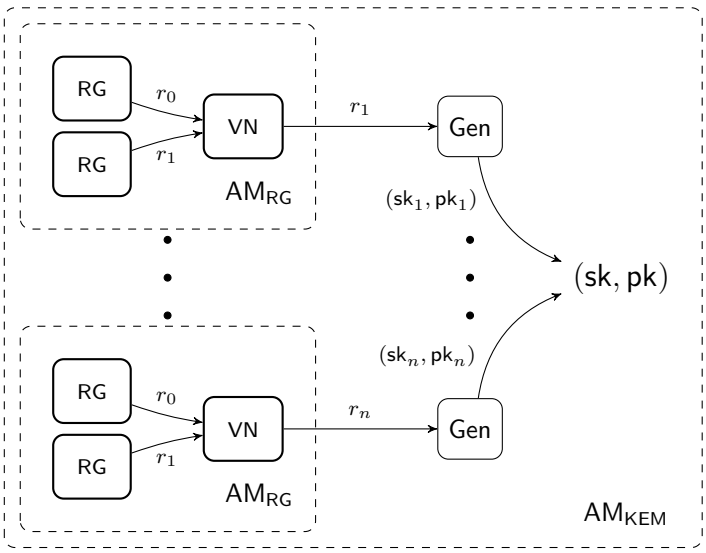


Figure 6.3: Subversion-resilient KEM: Key generation algorithm.

to simplify notation. Let (Gen, Encaps, Decaps) be a key encapsulation mechanism. From these building blocks, we define a specification of a subversion-resilient key encapsulation mechanism

$$\widehat{\mathsf{KEMSR}} = (\mathsf{AM}_{\mathsf{KEM}}, \mathsf{KEMSR}) = (\mathsf{AM}_{\mathsf{KEM}}, (\mathsf{RG}_{\mathsf{SR}}, \mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps}))$$

where the trusted amalgamation $\mathsf{AM}_{\mathsf{KEM}}$ defines algorithms (GenSR, EncapsSR, DecapsSR) as follows.

- $\mathsf{GenSR}()$: Compute $r_i \leftarrow \mathsf{RG}()$ and $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{Gen}(; r_i)$ for all $i \in [n]$ and output
$$\mathsf{pk} := (\mathsf{pk}_i)_{i\in[n]} \quad \text{and} \quad \mathsf{sk} := (\mathsf{sk}_i)_{i\in[n]}.$$
See Fig. 6.3 for an illustration.
- $\mathsf{EncapsSR}(\mathsf{pk})$: On input $\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ compute $r_i \leftarrow \mathsf{RG}()$ and $(C_i, K_i) \leftarrow \mathsf{Encaps}(\mathsf{pk}_i; r_i)$ for all $i \in [n]$ and output
$$C := (C_1, \ldots, C_n) \quad \text{and} \quad K := K_1 \oplus \cdots \oplus K_n.$$
See Fig. 6.4 for an illustration.
- $\mathsf{DecapsSR}(C, \mathsf{sk})$: On input $\mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ and $C = (C_1, \ldots, C_n)$ compute $K_i = \mathsf{Decaps}(\mathsf{sk}_i, C_i)$ for all $i \in [n]$. If there exists $i \in [n]$ such that $K_i = \perp$, then output $\perp$. Otherwise output
$$K = K_1 \oplus \cdots \oplus K_n.$$

The trusted amalgamation function $\mathsf{AM}_{\mathsf{KEM}}$ essentially consists of simple loops with n independent iterations of calls to the underlying RG and KEM procedures,

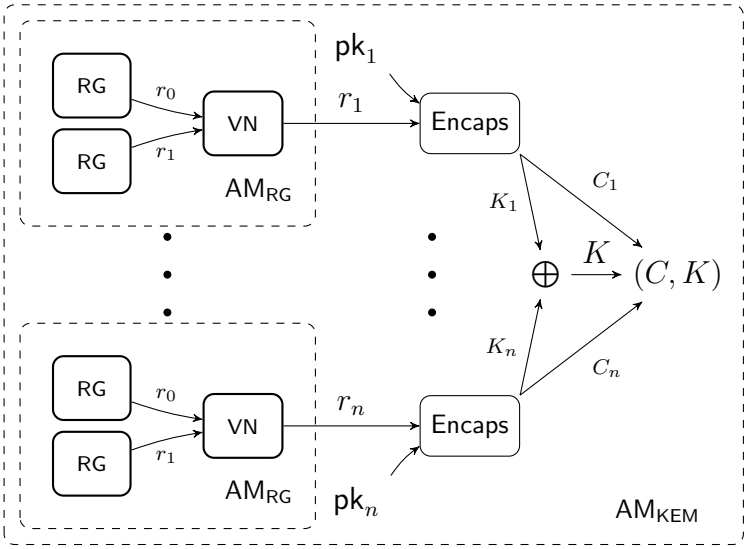


Figure 6.4: Subversion-resilient KEM: Encapsulation algorithm.

plus a simple XOR function. A trusted XOR was also used in [RTYZ17] to handle large message spaces for public-key encryption.

Security Analysis.

Theorem 2. *Let* KEM *be a* $(t_{\mathcal{A}}, \varepsilon)$ *one-way key encapsulation mechanism and* $\widehat{\mathsf{RGSR}}$ *be the specification of a* $(\mathcal{O}(1), t_{\mathcal{B}}, 0)$ *subversion-resilient randomness generator. Then* $\widehat{\mathsf{KEMSR}}$ *as defined above with parameters* $n, n_{\mathsf{WD}} > 0 \in \mathbb{N}$ *is* $(t_{\mathsf{WD}}, t'_{\mathcal{A}}, \varepsilon')$ *subversion-resilient one-way in the offline watchdog model with trusted amalgamation assuming a trusted XOR operation, with*

$$t_{\mathsf{WD}} \in \mathcal{O}(n_{\mathsf{WD}}), \qquad t'_{\mathcal{A}} \in \mathcal{O}(t_{\mathcal{A}} + t_{\mathcal{B}} + n),$$

$$\varepsilon' \leq \max\left\{2^{-n_{\mathsf{WD}}}, 2\varepsilon + \left(\frac{n_{\mathsf{WD}}}{n_{\mathsf{WD}} + n}\right)^{n_{\mathsf{WD}}} \cdot \left(1 - \frac{n_{\mathsf{WD}}}{n_{\mathsf{WD}} + n}\right)^{n}\right\} \ .$$

Proof. The following notation and helper functions are useful for the following proof. Let $R_{\mathsf{gen}}, R_{\mathsf{enc}}$ denote the randomness space of the algorithms Gen and Encaps, respectively. Let F_{KEM} be the deterministic function parameterized by a key encapsulation mechanism $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ which takes as input randomness $(r, s) \in R_{\mathsf{gen}} \times R_{\mathsf{enc}}$ for Gen and Encaps, respectively, and then computes

$$(\mathsf{pk}, \mathsf{sk}, C, K) = F_{\mathsf{KEM}}(r, s) \tag{6.5}$$

with

$$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(; r) \quad \text{and} \quad (C, K) \leftarrow \mathsf{Encaps}(\mathsf{pk}; s) \ .$$

For KEM (which is part of the specification $\widehat{\mathsf{KEMSR}}$) and a corresponding implementation $\widetilde{\mathsf{KEM}}$ we can now define sets Neq and Eq as

$$\mathsf{Neq} := \left\{(r,s) : F_{\mathsf{KEM}}(r,s) \neq F_{\widetilde{\mathsf{KEM}}}(r,s)\right\} \tag{6.6}$$

and

$$\mathsf{Eq} := \left\{(r,s) : F_{\mathsf{KEM}}(r,s) = F_{\widetilde{\mathsf{KEM}}}(r,s)\right\} \quad . \tag{6.7}$$

Hence, set Neq contains all "bad" randomness values where the implementation deviates from the specification, and Eq contains all "good" randomness values where specification and implementation match. Since Neq and Eq are disjoint sets, we have

$$\mathsf{Neq} \cup \mathsf{Eq} = R_{\mathsf{gen}} \times R_{\mathsf{enc}} \quad \text{and} \quad \mathsf{Neq} \cap \mathsf{Eq} = \emptyset \ .$$

Watchdog Construction. We construct a universal offline watchdog WD, which proceeds as follows.

1. First WD runs the watchdog for $\widehat{\mathsf{RGSR}}$ as a subroutine. If this algorithm outputs true, then WD outputs true. Otherwise, WD proceeds.
2. Then, for $i \in [n_{\mathsf{WD}}]$, WD picks $(r_i, s_i) \xleftarrow{\$} R_{\mathsf{gen}} \times R_{\mathsf{enc}}$ uniformly at random and checks whether

   $$F_{\mathsf{KEM}}(r_i, s_i) = F_{\widetilde{\mathsf{KEM}}}(r_i, s_i)$$

   holds where F is as defined in Equation (6.5). If any of these checks fail, then the watchdog outputs true. Otherwise, it outputs false.

Note that the above watchdog performs only a constant number of queries to check RG plus n_{WD} many evaluations of each Gen and Encaps.

Security analysis. Before we dive into the analysis, let us start with a general bound on the success probability of adversaries that help us out later. We will use that the fraction of randomness inputs eq to $\mathsf{Gen} \times \mathsf{Encaps}$ is at least $1/2$ for any adversary $\mathcal{A}$ winning the subversion security game with probability at least $1/2^{n_{\mathsf{WD}}}$. This is because any implementation winning the subversion experiment needs to pass the watchdog's test beforehand, and in both cases, uniformly random coins are provided to the subverted building blocks. Thus, we naturally obtain

$$\Pr[\mathsf{WD}^{\widetilde{\mathsf{KEMSR}}}] \geq \Pr[\mathsf{ExpSR}^{\mathsf{OW}}_{\mathsf{WD},\mathcal{A},\widehat{\mathsf{KEMSR}}} = 1].$$

Now, assume that $\mathsf{eq} < 1/2$. Then, we can conclude that

$$\Pr[\mathsf{ExpSR}^{\mathsf{OW}}_{\mathsf{WD},\mathcal{A},\widehat{\mathsf{KEMSR}}} = 1] \leq \Pr[\mathsf{WD}^{\widetilde{\mathsf{KEMSR}}}] < 1/2^{n_{\mathsf{WD}}}.$$

This contradicts that $\mathcal{A}$ wins with probability at least $1/2^{n_{\mathsf{WD}}}$, which is why we can conclude that $\mathsf{eq} \geq 1/2$ holds.

To analyze the security of our scheme with respect to this watchdog, consider the following sequence of games.

Game 0**.** This Game is the $\mathsf{ExpSR}^{\mathsf{OW}}_{\mathsf{WD},\mathcal{A},\widehat{\mathsf{KEMSR}}}$ experiment.

Game 1**.** This game is identical to Game 0, except all invocations of $\tilde{\mathsf{RG}}$ are replaced with uniformly random bits. Since WD runs the watchdog for $\widehat{\mathsf{RGSR}}$ as a subroutine, it outputs true only if the watchdog for $\widehat{\mathsf{RGSR}}$ does. By the $(\mathcal{O}(1), t_{\mathcal{B}}, 0)$-subversion-resilience of RG this game is therefore perfectly indistinguishable from Game 0.

Game 2**.** This game is identical to Game 1, except that the execution of game $\mathsf{Exp}^{\mathcal{A}}_{\mathsf{KEM}}$ is changed in the following way. After computing

$$(\mathsf{sk}, \mathsf{pk}) = ((\mathsf{pk}_i)_{i\in[n]}, (\mathsf{sk}_i)_{i\in[n]})$$

for $(\mathsf{sk}_i, \mathsf{pk}_i) = \widetilde{\mathsf{Gen}}(; r_i)$, and then

$$(C^*, K_1) = ((C_1^*, \ldots, C_n^*), (K_{11} \oplus \ldots \oplus K_{1n}))$$

with $(C_i^*, K_{1i}) = \widetilde{\mathsf{Encaps}}(\mathsf{pk}_i; s_i)$ and uniform r_i, s_i, the experiment checks whether an index $i \in [n]$ exists such that

$$F_{\mathsf{KEM}}(r_i, s_i) = (\mathsf{pk}_i, \mathsf{sk}_i, C_i, K_i) = F_{\widetilde{\mathsf{KEM}}}(r_i, s_i).$$

Thus, the experiment ensures that at least one ciphertext was computed according to the specification, with a public key that was also computed according to the specification. If such a ciphertext was not output, then the game simply aborts.

Note that Game 2 and Game 1 only differ if an abort occurs *after* the watchdog has approved the implementation. Therefore, the probability of this event is

$$\begin{aligned}
\Pr[\text{Abort}] &= \Pr\left[\begin{array}{c} \mathsf{WD}^{\widehat{\widetilde{\mathsf{KEM}}}} = \text{ false} \wedge \\ \text{Challenger aborts } \mathsf{Exp}^{\mathsf{OW}}_{\mathcal{A},\mathsf{KEM}} \end{array}\right] \\
&\overset{(*)}{=} \Pr\left[\mathsf{WD}^{\widehat{\widetilde{\mathsf{KEM}}}} = \text{ false}\right] \cdot \Pr\left[\text{ Challenger aborts } \mathsf{Exp}^{\mathsf{OW}}_{\mathcal{A},\mathsf{KEM}}\right] \\
&\leq \left(\frac{|\mathsf{Eq}|}{|R_{\mathsf{gen}} \times R_{\mathsf{enc}}|}\right)^{n_{\mathsf{WD}}} \cdot \left(\frac{|\mathsf{Neq}|}{|R_{\mathsf{gen}} \times R_{\mathsf{enc}}|}\right)^{n} \\
&\overset{(**)}{\leq} \left(\frac{n_{\mathsf{WD}}}{n_{\mathsf{WD}} + n}\right)^{n_{\mathsf{WD}}} \cdot \left(1 - \frac{n_{\mathsf{WD}}}{n_{\mathsf{WD}} + n}\right)^{n}
\end{aligned}$$

with Neq and Eq as defined in Equation (6.6) and Equation (6.7), respectively. Note that the equation marked with $(*)$ holds because the two events are independent since they only depend on the used randomness, and the watchdog samples its randomness independently from the experiment. The following inequality holds by the watchdog's definition and the abort condition. The bound marked with $(**)$ holds since the previous line can be written as $p^{\lambda} \cdot (1-p)^{\lambda}$ for

some $p \in [0,1]$. Calculating the derivative of this function and computing the root yields that the term is maximized for

$$p = \left(\frac{|\mathsf{Eq}|}{|R_{\mathsf{gen}} \times R_{\mathsf{enc}}|} \right) = \frac{n_{\mathsf{WD}}}{n_{\mathsf{WD}} + n}.$$

Thus, if we fix n_{WD} and n, the above term states the best bound any adversary can achieve.

Now, we are ready to argue that security in Game 2 is implied by the security of the underlying KEM. To this end, consider a $(t_{\mathcal{A}}, \varepsilon_2)$ adversary $\mathcal{A}_2$ which breaks the security of Game 2. From this, we construct a $(t_{\mathcal{B}}, \varepsilon)$ adversary $\mathcal{B}$ breaking the security of the underlying KEM.

Construction of $\mathcal{B}$. Adversary $\mathcal{B}$ receives as input a challenge $\mathsf{ch} = (\mathsf{pk}^*, C^*)$ and then simulates Game 2 as follows.

First, $\mathcal{B}$ obtains an implementation $\widetilde{\mathsf{KEMSR}}$ and state st from $\mathcal{A}_0$. It runs the watchdog for $\widehat{\mathsf{KEMSR}}$ as specified above. In case the watchdog outputs false, $\mathcal{B}$ outputs 0, just like the original security experiment. Otherwise, $\mathcal{B}$ continues to simulate Game 2.

If this is the case, then the adversary $\mathcal{B}$ generates n keys $(\mathsf{sk}, \mathsf{pk}) = ((\mathsf{sk}_1, \ldots, sk_n), (\mathsf{pk}_1, \ldots, \mathsf{pk}_n))$ using the amalgamated algorithm $\widetilde{\mathsf{Gen}}$, based on the implementation provided by $\mathcal{A}_0$. To compute the challenge ciphertexts, $\mathcal{B}$ computes ciphertexts C_i and keys K_i for $i \in [n]$ by running $\widetilde{\mathsf{Encaps}}$ using uniformly random coins. As in Game 2, the adversary $\mathcal{B}$ now checks whether there exists $(\mathsf{sk}_i, \mathsf{pk}_i, C_i, K_i)$ for some $i \in [n]$ which were computed according to the specification. In case no such pair is found, $\mathcal{B}$ aborts.

Otherwise, let i denote the smallest index for which this condition is fulfilled. Then, $\mathcal{B}$ computes the challenge ciphertext for $\mathcal{A}$ by replacing (pk_i, C_i) by its own challenge $\mathsf{ch} = (\mathsf{pk}^*, C^*)$. More formally, $\mathcal{B}$ outputs $(\mathsf{st}, \mathsf{pk}, C)$ with $\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ and $C = (C_1, \ldots, C_n)$ to $\mathcal{A}_1$, where $(\mathsf{pk}_i, C_i) = (\mathsf{pk}^*, C^*)$. Finally, let $\mathcal{A}$ output a key K. $\mathcal{B}$ then computes $K^* = K \oplus_{i=1}^{n} K_i$ where K_i are the keys simulated by $\mathcal{B}$.

Now observe that if $\mathcal{A}$ indeed computes the correct key, then so does $\mathcal{B}$, since $\mathcal{B}$ simulated $n-1$ keys and can therefore compute the correct solution for its own challenge.

It remains to analyze the advantage of $\mathcal{B}$. If the embedded challenge could also have been output by $\mathcal{A}$'s implementation, then $\mathcal{B}$ simulates Game 2 correctly. We can lower bound the probability of this event by $1/2$ by utilizing our observation earlier that $\mathsf{eq} \geq 1/2$. Thus, we have

$$\begin{aligned}
\Pr[\mathsf{Exp}^{\mathcal{B}}_{\mathsf{KEM}}] &= \Pr[\mathsf{ch} \in \mathsf{Eq}] \cdot \Pr[\mathcal{A} \text{ wins } G_2 | \mathsf{ch} \in \mathsf{Eq}] \\
&\quad + \Pr[\mathsf{ch} \notin \mathsf{Eq}] \cdot \Pr[\mathcal{A} \text{ wins } G_2 | \mathsf{ch} \notin \mathsf{Eq}] \\
&\geq 1/2 \cdot \Pr[\mathcal{A} \text{ wins } G_2 | \mathsf{ch} \in \mathsf{Eq}] \\
&= 1/2 \cdot \Pr[\mathcal{A} \text{ wins } G_2].
\end{aligned}$$

Security parameter λ	n	$\lceil \log_2(n_{\mathsf{WD}}) \rceil$
128	32	8
128	16	11
128	8	18
128	4	33
256	64	9
256	32	12
256	16	19
256	8	34

Table 6.1: Instantiating our construction and the watchdog with different values n and n_{WD}. Recall that n is the number of parallel KEM instances used in our construction, and n_{WD} is the number of tests (on each Gen and Encaps) done by the watchdog.

Putting the above together, we obtain $2\varepsilon \geq \varepsilon_2$. Since Game 2 and Game 1 only differ by the abort condition, we obtain that

$$\varepsilon_2 = \varepsilon_1 - \Pr[\text{ Abort}] \geq \varepsilon_1 - \left(\frac{n_{\mathsf{WD}}}{n_{\mathsf{WD}} + n}\right)^{n_{\mathsf{WD}}} \cdot \left(1 - \frac{n_{\mathsf{WD}}}{n_{\mathsf{WD}} + n}\right)^{n}.$$

Finally, Game 1 and Game 0 are perfectly indistinguishable due to the $(\mathcal{O}(1), t_{\mathcal{B}}, 0)$-subversion-resilience of $\widehat{\mathsf{RG}}$. Since Game 0 is the original subversion-resilience Game $\mathsf{ExpSR}^{\mathsf{OW}}_{\mathsf{WD},\mathcal{A},\widehat{\mathsf{KEMSR}}}$, we obtain that

$$\varepsilon_0 \leq 2\varepsilon + \left(\frac{n_{\mathsf{WD}}}{n_{\mathsf{WD}} + n}\right)^{n_{\mathsf{WD}}} \cdot \left(1 - \frac{n_{\mathsf{WD}}}{n_{\mathsf{WD}} + n}\right)^{n}$$

which completes the proof.

□

6.3 Efficient Instantiations

After presenting our construction, the question arises of how to instantiate our scheme in a meaningful way. The variable n determines the efficiency of the constructed scheme in terms of the number of parallel instances of the underlying KEM (note that this has a direct impact on the size of keys and ciphertexts), while n_{WD} determines the number of tests performed by the watchdog. Both together determine the overall security guarantee inherited from the underlying KEM, as well as the efficiency of the construction.

Defining n_{WD} and n as variables yields interesting tradeoffs between the watchdog's runtime, the size of ciphertexts and keys, and the obtained security bounds. In Table 6.1 we consider different choices of n and n_{WD} for $\lambda \in \{128, 256\}$, i.e., "128-bit" and "256-bit" security. For different values of n, we compute the number n_{WD} of tests performed by the watchdog to achieve that

$$\left(\frac{n_{\mathsf{WD}}}{n_{\mathsf{WD}}+n}\right)^{n_{\mathsf{WD}}} \cdot \left(1-\frac{n_{\mathsf{WD}}}{n_{\mathsf{WD}}+n}\right)^{n} \leq 2^{-\lambda}$$

holds. Note that together with the assumption that the underlying KEM is instantiated such that it provides $\varepsilon \leq 2^{-\lambda}$, we thus obtain a security bound on the subversion-resilient KEM of

$$\varepsilon_0 \leq 2\varepsilon + \left(\frac{n_{\mathsf{WD}}}{n_{\mathsf{WD}}+n}\right)^{n_{\mathsf{WD}}} \cdot \left(1-\frac{n_{\mathsf{WD}}}{n_{\mathsf{WD}}+n}\right)^{n} \leq 3 \cdot 2^{-\lambda}.$$

Table 6.1 shows how our subversion-resilient KEM can be instantiated. For instance, for $\lambda = 128$ and with $n = 8$, the watchdog only needs to test the Gen and Encaps algorithm only $2^{18} = 262.144$ times, which can be practically accomplished for many underlying KEM constructions within a short time on moderate hardware. Even for $\lambda = 128$ and with n as small as $n = 4$ only $2^{33} \approx 8.6$ billion tests are already sufficient, which also seems practically feasible since it can be accomplished for most underlying KEMs within minutes or at most few hours on standard hardware such as a laptop computer.

6.4 From OW to IND Security

Note that one-way security is strictly weaker than indistinguishability. However, as we show in Chapter 8, it seems that IND-security in our model and with our construction does not directly follow from a combiner-like approach. That being said, there are transformations to obtain an IND-secure KEM from a one-way KEM by applying a random oracle to the computed key. As the random oracle behaves like a random function and the key fed as input can not be computed by an adversary, the resulting output is indistinguishable from random keys. In [RTYZ18] Russell et al. showed how to sanitize subverted random oracles in an asymptotic setting. Unfortunately, their approach heavily relies on their asymptotic model, and after the watchdog's testing, specification and implementation deviate only on a negligible fraction of inputs, which does not seem possible to guarantee in our model.

7 Subversion-Resilient Public-Key Encryption

After successfully developing a subversion-resilient one-way KEM, we now aim to build a subversion-resilient one-way public-key encryption scheme. Public-key encryption is a cryptographic technique that employs a pair of keys: a public key, which is widely distributed and used for encrypting data, and a private key, which is kept secret by the recipient and employed for decrypting the data. The public key can only be used to encrypt messages, which can only be decrypted by the corresponding private key, ensuring secure communication and data protection.

We show that the standard way to construct public-key encryption from a KEM also preserves subversion-resilience, provided that a trusted XOR operation is given. We begin by recalling the standard definition of public-key encryption and its standard one-way security definition.

Definition 10. A *public key encryption* scheme $\mathsf{PKE} = (\mathsf{Gen}_{\mathsf{PKE}}, \mathsf{Encrypt}, \mathsf{Decrypt})$ consists of three algorithms with the following syntax:

- $\mathsf{Gen}_{\mathsf{PKE}}()$: The randomized *key-generation algorithm* outputs a key pair $(\mathsf{sk}, \mathsf{pk})$.
- $\mathsf{Encrypt}(\mathsf{pk}, M)$: The randomized *encryption algorithm* takes as input the public key pk and a message M and outputs the ciphertext C.
- $\mathsf{Decrypt}(\mathsf{sk}, C)$: The deterministic *decryption algorithm* takes as input the secret key sk and the ciphertext C. It outputs a message M or the error symbol $\perp$.

One-way security captures the notion that an adversary should be unable to compute the message encrypted in a ciphertext. Thus, the experiment (depicted in Fig. 7.1) chooses a random message, which is then encrypted under a previously generated public key. The adversary then wins if it can return the message that was encrypted. We want to mention that there are other ways of defining one-way security for public-key encryption schemes. The winning requirement is outputting a message M, such that the challenge ciphertexts decrypt to M. In a non-subversion setting, these two ways of defining one-way security are equivalent. However, we chose the following approach to guard the decryption algorithm from subversion due to the aforementioned challenges in guarding the decryption algorithm from subversion.

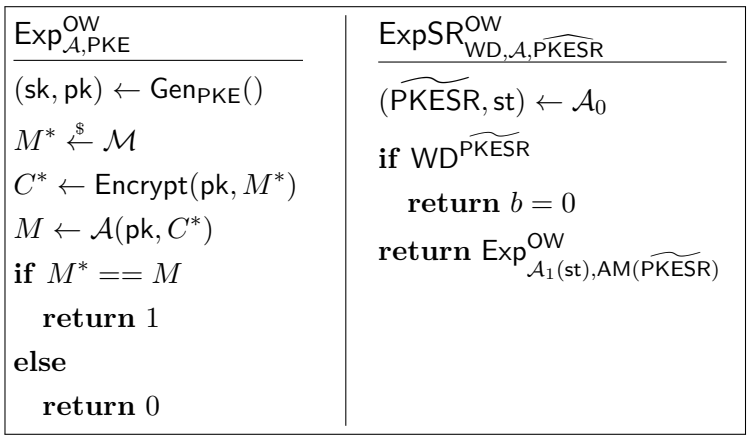


Figure 7.1: Left: One-way experiment for public-key encryption schemes with message space $\mathcal{M}$. Right: Subversion-resilience one-way experiment for public-key encryption schemes.

Definition 11. We say that $\mathsf{PKE} = (\mathsf{Gen}_{\mathsf{PKE}}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with message space $\mathcal{M}$ is $(t_{\mathcal{A}}, \varepsilon)$-*one-way* if for any adversary $\mathcal{A}$ running in time at most $t_{\mathcal{A}}$ it holds that

$$\mathsf{Adv}^{\mathsf{OW}}_{\mathcal{A},\mathsf{PKE}} := |\Pr[\mathsf{Exp}^{\mathsf{OW}}_{\mathcal{A},\mathsf{PKE}}]| \leq \varepsilon$$

with $\mathsf{Exp}^{\mathsf{OW}}_{\mathcal{A},\mathsf{PKE}}\mathcal{A}$ shown in Figure 7.1.

Definition 12. We say that a specification of a public-key encryption scheme $\widehat{\mathsf{PKESR}} = (\mathsf{AM}_{\mathsf{PKE}}, \mathsf{PKESR})$ is $(t_{\mathsf{WD}}, t_{\mathcal{A}}, \varepsilon)$-*subversion-resilient one-way in the offline watchdog model with trusted amalgamation* if one can efficiently construct a *correct* watchdog WD running in time at most t_{WD} such that for any adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ running in time $t_{\mathcal{A}}$ it holds that

$$\mathsf{AdvSR}^{\mathsf{OW}}_{\mathsf{WD},\mathcal{A},\widehat{\mathsf{PKESR}}} \leq \varepsilon$$

with the used experiments shown in Fig. 7.1.

Description of the Construction. We then can construct a subversion-resilient public-key encryption scheme from a subversion-resilient key encapsulation mechanism in a straightforward manner. Keys for the PKE are generated identically to the way keys are generated in the KEM. To encrypt a message, a ciphertext/key pair is computed by the encapsulation algorithm. The amalgamation then takes both the key and the message and applies a $\oplus$ operation to them. This value is then output together with the ciphertext computed by the encapsulation algorithm. Thus, let $\widehat{\mathsf{KEMSR}} = (\mathsf{AM}_{\mathsf{KEM}}, \mathsf{KEMSR})$ be the specification of a subversion-resilient key encapsulation mechanism with

$$\mathsf{AM}_{\mathsf{KEM}}(\widehat{\mathsf{KEMSR}}) = (\mathsf{GenSR}_{\mathsf{KEM}}, \mathsf{EncapsSR}, \mathsf{DecapsSR}).$$

We then construct the specification of a public-key encryption scheme $\widehat{\mathsf{PKESR}} = (\mathsf{AM}_{\mathsf{PKESR}}, \mathsf{KEMSR})$ with

$$\mathsf{AM}_{\mathsf{PKESR}}(\mathsf{KEMSR}) = (\mathsf{GenSR}_{\mathsf{PKE}}, \mathsf{EncryptSR}, \mathsf{DecryptSR})$$

where each algorithm is defined as follows:

- $\mathsf{GenSR}_{\mathsf{PKE}}()$: Output $(\mathsf{sk}, pk) = \mathsf{GenSR}_{\mathsf{KEM}}()$.
- $\mathsf{EncryptSR}(\mathsf{pk}, M)$: Compute $(C, K) \leftarrow \mathsf{EncapsSR}(\mathsf{pk})$ and output $(C, K \oplus M)$.
- $\mathsf{DecryptSR}(\mathsf{sk}, C)$: Parse $C = (C_0, C_1)$. Compute $K \leftarrow \mathsf{DecapsSR}(\mathsf{sk}, C_0)$. Output $M = C_1 \oplus K$.

Thus, the specification of our public-key encryption scheme is basically the specification of the underlying subversion-resilient key encapsulation mechanism, and the amalgamation $\mathsf{AM}_{\mathsf{PKESR}}$ is almost identical to $\mathsf{AM}_{\mathsf{KEM}}$. The only difference is that the message is additionally computed via an XOR to the key K during encrypt.

Security Analysis. With this simple construction setup, we continue with its security analysis. Subversion-resilience of the new public-key encryption scheme follows directly from the security of the underlying KEM and the usage of a trusted XOR.

Theorem 3. *Let* $\widehat{\mathsf{KEMSR}} = (\mathsf{AM}_{\mathsf{KEM}}, \mathsf{KEMSR})$ *be the specification of a* $(t_{\mathsf{WD}}, t_{\mathcal{A}}, \epsilon)$ *subversion-resilient one-way KEM. Then* $\widehat{\mathsf{PKESR}}$ *as described above is* $(t_{\mathsf{WD}}, t_{\mathcal{A}}, \epsilon)$ *subversion-resilient one-way in the offline watchdog with trusted amalgamation assuming a trusted XOR operation.*

Proof. The watchdog for PKE simply runs the watchdog for $\widehat{\mathsf{KEMSR}}$ as a subroutine. Any adversary $\mathcal{A}$ against the one-wayness of $\widehat{\mathsf{PKESR}}$ can then be used to construct an adversary $\mathcal{B}$ that breaks the one-wayness of $\widehat{\mathsf{KEMSR}}$. Thus, let $\mathcal{B}$ receive the challenge (pk, C) and then be asked to return the encapsulated key K in C. $\mathcal{B}$ then forwards $(\mathsf{pk}, (C, R))$ to $\mathcal{A}$, where R is chosen uniformly at random from the message space. Thus, instead of actually computing $K \oplus M$ for some random M, $\mathcal{B}$ chooses a uniformly random element since K is not known. Due to the XOR, for every R chosen by $\mathcal{B}$, there exists some M' such that $R = K \oplus M'$ where K is the key encapsulated in $\mathcal{B}$'s challenge. Thus, if $\mathcal{B}$ indeed breaks the one-wayness of $\widehat{\mathsf{PKESR}}$, it will return M'. This then allows $\mathcal{B}$ to compute $R \oplus M' = K \oplus M' \oplus M' = K$, which breaks the one-wayness of $\widehat{\mathsf{KEMSR}}$. As the success probabilities of $\mathcal{A}$ and $\mathcal{B}$ coincide, it follows that PKE is subversion-resilient iff $\widehat{\mathsf{KEMSR}}$ is subversion-resilient. □

Given a subversion-resilient KEM, it was not hard to construct a subversion-resilient PKE scheme with a very simple trusted amalgamation function in the watchdog model. Note that our construction avoids input-trigger attacks, as

during encryption the message is directly fed into the trusted XOR operation. A similar technique was used by Russell et al. [RTYZ17] also used a trusted XOR operation to defend against input trigger attacks but in a different way from ours. In their approach, they sample a uniform mask to XOR the message before being encrypted. The mask sampling is also sanitized via their proposed double-split construction. Since our construction of subversion-resilient KEM needs a trusted XOR operation already, the construction of PKE does not require any additional assumption if compared with the proposed KEM. Furthermore, our construction is simpler and only requires a much less runtime-bounded watchdog.

One may wonder why we do not directly build a subversion-resilient public-key encryption scheme since the experiment chooses M uniformly at random and thus does not allow the adversary to choose input triggers freely. However, we then again run into problems similar to the ones presented in Chapter 8. There will always be non-negligible many input triggers in the implementation, as long as the watchdog engages in polynomial many testing queries. Thus, we utilize a trusted XOR and base the security of our scheme on a subversion-resilient KEM.

8 On the Impossibility of Subversion-Resilient Indistinguishability

The original security proof in [BCJ21] aimed to show subversion-resilience with regard to the security notion of *indistinguishability*, i.e. Definition 7. Unfortunately, as we show in the following, a very subtle detail invalidates the proposed security proof. On a very high level, the proof strategy was the following:

- The testing of the watchdog guarantees with overwhelming probability that at least one pair (pk_i, C_i) generated in the subversion security experiment does not deviate from the specification.
- We can replace this pair by the challenge pair (pk^*, C^*) obtained in the KEM security experiment.
- To be successful, the subversion adversary thus needs also to solve this challenge (pk^*, C^*) due to the combiner property.

Thus, this strategy mostly followed directly the proof strategy of the previous section. Unfortunately, the above strategy does not hold up against scrutiny, as (pk^*, C^*) might never be the output of the subverted implementation. A simple example of this would be an implementation, which would never output a public key ending in "0". The adversary could thus identify such instances and fail *intentionally*, effectively canceling out its advantage. Based on this, we now show that every reduction that performs such a simple embedding such that the adversary has "direct access" to it can not be successful.

Theorem 4. *Let R be a reduction that reduces the security of a subversion-resilient KEM* KEMSR *to the security of an indistinguishable KEM* KEM*. Let $\mathsf{pk} = (\mathsf{pk}_0, \ldots, \mathsf{pk}_n)$, $C = (C_0, \ldots, C_n)$ be the public key and ciphertext of* KEMSR*. Furthermore, let R be any reduction, that after running the watchdog for* KEMSR*, takes a public key / ciphertext pair (pk^*, C^*) of* KEM *and replaces any pk_i, C_i where $i \in [n]$ with (pk^*, C^*) before handing this new challenge to some adversary. Then, if there exists an adversary $\mathcal{B}$ which breaks the (t, ε)-subversion-resilience of* KEMSR*, there exists an adversary $\mathcal{A}$ which breaks the (t, ε)-subversion-resilience of* KEMSR*, but cannot be used by the (single-stage) reduction R of the form defined above to break the indistinguishability of* KEM*.*

In other words, for every reduction R that follows the strategy explained above, there is always an attacker $\mathcal{A}$ that can break the subversion-resilience without breaking the indistinguishability for R.

Proof. First, the adversary prepares its implementation, such that the outputs of (Gen, Encaps) deviate from the specification for a fraction neq of all (randomness) inputs of Gen × Encaps. Similarly, we use $\mathsf{eq} = 1 - \mathsf{neq}$ to denote the fraction of inputs for which outputs follow the specification. Then, the q denote the amount of tests performed by watchdog on(Gen, Encaps).

Now, we construct the adversary $\mathcal{A}$. If the watchdog approves the implementation (i.e., it never encounters a deviation from the specification), $\mathcal{A}$ is given its challenge $\mathsf{ch} = (\mathsf{pk}^*, K^*, C^*)$. Due to the possibility of using our von Neumann construction, the keys in the challenge will always be uniform bitstrings. Thus, we can ignore K^* in the following and simply use $\mathsf{ch} = (\mathsf{pk}^*, C^*)$. We now distinguish the two following scenarios:

- If the challenge given to $\mathcal{A}$ is correctly distributed with regard to the subverted implementation, the adversary flips a biased coin with $\mathsf{bias} = \mathsf{neq}/\mathsf{eq}$. For the event with probability bias, the adversary runs $\mathcal{B}$ to break the security of the scheme with probability ε. Otherwise (which happens with probability $1 - \mathsf{bias}$), the adversary samples a uniformly distributed bit and returns this random bit.
- If the challenge given to $\mathcal{A}$ is not correctly distributed, there is a public key/ciphertext pair that the subverted implementation could not have produced. Now, $\mathcal{A}$ tries to deliberately output a wrong solution, as they know they are used in a reduction R. To do so, $\mathcal{A}$ runs $\mathcal{B}$ to obtain a bit b and then outputs $1 - b$. Hence, it outputs the correct bit with probability $1 - \varepsilon$.

Now, let us analyze $\mathrm{Adv} := \left|\Pr[\mathsf{Exp}^{\mathsf{IND}}_{\mathcal{R},\mathsf{KEM}} = 1] - 1/2\right|$, i.e., the advantage of the reduction R in breaking the indistinguishability security of the underlying KEM. By definition, this advantage equals $|\Pr[\mathrm{SR} = 1] - 1/2|$. Here, SR is the subversion security game played by $\mathcal{A}$, where the reduction embeds a challenge somewhere in a construction such that the adversary has access to this challenge. To analyze Adv, we introduce the following events:

detected: This event describes that the Watchdog outputs "true" during its testing phase, i.e., the watchdog accepts the implementation.

A: This event ("attack") refers to $\mathcal{A}$ flipping its coin and then trying to break the security of the considered scheme (which happens with probability bias).

R: This event ("random") is the complementary event to "attack" and refers to $\mathcal{A}$ simply outputting a random bit.

Now, the law of total probability gives

$$\begin{aligned}\mathrm{Adv} = |&\Pr[SR = 1|\text{detected}] \cdot \Pr[\text{detected}] \\ &+ \Pr[\mathrm{SR} = 1|\neg \text{ detected} \wedge \mathsf{ch} \notin \mathsf{Eq}] \cdot \Pr[\neg \text{ detected} \wedge \mathsf{ch} \notin \mathsf{Eq}] \\ &+ \Pr[\mathrm{SR} = 1|\neg \text{ detected} \wedge \mathsf{ch} \in \mathsf{Eq} \wedge A] \cdot \Pr[\neg \text{ detected} \wedge \mathsf{ch} \in \mathsf{Eq} \wedge A] \\ &+ \Pr[\mathrm{SR} = 1|\neg \text{ detected} \wedge \mathsf{ch} \in \mathsf{Eq} \wedge R] \cdot \Pr[\neg \text{ detected} \wedge \mathsf{ch} \in \mathsf{Eq} \wedge R] \\ &- 1/2|.\end{aligned}$$

Plugging in the probabilities that the subverted implementation is detected (or not) and the different scenarios for $\mathcal{A}$ shows that this is equal to

$$\begin{aligned}\mathrm{Adv} \overset{(1)}{=} |&1/2 \cdot (1 - \mathsf{eq}^q) \\ &+ \varepsilon \cdot (\mathsf{eq}^q \cdot \mathsf{neq}) \\ &+ (1 - \varepsilon) \cdot (\mathsf{eq}^{q+1} \cdot \mathsf{bias}) \\ &+ 1/2 \cdot (\mathsf{eq}^{q+1} \cdot (1 - \mathsf{bias})) \\ &- 1/2| \\ \overset{(2)}{=} |&1/2 \cdot (1 - \mathsf{eq}^q) \\ &+ \varepsilon \cdot (\mathsf{eq}^q \cdot \mathsf{neq}) \\ &+ (1 - \varepsilon) \cdot (\mathsf{eq}^{q+1} \cdot \mathsf{neq}/\mathsf{eq}) \\ &+ 1/2 \cdot (\mathsf{eq}^{q+1} \cdot (1 - \mathsf{neq}/\mathsf{eq})) \\ &- 1/2| \\ \overset{(3)}{=} |&1/2 \cdot (1 - \mathsf{eq}^q) \\ &+ (\mathsf{eq}^q \cdot \mathsf{neq}) \\ &+ 1/2 \cdot \mathsf{eq}^{q+1} \\ &- 1/2 \cdot \mathsf{eq}^{q+1}\mathsf{neq}/\mathsf{eq})) \\ &- 1/2| \\ =|&1/2 \cdot \left(1 - \mathsf{eq}^q + \mathsf{eq}^q \cdot \mathsf{neq} + \mathsf{eq}^{q+1}\right) - 1/2| \\ =|&1/2 \cdot (1 - \mathsf{eq}^q + \mathsf{eq}^q(\mathsf{neq} + \mathsf{eq})) - 1/2| \\ =|&1/2 \cdot (1 - \mathsf{eq}^q + \mathsf{eq}^q) - 1/2| \\ =&0\end{aligned}$$

where in equation (1), we simply replaced the probabilities of the events with the corresponding values of eq and neq and the probabilities according to $\mathcal{A}$'s strategy. In (2) we substituted bias with neq/eq, according to its definition. All equations after (3) are simple manipulations. Hence, we conclude that R is not breaking the indistinguishability of KEM. □

Note that the bias $\mathsf{bias} = \mathsf{neq}/\mathsf{eq}$ chosen by the adversary $\mathcal{A}$ is independent of the number of tests q performed by the watchdog. Hence, this value cancels

out one additional eq term, which is *always* true, independent of the size of q. Also, the size of Neq (and therefore neq) can not be made arbitrarily small by simply more testing of the watchdog, as long as the watchdog is independent of the adversary and engages in a fixed number of tests. Consider an adversary which prepares its implementation with $\mathsf{neq} = ((q/q+1))$ were q denotes the amount of testing queries by the watchdog, The probability that the adversary avoids detection is then $(1-(q/q+1))^q$ and approaches $1/e$ for $q \to \infty$, which is non-negligible, while the probability of being handed a subverted challenge is neq, which is also non-negligible. Thus, as long as the adversary is allowed to make its implementation depend on the watchdog, the above impossibility can not be circumvented with additional polynomial-time testing.

Note that Theorem 4 is phrased explicitly to capture the results proposed in [BCJ21]. It seems that even broader statements about impossibility are possible in this setting of subversion-resilience: Instead of directly embedding a challenge, a reduction could also try to rerandomize it before embedding it if KEM is sufficiently rerandomizeable. However, this approach also does not seem to work, as the reduction does not know whether the rerandomized challenge can be computed by the implementation, as the reduction can not recompute the randomness used for the challenge. Otherwise, it would directly break the security of KEM. On the other hand, a general impossibility result regarding indistinguishability with limited testing is also not possible. If a construction shifts all computations to the trusted amalgamation, it is trivially subversion-resilient. However, we would deem such a scheme impractical, since it removes all attack capabilities of the adversary, Thus, one would require a formalization that not "too much trust" is put into the amalgamation to acquire meaningful impossibility results. However, such a formalization is far from trivial and beyond the scope of this thesis. We thus leave this formalization as an open direction for future work. Additionally, our impossibility result only captures an approach where we run the adversary a *single* time. Breaking with this assumption and utilizing multiple runs of the adversary is another interesting direction for future research to improve our results.

9 Discussion

After presenting our results, let us discuss our contributions.

Efficient Watchdogs but Increased Key and Ciphertext Size. Further reducing the watchdog's testing overhead is crucial to make our approach more suitable for efficient use on a larger scale. While previous works achieved results in an asymptotic setting, the question remains how these results would be realized in practice. Especially only requiring a non-negligible detection advantage is most likely not accurately modeling the requirements intuitively expected from subversion-resilient systems. Our contributions with regard to efficient watchdogs can thus be seen as a bridging step between these two worlds. Our public-key encryption scheme guarantees that the user running the watchdog can be sure that no adversary can utilize a backdoor. This security comes with the price of increased key and ciphertext size and only guarantees one-way security. We still believe that our scheme can be helpful in applications where security against subversion is critical while the decreased efficiency might be acceptable. This could, for example, be an investigative journalist who wishes to communicate with its sources. In this scenario, only small amounts of data must be exchanged, and the additional overhead might be acceptable. Also, if our PKE (or KEM) scheme would be used to exchange key material and, on top of that, another subversion-resilient scheme (like our AEAD scheme presented in Section 12.8), the increased key and ciphertext size could also be justified. See Chapter 14 for a deeper discussion on this possible direction for future research.

(Im)possibility of Black-Box Constructions. Both our constructions of subversion-resilient KEM and PKE schemes rely on a trusted XOR operation. We conjecture that such a trusted operation is necessary to achieve subversion-resilience with a watchdog of "low" runtime. This holds even if the randomness generation of the algorithm is split from the deterministic part of the encryption algorithm. We illustrate the observation underlying this conjecture with the example of public-key encryption. To achieve indistinguishability against an adversary with polynomial running time, an exponentially big key and randomness space are required (as otherwise, an adversary could obviously simply compute the secret key or determine the randomness used in the challenge ciphertext by exhaustive search). Thus, a watchdog with a non-negligible success probability in detecting a subversion would have to test "many" inputs to rule out input-trigger style attacks, such as those described by Degrabiele et al. [DFP15]. Therefore, constructing a watchdog with high detection probability immediately requires exponential running time. Thus, we conjecture that some

sort of trusted operation is necessary to achieve subversion-resilience.

Repeated Oracle Access. In the security models used in this chapter, the adversary is given only a single challenge and is then asked to provide some response. In the security model for public key encryption by Russell et al. [RTYZ17], the adversary was also given access to an encryption oracle. Since they utilize asymptotic security definitions and prove that outputs of their subverted encryption algorithm are indistinguishable from non-subverted runs of the encryption algorithm, this additional oracle access is of no use to the adversary. For our approach, it is not clear how access to an encryption oracle could be given to the adversary while providing a sound simulation during the reduction. This is because of the following. Our proof idea requires that one of many ciphertext / public key pairs is computed according to the specification. This allows us to embed a challenge for that pair. However, if repeated oracle access is granted to the adversary, it cannot be guaranteed that this honest pair always occurs on the same index. For example, in the first query, the first output is in accordance with the specification, while in the second query, the second pair is. Thus, it is not clear how a reduction would be able to simulate this correctly. However, this reflects the different approaches of the results of this thesis and the results of Russell et al. [RTYZ17]. While Russell et al. aimed for indistinguishability of the implementation and the specification and then used this to argue for the subversion-resilience of their construction, our goal is only to guarantee some security guarantee. For our construction, the adversary is able to distinguish its implementation from the specification, but this still does not give it the power to break the one-way security of our scheme. Additionally, not providing an encryption oracle gives very little additional power to the adversary. This is because the adversary is aware of both the specification and the implementation and could thus compute all outputs of an encryption oracle by itself.

Limited Impossibility. Our impossibility result in Chapter 8 is phrased to specifically capture the results presented in [BCJ21]. A natural question is whether our limited impossibility results can be proven in a more general form or whether other proof techniques allow our (or a similar) construction to be proven subversion-resilient indistinguishable. As we only show the impossibility of reductions that run the adversary only once, a natural option could be to run the adversary *multiple times*. This, together with some bound on the probability that the adversary is simulated correctly, and for example, a majority vote over all runs of the adversary, could potentially lead to positive results. However, a careful investigation is necessary to bring substance to this approach.

Part II

Security under Complete Subversion Without Random Oracles

In the following chapters, we present the first construction of subversion-resilient authenticated encryption where encryption and decryption are subject to subversion, and the first construction of digital signatures where all algorithms are subject to subversion, in an asymptotic offline watchdog model while abstaining from using random oracles.

Author's Contributions. The author of this thesis observed that the Encrypt-then-MAC approach can be leveraged for subversion-resilience. The main building block of subversion-resilient PRFs was constructed as joint work with Sebastian Berndt, Denis Diemert, Tibor Jager, and Thomas Eisenbarth through several iterations while the author of this thesis focused on the security of PRFs and MACs. These results were published in [BBD+23].

With subversion-resilient PRFs being available, the author of this thesis suggested revisiting classical results in cryptography and using this primitive to obtain subversion-resilient signatures. The author's main contributions are the construction of subversion-resilient one-way functions and the idea of using classical approaches to obtain digital signatures from symmetric primitives, while Sebastian Berndt focused on the details of the Naor-Yung construction. The construction details, such as target-collision-resistant hash functions, were developed in joint work with Sebastian Berndt and Rongmao Chen. The results on subversion-resilient signatures were published in [BBC24].

10 Complete Subversion

In Part I, we saw that achieving one-way security for KEMs and PKE with practical watchdogs in a concrete security setting is indeed possible. However, constructing indistinguishable KEMs in our model remains an interesting open problem. While our proposed model and constructions allow for more practical watchdogs, this comes with the downside of increased key and ciphertext size. Although utilizing an efficient watchdog in a concrete security setting is desirable, improving our results seems hard. As indicated by our results from the previous chapter, this task could even be impossible.

As we see shortly, in an asymptotic security setting using watchdogs described with asymptotic running times, many issues that arise in a concrete security setting with practical watchdogs can be circumvented. Therefore, this part of this thesis focuses on enhancing both security and practicality in an asymptotic setting. Although the runtime of the watchdog is crucial for the practicality of a subversion-resistant scheme, other factors must also be considered. What other properties would we intuitively expect from a *practical* subversion-resilient scheme? Since the adversary provides the implementation of a scheme, we would assume that subversion resilience guarantees security even if *all* algorithms of the cryptographic scheme are provided by the adversary. However, in prior works, subversion-resilience was often only proven with regard to *partial* subversion. This means that the adversary only provides implementations for selected (often only a single) algorithms from a cryptographic scheme. Depending on the application, it can be difficult to justify that a powerful adversary is capable of modifying the implementation of widely used cryptographic schemes to exclude critical algorithms from subversion attacks. It can be difficult to justify excluding algorithms critical for security from subversion attacks if a powerful adversary changes the implementation of widely used cryptographic schemes.

Thus, while allowing results that can be seen as important stepping stones, partial subversion does not accurately model the threats we face in practice. A more practical approach models adversaries with regards to *complete subversion*, i.e., consider adversaries that provide implementations for *all* algorithms of a cryptographic primitive. However, in most prior works, either

- adversaries are not allowed to provide implementations of the decryption or verification algorithms or
- primitives use idealized primitives such as random oracles.

Generally, security experiments for weaker security properties like one-way security often grant the adversary oracle access to fewer algorithms than stronger

security notions. Note, however, that in some cases excluding some algorithms of a cryptographic scheme does not make a difference with regard to security properties, as for some security notions, not all algorithms are relevant for security. Consider our model and result on subversion-resilient KEMs from Part I as an illustrative example. There, the adversary provides an implementation for the key generation and encapsulation algorithm. Technically, the adversary could also provide an implementation of the decapsulation algorithm. However, neither the one-way nor the indistinguishability security definitions ever use this algorithm. Thus, even if the adversary provides an implementation of the decapsulation algorithm, it does not make a difference in *security*[1] in our case. For stronger[2] security properties like the security of authenticated encryption, the decryption algorithm is an integral part of the security experiment. Excluding this algorithm from subversion in this setting is thus far more restricting than excluding the decapsulation algorithm in a one-way secure KEM.

Sometimes, algorithms are excluded to enable positive results in the first place in the presence of generic attacks. For example, previous works only consider subversion of parameter generation [BFS16], key generation [Bag20], or key generation and signing [CRT+19] while the remaining algorithms are not subverted. This enabled researchers to develop techniques to achieve subversion-resilience at all, potentially with additional assumptions, like for example the trusted amalgamation model or a trusted XOR operation.

Attacks. Only considering the subversion of selected algorithms usually meant excluding algorithms run by the receiving party, i.e. the decryption of an encryption scheme or verification algorithm of a signature scheme. In general, it is unknown what strategy an adversary uses when making querying oracles corresponding to these algorithms. Thus, the watchdog cannot effectively test these algorithms according to the same distribution that the adversary accesses the oracles. One of the main problems is that this implies input-trigger style attacks [DFP15], where the algorithm deviates from the specification for adversarially chosen inputs. In the case of decryption or verification, simple attacks allow to break the security of the considered scheme. For this, the adversary prepares its implementation such that for some x (either a ciphertext or signature), the decryption oracle outputs the symmetric key, or the verification algorithm outputs the signing key (or accepts the signature). With an offline watchdog, it is impossible to detect this attack as long as the ciphertext or signature space is "sufficiently large". This is because the watchdog does not know the distribution, according to which the adversary will query the decryption or verification oracle. Thus, the probability that a watchdog tests x by choosing random in-

[1] This does however influence the *corretness* of the scheme, as this property can not be guaranteed anymore, at least not for all inputs. While correctness can be important for *some* constructions, including our construction of authenticated encryption, *in general*, correctness is *not necessary* to achieve security, and we thus do not consider it a security property.

[2] If compared to indistinguishability notions of encryption

puts is negligible. As we discussed in Section 1.2, there is a series of works by Armour, Farshim, and Poettering [AP22, DFP15, AP19] that focuses on attacking the algorithms used by message receivers, i.e. verification or decryption algorithms. They propose novel attacks that can break the security of various schemes, such as encryption and signature schemes while remaining undetected. Since their attacks are generic, it is impossible to defend against these attacks without further assumptions. Their work highlights the need for constructions to defend against attacks under complete subversion, i.e., where *all* algorithms that are relevant in practice, are subject to subversion, while minimizing the additional assumptions.

Random Oracles. In previous works, one way to achieve subversion resilience under complete subversion and to circumvent the attacks mentioned above was using a (subversion-resilient) random oracle [RTYZ16, RTYZ17, CRT+19]. In a series of works, Russell et al. [RTYZ16, RTYZ17] modeled building blocks as random oracles obtaining constructions for subversion-resilient randomness generators, encryption schemes, one-way permutations and signatures under complete subversion. However, these works only use random oracles in specific circumstances (i.e., known input distributions) and do not provide constructions for subversion-resilient random oracles. Russell et al. [RTYZ18] then showed that subversion-resilient random oracles can be constructed in an asymptotic setting under the assumption that a trusted XOR operation is available. Their construction uses many different random oracles. For each instance of a random oracle, a random blinding value is chosen, after which a trusted XOR is applied to this blinding value and the input. The key insight is that these random values are chosen *after* the adversary provided its implementation. A trusted XOR is then applied to all these instances with different blinding values and once again feed into a random oracle to obtain the final output. Russell et al. provide a sophisticated analysis of their scheme, overcoming the main challenge that the adversary can adaptively query the random oracle, making sound simulation a tough challenge.

Subversion-resilient oracles were then utilized by Chow et al. [CRT+19] to develop subversion-resilient signature schemes under complete subversion using subversion-resilient random oracles as an essential building block. This is because random oracles can be used to *randomly* map potentially input triggers on a *large* domain. This allows the watchdog again to test algorithms according to a known input distribution, i.e. sample *random* inputs. This way, it is unlikely that an input trigger that is fed into the random oracle, again "hits" a trigger, where the implementation deviates from the specification.

While this approach yielded positive results, the question remained whether random oracles are necessary to achieve subversion-resilience under complete subversion. Ateniese, Francati, Magri, and Venturi [AFMV19] proposed a generic approach to guard cryptographic schemes against complete subversion, *without* using random oracles. While granting new insights and developing interesting

techniques to avoid random oracles, their work has a few limitations. First, their work only applies to deterministic primitives. Further, their approach relies on an additional independent source of public randomness, while some of their results (e.g., with regard to signatures) also require online watchdogs. Thus, they avoided random oracles but could not guard randomized primitives and required a public randomness source, which might not be available. Therefore, our primary research question for the second part of this thesis is:

Is it possible to construct subversion-resilient primitives without random oracles in an offline watchdog model under complete subversion?

We can answer this question positively, even for schemes that are not deterministic. We show that subversion-resilietn authenticated encryption can be constructed while encryption and decryption are subject to subversion. For digital signatures we show how to achieve subversion-resilience, even if *all* algorithms are subject to subversion in the trusted amalgamation model and a few simple, trusted operations are available. As we explain in Section 12.1 in more detail, our results for subversion-resilient authenticated encryption exclude key generation from subversion to simplify our contributions while emphasizing that constructions from this and prior work be used to sanitize key generation for our construction. While this may sound to violate the notion of *complete* subversion, we argue that this still captures complete subversion for the case of symmetric cryptography in meaningful way.

Both our constructions share a similar underlying idea in their approach. We revisit classical results from cryptography and build both primitives using symmetric primitives. We observe that certain building blocks of these classical results naturally obtain subversion-resilience. Given some additional operations, i.e., a trusted XOR, a trusted comparison, a "trusted data structure", and more fine-grained access to the building blocks, we can prove our constructions subversion-resilient. Both approaches share the idea of building the corresponding primitive starting from *symmetric* primitives, which enables the recomputation of symmetric primitives during decryption/verification, circumventing input trigger attacks. Thus, we utilize trusted assumption to avoid feeding adversarially chosen inputs directly into subverted components, as mentioned by Russell et al. [RTYZ16, RTYZ17]. Our results prove that various building blocks are subversion-resilient as bridging steps.

11 Asymptotic Subversion Model

As in Part I, we begin by providing the security model for our constructions. We define the notion of subversion-resilience in an asymptotic security setting and loosely follow the approach by Russell, Tang, Yung, and Zhou [RTYZ17, RTYZ16].

We again use the trusted amalgamation model to enable our results. We provide generic security definitions in an asymptotic setting and briefly recall the trusted amalgamation model for completeness and refer to Section 4.3 for a more detailed discussion.

Security Experiments. A *security experiment* $\mathsf{Exp}^{\mathrm{GOAL}}_{\mathcal{A},\Pi}$ 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, a *subversion experiment* $\mathsf{ExpSR}^{\mathrm{GOAL}}_{\mathsf{WD},\mathcal{A},\widehat{\Pi}}$ 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 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, 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 [BL17]. To still give security guarantees against subverted implementations, different non-black-box models were presented in the literature. In this part of this thesis, we follow Russell, Tang, Yung, and Zhou [RTYZ17] 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 $\mathsf{Am}(\widetilde{\Pi})$, 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 simple operations like an XOR. To formalize this scenario, we represent the specification $\widehat{\Pi}$ of a primitive as $\widehat{\Pi} = (\mathsf{Am}, \Pi)$. 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 with the understanding that Am uses Am_Ψ as a subroutine. In case the amalgamation is very simple, we take the liberty to describe it only informally via text to avoid notional overhead. This way we can avoid a big notional overhead and put more focus on our used techniques. In case the amalgamation function is the identity function, we refer to a specification as the *trivial specification* and do not explicitly state Am. Sometimes, we consider the amalgamation function for a *single* algorithm Π_i of a primitive, denoted by Am_i. Unlike to Part I.

Split-program model. In addition to trusted amalgamation, Russell, Tang, Yung, and Zhou [RTYZ16] used the *split-program* methodology. Like modern programming 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 and will simply see it as part of the trusted amalgamation model.

Deterministic and Stateless Subversion. Due to the inherent limitations of a universal offline watchdog model, we will also assume in this part of this thesis that the subverted implementation of a deterministic building block is also deterministic and assume that the implementation is not allowed to hold any state between executions.

Asymptotic Subversion-Resilience. Next, we define the notion of subversion-resilience in an asymptotic setting based on the definitions by Russell et al. [RTYZ16, RTYZ17]. 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. 11.1. 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. 11.1, l. 1). The watchdog WD then tests the implementation before we run the security experiment Exp with adversary $\mathcal{A}_1$ on the amalgated, subverted implementation $\mathsf{Am}(\widetilde{\Pi})$ as usual (Fig. 11.1, l. 3). The variable st is only used to synchronize $\mathcal{A}_0$ and $\mathcal{A}_1$. Throughout this thesis, we use the convention that the watchdog outputs "true" if subversion is detected. To formalize subversion-resilience, consider the next definition and the corresponding security experiment shown in Fig. 11.1.

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

$(\widetilde{\Pi}, \mathsf{st}) \leftarrow \mathcal{A}_0(1^\lambda)$

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

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

Figure 11.1: The security experiment for subversion-resilient GOAL with specification $\widehat{\Pi} = (\mathsf{Am}, \Pi)$ in an offline watchdog model with trusted amalgamation.

Definition 13. A specification of a primitive $\widehat{\Pi} = (\mathsf{Am}, \Pi)$ is *subversion-resilient* GOAL *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

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

where

$$\mathsf{AdvSR}^{\mathrm{GOAL}}_{\mathcal{A},\widehat{\Pi}}(1^\lambda, \delta) = |\Pr[\mathsf{ExpSR}^{\mathrm{GOAL}}_{\mathsf{WD},\mathcal{A},\widehat{\Pi}}(1^\lambda) = 1] - \delta|$$

and

$$\mathsf{Det}^{\mathsf{WD}}_{\mathcal{A},\widehat{\Pi}}(1^\lambda) = |\Pr[\mathsf{WD}^{\widetilde{\Pi}}(1^\lambda)] - \mathsf{WD}^{\Pi}(1^\lambda)]|$$

using the experiment shown in Fig. 11.1, with $\delta \in \{0, \frac{1}{2}\}$ indicating whether a search or a decision problem is considered.

Note that $\mathsf{AdvSR}^{\mathrm{GOAL}}_{\mathcal{A},\widehat{\Pi}}(1^\lambda, \delta)$ is not parameterized by the watchdog WD. We chose this approach to simplify notation, as the watchdog testing does not directly influence the adversary's advantage. We also drop δ to help readability further, as δ is apparent from the context.

All watchdogs we propose in this part of this thesis will also be *correct*, i.e. $\Pr[\mathsf{WD}^{\Pi}(1^\lambda) = 1]$, meaning the watchdog will never reject the specification. This is because all watchdogs presented in this thesis only compare the outputs of the specification and the implementation on common inputs. While this allows a simplification for the detection advantage, we stick to the above definition to more closely follow prior work.

For public-key encryption, the above model is not equivalent to the model proposed by Russell, Tang, Yung, and Zhou [RTYZ17], where the adversary has access to a subverted encryption oracle, unlike our model, the adversary cannot access such an oracle. For symmetric encryption, our more general definition captures theirs with some differences in syntax.

Further, note that GOAL-subversion-resilience implies GOAL security as the above definition also has to hold for an adversary that outputs the specification as its implementation. To shorten the notation, in the following, we call primitives just subversion-resilient GOAL with the understanding that they fulfill

Definition 13. Finally, all upcoming construction use *correct* watchdogs, i.e. fulfill Definition 2.

Achieving Subversion-Resilience. To prove our upcoming constructions of PRFs and one-way functions subversion-resilient, we rely on an observation made by Russell, Tang, Yung, and Zhou [RTYZ16] that is particularly useful for algorithms with a *public* input distribution. This means that all parties (including the watchdog) can sample inputs according to this input distribution. For instance, an algorithm where all inputs are uniformly random bitstrings has a public input distribution. On the other hand, an algorithm that obtains inputs from an adversary during a security game where the adversary can freely choose the input is not considered public. In this case, the input distribution is not known in advance for arbitrary adversaries.

Russell et al. then observe the following: 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, the number of inputs the implementation deviates from the specification must be negligible to stay undetected.

Lemma 1 ([RTYZ16])**.** *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* $\widehat{\Pi}_1, \ldots, \widehat{\Pi}_k$ *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* $\Pr[\widetilde{\Pi}_j(x) \neq \widehat{\Pi}_k(x) : x \xleftarrow{\$} X_\lambda^j] = \delta$*, this can be detected by a ppt offline watchdog with probability at least* δ*.*

An instructive example to understand the usefulness of this lemma is the following. Suppose 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 $(x, f(x))$ are 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 $X^\star$ with regards to 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. This lemma is used to argue for the subversion-resilience of our most fundamental building blocks, one-way functions, and weak PRFs.

Comparison with Concrete Model. Let us compare this asymptotic model to the concrete model proposed in Chapter 4. In the concrete model, we explicitly make the adversary lose the security game in case of detection, while the asymptotic model instead makes use of two distinct advantages. This creates a

gap between the two models. Consider an attack where the implementation of the adversary gets rejected by the watchdog with some non-negligible probability. While the asymptotic model deems this attack unsuccessful, it is allowed in the model of Chapter 4. Therefore, the asymptotic model is weaker, as it outrules attacks that are allowed in our concrete model. However, the big advantage of the asymptotic approach is that it Lemma 1 to be applied. This is not possible in the concrete model, as we do not have a baseline to outrule attacks with non-negligible detection probability. Thus, while outruling an arguably bis class of potential adversaries, this gives us immense leverage in designing subversion-resilient schemes.

Randomness Generation. All following construction will rely on "good" randomness being available. Thus, we need a subversion-resilient randomness generator (see Chapter 5). Currently, there are two options available. First, one can directly use our randomness generator from Chapter 5. While our previous result was stated in the concrete setting, it directly translates into the asymptotic setting. The other option would be to use one of the constructions proposed by Russell, Tang, Yung, and Zhou [RTYZ17]. As mentioned in Chapter 5, the authors propose two constructions of randomness generators without random oracles. One is based on simple multi-splitting, where a randomness generator is used to obtain a single random bit, while the other uses randomness extractors. Both utilize the asymptotic framework proposed in [RTYZ17]. For our upcoming constructions, any fo these constructions can be used. Hence, for simplicity, we abstract away the randomness generation and assume that our constructions generate uniformly random bits while being able to test the randomized algorithms on selected random coins. This assumption allows us to simplify notation, significantly help readability, and focus on our contributions with regard to achieving subversion-resilience under complete subversion.

12 Subversion-Resilient Authenticated Encryption

Before we present the details of our approach to achieving subversion-resilient authenticated encryption, let us briefly discuss the main challenges for this goal and the structure of this chapter. First, let us explain the intuition behind the notion of authenticated encryption. Authenticated encryption combines symmetric encryption and data authentication to provide confidentiality and data integrity. Thus, not only is the encrypted data kept secret, but it can also be verified that it has not been tampered with during transmission. Authenticated encryption has many applications. It is for example an essential building block in cryptographic protocols like TLS [Res18] or Instant Messaging protocols like WhatsApp or Signal. Examples of authenticated encryption schemes are AES-GCM (Galois Counter Mode) [SMC08] and AES-CCM (counter with cipher block chaining message authentication code) [WHF03]. A prominent example where authenticated encryption is used is TLS. There, authenticated encryption provides confidentiality of transmitted messages and guarantees that the encrypted data was not manipulated during transit. Many authenticated encryption schemes also allow the message to contain so-called associated data (AD). This data is not confidential, but tampering with it will be detected. A typical example of associated data is the header of a network packet. For a packet to be routed, it is necessary to be able to access the destination address within the header. However, no intermediate node during the transmission should be able to change the destination address.

While there has been tremendous progress in constructing subversion-resilient schemes, including subversion-resilient symmetric and public-key encryption [RTYZ17], authenticated encryption where both encryption and decryption are subject to subversion has not been achieved in an offline watchdog model. This is due to input-trigger attacks, which previously could not be avoided without using heavy machinery such as a random oracle or a model where the decryption algorithm is exempt from subversion. Thus, this leads to the following research question:

Is it possible to design a subversion-resilient authenticated encryption without random oracles in the offline watchdog model under complete subversion?

In this chapter, we answer this question affirmatively. We revisit classical results from cryptography and see that these constructions grant subversion-resilient

authenticated encryption, given a few trusted operations. As an important stepping stone, we also propose the first construction of message authentication code (MAC), where both the tag and the verification algorithm are implemented by the adversary. Before we dive into the details of our approach, let us discuss our view on authenticated encryption under complete subversion and briefly highlight the main challenges that need to be overcome.

12.1 Symmetric Cryptography under Complete Subversion

While we state our goal as constructing subversion-resilient authenticated encryption under *complete* subversion, we will exclude key generation from subversion in our model and constructions. While this might sound counterintuitive, let us explain why we chose this approach contrary to previous works [RTYZ17] where key generation is explicitly sanitized. Although a key generation algorithm is necessary for formal security models to obtain keys used in the security experiment, the key generation algorithm of symmetric primitives, including authenticated encryption, is not *always* directly executed in practice. To illustrate this, we consider two examples of how symmetric cryptography is usually used in practice.

Local Encryption. In some cases, symmetric encryption is used "locally". One example of this is a party sampling a random key (which usually is a random bitstring) and using it for encrypting local files. In this case, providing a subversion-resilient key generation algorithm can be meaningful if key generation simply means generating random bits. Currently, two options for achieving this are available. Again, these are either the randomness generator from Chapter 5 or the randomness generator from [RTYZ17]. Both can directly be applied, as they produce random bits in a subversion-resilient manner without using random oracles. It is straightforward to add these to our following contributions by adding an additional simple game hop to all our proofs, replacing the key generation algorithm with a subversion-resilient randomness generator. However, even in the case of local encryption, there are use cases where the computation of keys is not done by simply choosing random bits but rather by deriving keys from some other input. For the example of encrypted local files, this can mean to derive a symmetric key from a user password, which can include the usage of a key derivation function.

Communication via the Internet. The other scenario is when an encryption scheme is used to secure communication via the Internet. In this case, the key generation algorithm is just an abstraction of some way two communicating parties derive a common secret key. Simply running a (subversion-resilient) randomness generator is not meaningful in this context, as it is

unclear how both parties would obtain the same key. One approach would be that one party computes the symmetric key and then sends this key to its intended communication partner, potentially with our KEM construction of Chapter 6. This however potentially requires an additional secure channel for key transportation, depending on the considered setting. Another approach for this issue is the execution of a (subversion-resilient) key exchange protocol between the two parties. This could for example be realized by a key exchange protocol guarded by a reverse firewall [DMS16, MS15].

Thus, while a useful abstraction in security definitions, a key generation algorithm does not necessarily accurately model the usage of symmetric cryptography in practice *in general*. While the subversion-resilient generator from this and previous works can indeed generate random keys in a subversion-resilient manner, this does not grant significant new insights.

Thus, we choose to exclude the key generation algorithm from subversion in this chapter while still claiming to aim for *complete* subversion. This simplifying assumption enables us to focus on previously unsolved challenges. We can avoid notational overhead and focus on the protection of the vulnerable decryption algorithm. Additionally, the abstraction of the key generation also allows the inclusion of other means of deriving a symmetric key. For a deeper discussion on possible future research in this direction, consider Section 14.3.

To model our approach of complete subversion while excluding key generation accurately, the specification of our schemes in this chapter will not include the key generation algorithm and thus will not be implemented by the adversary. Nevertheless, we include the key generation algorithm in the description of our schemes for completeness to formalize the above-mentioned abstraction. Thus, a key generation algorithm always samples keys uniformly at random from an associated key sapce.

12.2 Technical Challenges

Before we describe our solution, it is instructive to understand the difficulties in constructing subversion-resilient authenticated encryption where both the encryption and decryption algorithms are provided by the adversary. The main challenge is guarding the decryption algorithm against subversion. This is (once again) due to the aforementioned input trigger attacks, as first formalized by Degabriele, Farshim, and Poettering [DFP15]: Such an attack modifies the underlying algorithm only on a single, arbitrary input x^* called the trigger. Whenever the algorithm is given x^* 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. 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 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 attacker's task is to produce some forgery. Hence, such primitives, including authenticated encryption and MACs are 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. But even stronger attacks are possible, as shown by Armour and Poettering [AP19]. They proposed generic attacks on the decryption algorithm of authenticated encryption schemes. Concretely, they propose both an inactive and an active attack, where they manipulate the decryption algorithm to accept certain bogus ciphertexts. This way, they can still guarantee perfect correctness while establishing a covert channel through decryption error events.

12.3 Our Contributions

Our overall approach is visualized in Fig. 12.1 and can be summarized as follows. As our main building block, we rely on weak pseudorandom functions. When evaluated on random inputs, these keyed functions are indistinguishable from truly random functions. One main insight is that Weak PRFs are naturally resilient to subversion in our model, assuming stateless subversion. This is because this primitive only obtains random inputs, which can be tested by an offline watchdog. Therefore, any implementation that passes the watchdog's check is indistinguishable from random. By applying the classical Naor-Reingold transformation [NR99], we can generate subversion-resistant PRFs from weak PRFs. The trusted amalgamation within our constructions can be viewed as a *trusted data structure*. Adversarially chosen inputs are only accessible by the trusted amalgamation and can be interpreted as a path over uniformly random keys. Using subversion-resilient PRFs, the classical "PRF-as-MAC" approach by Goldreich, Goldwasser, and Micali [GGM84b] guarantees subversion-resilience through canonical verification and a trusted comparison operation. We further show that the randomized counter mode can also be made subversion-resilient by utilizing subversion-resilient PRFs, assuming a trusted XOR operation, which can be generalized to stream ciphers. Finally, we prove that subversion-resilient authenticated encryption can be achieved by combining the aforementioned ingredients through the traditional "Encrypt-then-MAC" approach [BN00, BN08]. Overall, we revisit classical results from cryptography to show that well-known constructions can be proven subversion-resilient with more fine-grained access due to the trusted amalgamation model and a few trusted operations.

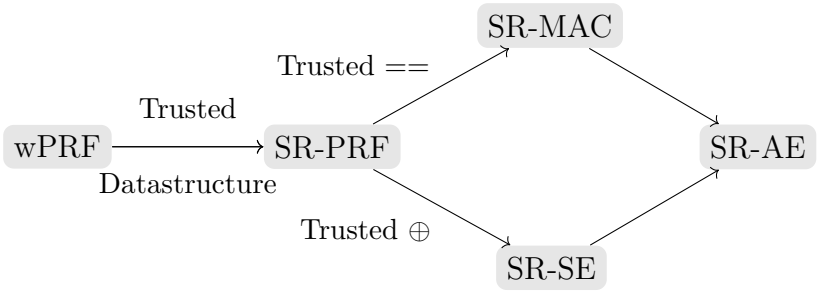


Figure 12.1: Overview of our construction. Here, SR denotes subversion-resilience, HF denotes hash function, wPRF/PRf denotes (weak) pseudorandom function, and AE denotes authenticated encryption. Comments beside arrows highlight trusted operations.

12.4 Comparison with Prior Work

This thesis presents the first constructions of subversion-resilient message authentication codes and subversion-resilient authenticated encryption under complete subversion, i.e. where both the tag and verification algorithms, or both the encryption and decryption algorithms, respectively, are implemented by the adversary. Prior work by Russell et al. [RTYZ17] achieved IND-CPA secure symmetric encryption utilizing a trusted XOR using offline watchdogs. Ateniese et al. [AFMV19] security under complete subversion, but only for deterministic primitives or using online watchdogs.

Thus, achieving authenticated encryption where both encryption and decryption are subject to subversion and MACs where verification is subject to subversion have not been achieved in an offline watchdog model.

As we use subversion-resilient PRFs as our main building block, let us discuss other works investigating this notion. Fischlin, Janson, and Mazaheri [FJM18] 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 [MT08] to a weak PRF immunizes HMAC against backdoors. As discussed later, using the RC construction also works for our construction but requires modeling 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.

Dodis, Ganesh, Golovnes, Juels and Ristenpart [DGG+15] 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 by Goldreich, Goldwasser, and Micali [GGM84a], can also be used to 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) are hidden from the subverter. These assumptions are orthogonal to each other.

Note that our approach could also be realized using a subversion-resilient random oracle instead of a PRF. Arguably, a subversion-resilient random oracle is a stronger assumption than a weak PRF (in the standard model), which we base our results on in this thesis.

Outline. We continue by constructing subversion-resilient pseudorandom functions in Section 12.5. This is our main building block for our construction of subversion-resilient message authentication codes in Section 12.6 and subversion-resilient symmetric encryption in Section 12.7. Finally, we combine all these ingredients to obtain subversion-resilient authenticated encryption in Section 12.8 and discuss our results in Section 12.9.

12.5 Pseudorandom Functions

A PRF is a mathematical function that generates data that appears random but is actually generated deterministically using a key and input. Inputting the same key and input into a PRF always produces the same output, which appears random to any party without the key. PRFs are commonly used in cryptographic constructions, such as deriving encryption keys from previously negotiated secret keys.

To make things more formal, 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 $\mathcal{D} \to \mathcal{R}$. Specifically, to account for the fact that the (size of) the domain and range of a function may differ depending on the security parameter, let $\mathcal{K} = \bigcup_{\lambda\in\mathbb{N}} \mathcal{K}_\lambda$, $\mathcal{D} = \bigcup_{\lambda\in\mathbb{N}} \mathcal{D}_\lambda$, 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 $\mathcal{R}$. This part of this thesis only considers spaces that are subsets of $\{0,1\}^*$.[1] Let us recall the standard definition of (weak) PRFs, which captures that these functions are indistinguishable from random functions if only evaluated under random inputs. In the security experiment $\mathsf{Exp}^{\mathsf{wPR}}_{\mathcal{A},F}$, the adversary is given both the function evaluation and the random input used to evaluate the function. This is important because without providing the random input, even simple functions like the identity function would meet the security definition since the random input would not be known to the adversary.

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

$\mathsf{Exp}^{\mathsf{PR}}_{\mathcal{A},F}(1^\lambda)$	$\mathsf{Exp}^{\mathsf{wPR}}_{\mathcal{A},F}(1^\lambda)$
$b \xleftarrow{\$} \{0,1\}$	$b \xleftarrow{\$} \{0,1\}$
$K \xleftarrow{\$} \mathcal{K}_\lambda$	$K \xleftarrow{\$} \mathcal{K}_\lambda$
if $b == 1$	**if** $b == 1$
$\quad b' \leftarrow \mathcal{A}^{F(K,\cdot)}(1^\lambda)$	$\quad b' \leftarrow \mathcal{A}^{(\$,F(K,\$))}(1^\lambda)$
else	**else**
$\quad g \xleftarrow{\$} \mathsf{Func}(\mathcal{D}_\lambda, \mathcal{R}_\lambda)$	$\quad g \xleftarrow{\$} \mathsf{Func}(\mathcal{D}_\lambda, \mathcal{R}_\lambda)$
$\quad b' \leftarrow \mathcal{A}^{g(\cdot)}(1^\lambda)$	$\quad b' \leftarrow \mathcal{A}^{(\$,g(\$))}(1^\lambda)$
return $b' == b$	**return** $b' == b$

Figure 12.2: 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 $\mathcal{D}_\lambda$.

Definition 14. Let $T \in \{\mathsf{wPR}, \mathsf{PR}\}$ and let $\mathsf{Exp}^T_{\mathcal{A},F}$ be defined as shown in Fig. 12.2. 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 $\mathcal{A}$.

In the experiments displayed in Fig. 12.2 keys are chosen uniformly randomly from the key space. As mentioned earlier, this approach is an abstraction of any means to derive keys used for the PRF, including using our randomness generator from Chapter 5 as a building block.

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 publicly known distribution. 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 adversaries, as q is bounded by a polynomial in λ, yielding the following theorem.

Theorem 5. *If F is weakly pseudorandom, then the trivial specification $\widehat{F} = F$ is subversion-resilient weakly pseudorandom in the offline watchdog with trusted amalgamation.*

12.5.1 Constructing Subversion-Resilient PRFs

As we now have access to subversion-resilient weak PRFs, the question becomes how we can elevate this to subversion-resilient PRFs. Revisiting classical results, we see that the classical Naor–Reingold construction [NR99] can be used to construct a (standard) PRF from a weak PRF that is subversion-resilient. An intuitive way to understand the Naor-Reingold construction is the following. Instead of directly feeding inputs into a weak PRF, the input is interpreted in a specific way. Each input bit is seen as a pointer to one of two (previously) randomly generated keys. These two keys are then fed into a weak PRF, which now obtains two random values as input. This way, two input bits lead to one output of a weak PRF. Based on this, we can build a tree structure to combine two evaluations of a weak PRF by again feeding their outputs into a weak PRF. For a simple visualisation of this approach, consider Fig. 12.3.

Thus, we can use this construction to construct subversion-resilient PRF but must allow our trusted amalgamation to handle adversarially chosen inputs and interpret single bits as pointers to well-generated keys to achieve this.

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 [BR17] 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,

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

A useful alternate interpretation is the following (with a simple example shown in Fig. 12.3). 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

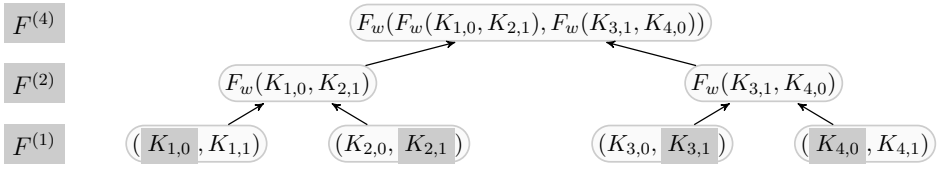


Figure 12.3: The alternate interpretation of the Naor-Reingold construction as labeling of a complete binary tree for the value $x = (0, 1, 1, 0)$. The corresponding leaf values are highlighted in grey.

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

Theorem 6 ([NR99, Thm. 5.1][2])**.** *Let $\ell \in \mathbb{N}$ with $\ell = 2^r$. If F_w is weakly pseudorandom, then $F^{(\ell)}$ is pseudorandom.*

Thus, we now consider the specification $\widehat{F^{(\ell)}} = (\mathsf{Am}, F_w)$ which takes inputs of length $l = 2^r$ for some $r \in \mathbb{N}$. The amalgamation function Am takes as input the weak PRF and parses adversially provided input according to the above description of the Naor-Reingold construction. Now, observe that a non-subverted, honest call structure to the underlying function F_w (provided by the trusted amalgamation) directly implies the subversion-resilience of $F^{(\ell)}$. At the lowest level, $F^{(1)}$ only returns completely random values (due to the random keys), which is clearly subversion-resilient. The inputs to $F^{(2)}$ are thus completely random values that follow a public input distribution, and Lemma 1 directly implies subversion-resilience.

Theorem 7. *If F_w is weakly pseudorandom, then for each ℓ with $\ell = 2^r$, $\widehat{F^{(\ell)}} = (\mathsf{Am}, F_w)$ is subversion-resilient pseudorandom in the offline watchdog with trusted amalgamation.*

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. We only choose one of two random values on the lowest level, corresponding to $F^{(1)}$. Hence, the watchdog can

[2]Naor and Reingold use the notion of a *synthesizer*, which are in our context equivalent to weakly PRFS [BR17].

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 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

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

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 6 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 additional trusted operations like an XOR. Alternatively, the randomized cascade construction by Maurer and Tessaro [MT08] can be used. There, adversarially chosen messages are directly fed into a prefix-free encoding, which must be modeled as a trusted operation to prevent input triggers. Another alternative is the IC construction by Maurer and Sjödin [MS07], where the input provided by the adversary is processed bit-wise, and either a weak PRF is executed, or a previously computed value is used in an iterative process.

12.6 Message Authentication Codes

We present the first construction of subversion-resilient MACs, where also the verification algorithm is subject to subversion. Our construction uses any subversion-resilient PRFs, e.g., the one from the previous section. Informally speaking, a MAC is a digital fingerprint of a message. It's like adding a seal to a message to prove it has not been tampered with during transmission. Utilizing symmetric keys, a MAC ensures a message's integrity so that if some tampers with the

message, the receiving party of the MAC will notice during verification. MACs are commonly used in communication protocols like TLS to ensure the data a browser sends and receives has not been tampered with during transit. Thus, it is an essential tool for modern digital communication.

Having established an intuition for MACs, we recall the standard formal 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 15. We call a triple $\mathsf{MAC} = (\mathsf{KGen}, \mathsf{Tag}, \mathsf{Vf})$ a *message authentication code* for key space $\mathcal{K}$, message space $\mathcal{M}$, and tag space $\mathcal{T}$ and each ppt algorithm is defined as follows:

- 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 perfect 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. This property captures that an adversary can not forge MACs while being allowed to obtain MACs for messages of its choice. The adversary is deemed successful if it can compute a tag T for a message M, where the pair (M, T) was not received as an answer to a previous oracle query. This captures *strong* unforgeability, as it allows the adversary to output forgeries for a message M it issued to its oracle, as long as it provides a different tag. This stronger property is necessary for the Encrypt-then-MAC approach to be secure [BN00, BN08].

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

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

$\mathsf{Query} \leftarrow \emptyset$

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

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

Figure 12.4: The forgery experiment for MACs. On input $M \in \mathcal{M}_\lambda$ the oracle $\mathsf{Tag}(K, \cdot)$ computes $T \leftarrow \mathsf{Tag}(K, M)$, stores (M, T) in Query and returns T.

Definition 16. Let MAC be a message authentication code and let $\mathsf{Exp}^{\text{SUF-CMA}}_{\mathcal{A},\mathsf{MAC}}$ be defined as shown in Fig. 12.4. 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 $\mathcal{A}$.

12.6.1 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

Key generation. KGen_F on input 1^λ outputs a uniform key $K \xleftarrow{\$} \mathcal{K}_\lambda$,

Tagging. Tag_F on input key $K \in \mathcal{K}_\lambda$ and message $M \in \mathcal{D}_\lambda$, returns $T = F(K, M)$, and

Verification. 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 8 ([GGM84b])**.** *If F is pseudorandom, then MAC_F is* SUF-CMA*-secure.*

Sketch. Since F is a PRF, no ppt adversary detects (with non-negligible probability) whether the tag oracle is implemented with a truly random function instead of the PRF. Thus, if we instantiated the above MAC construction with a truly random function $f \in \mathsf{Func}(\mathcal{D}, \mathcal{R})$, then any adversary can only forge a tag for some message m by guessing the image of m under f. This probability, however, is negligible. Hence, MAC_F is secure. □

Theorem 8 guarantees that the security of the underlying function F directly transfers to MAC_F. Thus, this approach directly grants subversion-resilience if the == operation during the verification is part of the trusted amalgamation, and we use a subversion-resilient PRF. Thus, $\widehat{\mathsf{MAC}_F} = (\mathsf{Am}, \widehat{F})$, where $\widehat{F}$ is the specification of a subversion-resilient PRF, and Am calls the PRF for tagging and verification it recomputes the MAC using the implementation of the PRF and compares the result with its input. Thus, the watchdog simply runs the watchdog of the underlying subversion-resilient PRF.

Theorem 9. *If $\widehat{F}$ is subversion-resilient pseudorandom , then $\widehat{\mathsf{MAC}_F} = (\mathsf{Am}, \widehat{F})$ is subversion-resilient* SUF-CMA*-secure in the offline watchdog with trusted amalgamation assuming a trusted == operation.*

Proof. The watchdog for $\widehat{\mathsf{MAC}_F}$ simply runs the watchdog for $\widehat{F}$. Thus, like in the proof of Theorem 8, no ppt adversary detects whether the tag oracle uses a truly random or $\widehat{F}$. This time, however, this follows from the subversion-resilience of the PRF rather than "standard" pseudorandomness. Therefore, if we instantiated $\widehat{\mathsf{MAC}_F}$ with a truly random function $f \in \mathsf{Func}(\mathcal{D}, \mathcal{R})$, then any adversary can only forge a tag for some message m by guessing the image of m under f. This probability, however, is negligible. Adding that tags, which would be rejected by the specification, are also rejected by our construction since the comparison operation is trusted, grants the above theorem. □

Additionally, we observe that $\widehat{\mathsf{MAC}_F}$ also guarantees perfect correctness. This is because the (deterministic) verification algorithm always recomputes tags the same way the (determinstic) tagging algorithm does and the comparison is a trusted operation. Thus, we see that canonical verification grants perfect correctness.

Theorem 10. $\widehat{\mathsf{MAC}_{F^{(\ell)}}} = (\mathsf{Am}, \widehat{F^{(\ell)}})$ *is perfectly correct.*

We can conclude that the PRF presented in Section 12.5.1 (built from a weak PRF) is a subversion-resilient, perfectly correct, and deterministic MAC as well.

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

12.7 Symmetric Encryption

This section discusses how to construct subversion-resistant symmetric encryption using a classical construction method based on PRFs. Symmetric encryption involves using a single key shared between two parties for encrypting and decrypting messages. Secure encryption guarantees confidentiality, meaning only the party with the secret key can decrypt the ciphertext to access the encrypted message. Symmetric encryption is much faster than asymmetric (public-key) encryption. This is why it is commonly used to encrypt large amounts of data. However, the downside is that that if two parties want to use symmetric encryption to encrypt their communication, they need a way to agree on a common secret key before communication.

More formally, 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 $\mathcal{C} = \bigcup_{\lambda\in\mathbb{N}} \mathcal{C}_\lambda$. Our construction does not increase ciphertext size compared to the standard construction of encryption from PRFs.

Definition 17. 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 $\mathcal{C}$, and each ppt algorithm is defined as follows:

- 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 $\perp$.
- 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 $\perp$.

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 $C \neq \perp$. For security, we consider IND\$-CPA-security, i.e., indistinguishability from random bits [RBBK01, Rog02], which can be shown to imply IND-CPA-security in the left-or-right sense (see, e.g., [BDJR97]) by a straightforward reduction.

Definition 18. Let SE be a symmetric encryption scheme and let $\mathsf{Exp}^{\text{IND\$-CPA}}_{\mathcal{A},\mathsf{SE}}$ be defined as shown in Fig. 12.5. We define

$$\mathsf{Adv}^{\text{IND\$-CPA}}_{\mathcal{A},\mathsf{SE}}(1^\lambda) := \left|\Pr[\mathsf{Exp}^{\text{IND\$-CPA}}_{\mathcal{A},\mathsf{SE}}(1^\lambda) = 1] - 1/2\right|$$

and say that SE is IND\$-CPA-secure if $\mathsf{Adv}^{\text{IND\$-CPA}}_{\mathsf{SE}}(\mathcal{A})$ is negligible for all ppt adversaries $\mathcal{A}$.

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

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

$b \xleftarrow{\$} \{0,1\}$

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

return $(b == b')$

Figure 12.5: 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 $C \leftarrow \mathsf{Enc}(K, M)$ and if $C = \perp$, outputs $\perp$, and otherwise, outputs a random string of length $|C|$.

Symmetric Encryption from PRFs. In Section 12.5, we construct subversion-resilient PRFs. A classical use case of PRFs is the construction of symmetric encryption. While one can use PRFs to generically construct a fixed-length symmetric encryption scheme (or a block cipher), we can also use PRFs to construct *stream ciphers* for messages of (almost) arbitrary length. Stream Ciphers produce a pseudorandom keystream, which is used to encrypt data via a bitwise XOR. 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 $M := \mathrm{KS}(K, \mathsf{IV}) \oplus C$.

12.7.1 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 chooses uniformly random IVs (with a subversion-resilient randomness generator) and then calls the underlying PRF and applies the trusted XOR operation.

Theorem 11. *If $\widehat{\mathrm{KS}}$ is subversion-resilient weakly pseudorandom, then $\widehat{\mathsf{SE}_{\mathrm{KS}}} = (\mathsf{Am}, \widehat{\mathrm{KS}})$ is subversion-resilient* IND\$-CPA*-secure in the offline watchdog with trusted amalgamation given that the randomness generation and XOR operation are trusted.*

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 be instantiated. The (randomized) counter mode (CTR\$) is a popular instantiation of the above construction. 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) as well as the XOR operation is modeled as part of the amalgamation.

Theorem 12. *If $\widehat{F}$ is subversion-resilient pseudorandom, then $\widehat{\mathrm{KS}_{\mathrm{CTR}}} = (\mathsf{Am}, \widehat{F})$ is subversion-resilient weak pseudorandom in the offline watchdog with trusted amalgamation assuming randomness generation, the XOR operation and the handling of the counter are part of the trusted amalgamtion.*

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 could directly 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 the above theorem could alternatively also be proven directly, as the KDF has a public input distribution since both K and IV are chosen uniformly at random. Then, however, the watchdog would need to test the KDF with random K and IV for l blocks polynomial many times.

Furthermore, $\mathrm{KS}_{\mathrm{CTR}}$ is not a PRF, as the adversary can 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 11 and 12 implies 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 subversion-resilient* IND$-CPA*-secure in the offline watchdog with trusted amalgamation assuming that randomness generation, the XOR operation and the handling of the counter are part of the trusted amalgamation.*

Thus, IND$-CPA security directly follows from the security of the underlying PRF. While security is preserved, this does not automatically mean that correctness is also preserved: the decryption algorithm is *not* executed in the IND$-CPA security experiment but is fundamental for correctness. As discussed by Russell, Tang, Yung, and Zhou [RTYZ17] this would allow for censorship of chosen messages. If we 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 outputs 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 implements the underlying PRF, so our construction automatically satisfies *perfect* correctness.

Theorem 13. *The specification* $\widehat{\mathsf{SE}_{\mathrm{KS}}}$ *is perfectly correct.*

Proof. Correctness follows from the "canonical decryption" [3] 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

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

out by the $\oplus$ operation:

$$\begin{aligned}\widetilde{\mathsf{Dec}}(K, \widetilde{\mathsf{Enc}}(K, M)) &= \widetilde{\mathsf{Dec}}(K, (\widetilde{\mathrm{KS}}(K, IV) \oplus M)) \\ &= \widetilde{\mathrm{KS}}(K, IV) \oplus \widetilde{\mathrm{KS}}(K, IV) \oplus M \\ &= M\end{aligned}$$

□

Note that Russell et al. [RTYZ17] also achieved correctness for subversion-resilient symmetric 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 and the trusted XOR, we can guarantee *perfect* correctness.

12.8 Constructing Subversion-Resilient Authenticated Encryption

Given subversion-resilient building blocks, we now see that the classical Encrypt-then-MAC approach grants us subversion-resilient authenticated encryption, for which we now provide a formal definition.

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 19. We call a triple $\mathsf{AD} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ a *authenticated encryption scheme with associated data* for key space $\mathcal{K}$, message space $\mathcal{M}$, data space $\mathcal{D}$, and ciphertext space $\mathcal{C}$, and each *ppt* algorithm is defined as follows:

- 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 $M \in \mathcal{M}_\lambda$.

A *authenticated 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 $\mathsf{Dec}(K, \mathsf{Enc}(K, (M, D)), D) = (M, D)$.

$\mathsf{Exp}^{\mathrm{AE}}_{\mathcal{A},\mathsf{AD}}(1^\lambda)$

$\mathcal{Q} \leftarrow \emptyset$

$b \xleftarrow{\$} \{0,1\}$

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

if $b == 1$

  $b' \leftarrow \mathcal{A}^{\mathsf{Enc}(K,\cdot,\cdot),\mathsf{Dec}(K,\cdot,\cdot)}(1^\lambda)$

else

  $b' \leftarrow \mathcal{A}^{\$_K(\cdot,\cdot),\perp(\cdot,\cdot)}(1^\lambda)$

return $b' == b$

Oracle $\$_K(M, D)$

$C \xleftarrow{\$} \{0,1\}^{|C|}$

$\mathcal{Q} = \mathcal{Q} \cup \{((M, D), C)\}$

return C

Oracle $\perp(M, D)$

if $((M, D), C') \in \mathcal{Q}$

  return (M, D)

else

  return $\perp$

Figure 12.6: 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 $D \in \mathcal{D}_\lambda$.

Definition 20. Let AD be an authenticated encryption scheme with associated data and let $\mathsf{Exp}^{\mathrm{AE}}_{\mathcal{A},\mathsf{AD}}$ be defined as shown in Fig. 12.6. We define

$$\mathsf{Adv}^{\mathrm{AE}}_{\mathcal{A},\mathsf{AD}}(1^\lambda) := \left|\Pr[\mathsf{Exp}^{\mathrm{AE}}_{\mathcal{A},\mathsf{AD}}(1^\lambda)] - 1/2\right|$$

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. This case is excluded in most works as this would trivially break security (and the adversary already knows the answer to the query). 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 the 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 to defend against this kind of attacks. As seen by our construction of stream ciphers with "canonical decryption" in combination with a trusted XOR operation and our MAC construction with canonical verification, perfect correctness is guaranteed. To include this attack in our model, we change the behavior of the oracles in the experiment for $b = 0$ by introducing bookkeeping 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 could distinguish the two worlds of the experiment otherwise.

12.8.1 Achieving Subversion-Resilience via Encrypt-then-MAC

The classical way to construct authenticated encryption relies on using a MAC that is applied on a ciphertext, *after* a message is encrypted. Decryption then first verifies the MAC and afterwards performs the underlying decryption algorithm. Due to the strong unforgeability of the MAC, an adversary cannot use decryption, and the security experiment reduces to IND$-CPA-security. This is useful for achieving subversion-resilience, as symmetric encryption, which is IND$-CPA secure under subversion, does not guarantee security 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 is 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. Combined with a subversion-resilient encryption scheme, the adversary must 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 authenticated 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 $K_{\mathsf{MAC}} \leftarrow \mathsf{KGen}_{\mathsf{MAC}}(1^\lambda)$, another key $K_\$ \leftarrow \mathsf{KGen}_\$(1^\lambda)$ 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 $C_\$ \leftarrow \mathsf{Enc}_\$(K_\$, M)$, then produces a tag $T \leftarrow \mathsf{Tag}(K_{\mathsf{MAC}}, C_\$ \mid\mid D)$, 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 $b = \mathsf{Vf}(K_{\mathsf{MAC}}, C_\$ \mid\mid D, T)$. If $b = 0$ (i e. the verification failed), it returns $\perp$. Else, it computes $M \leftarrow \mathsf{Dec}_\$(K_\$, C_\$)$ 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 14 ([BN08])**.** *If* MAC *is* SUF-CMA*-secure and* $\mathsf{SE}_\$$ *is* IND$-CPA*-secure, then* AD *is* AE*-secure.*

Thus, we consider the specification $\widehat{\mathsf{AD}} = (\mathsf{Am}, (\widehat{\mathsf{SE}}, \widehat{\mathsf{MAC}}))$ where the amalgamation function Am builds the authenticated encryption scheme as described

above. First, it uses the specification of the encryption scheme to encrypt the message. Afterwards, it uses the specification of the MAC to compute a tag over the previously obtained ciphertext.

Correctness of SE and MAC is directly inherited by AD. Note that in [BBC24], only correctness of SE was required. This however is not sufficient to argue that AD is also perfectly correct. While the security of MAC guarantees that no invalid tags are accepted, it does not guarantee that valid tags get accepted.

Theorem 15. *If* $\widehat{\mathsf{SE}_\$}$ *and* $\widehat{\mathsf{MAC}}$ *are 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 and the MAC are subversion-resilient, following the proof idea of Bellare and Namprempre [BN00, BN08].

Theorem 16. *If* $\widehat{\mathsf{SE}_\$}$ *is subversion-resilient* IND$-CPA*-secure,* $\widehat{\mathsf{MAC}}$ *is subversion-resilient* SUF-CMA*-secure and* $\widehat{\mathsf{SE}_\$}$ *and* $\widehat{\mathsf{MAC}}$ *are perfectly correct, then* $\widehat{\mathsf{AD}} = (\mathsf{Am}, (\widehat{\mathsf{SE}_\$}, \widehat{\mathsf{MAC}}))$ *is subversion-resilient* AE*-secure in the offline watchdog with trusted amalgamation and perfectly correct.*

Sketch. The watchdog for $\widehat{\mathsf{AD}}$ runs the watchdog for $\widehat{\mathsf{SE}}$ and $\widehat{\mathsf{MAC}}$. Note that there is no need to test the components in combination, as both allow arbitrary input distributions while being subverted and executed independently. Assuming that the watchdog does not detect subversion, we can follow the proof of Bellare and Namprempre [BN00, BN08] for the Encrypt-then-MAC approach. The main difference is to base the security on the subversion-resilience of the building blocks rather than the "standard" security properties. However, these only differ by the watchdog's preceded check and the implementation usage 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. Suppose the output of the encrypt oracle was handed to the decryption oracle. In that case, the adversary always obtains the correct answer, either by the correctness of the encryption scheme for $b = 1$ or due to the bookkeeping if $b = 0$. Now assume C was not obtained via the encrypt oracle. If C is malformed and the decryption oracle returns a $\perp$ symbol, the adversary cannot distinguish this from an oracle that 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. Again, This case is impossible since any adversary reaching this case would have successfully forged a tag for (M, D), thus breaking the 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 effectively use 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. □

12.9 Discussion

As we have presented our construction, let us discuss our results and alternative approaches for proofing our results.

Using weak PRFs instead of PRFs. In our construction of subversion-resilient authenticated encryption, one may think that using weak PRFs instead of "standard" PRFs as a building block may be sufficient. However, it is unclear how to obtain MACs from (subversion-resilient) weak PRFs, as the security of MACs against forgery attacks is 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 [WC81] construction), handling the final forgery query is a challenge, as this query is directly made on the verification algorithm.

The Price of Complete Subversion without Random Oracles. Achieving subversion-resilience under *complete* subversion is crucial for meaningful security guarantees in practice. Excluding certain algorithms from subversion seems hard to justify if the adversary is even able to alter a standardized cryptographic scheme, as illustrated by the Dual_EC incident. As mentioned earlier, defending against subversion attacks on the receiving algorithms, i.e., decryption and verification, is challenging in an offline watchdog model. This is due to crippling input trigger attacks, quickly leading to impossibility statements without further assumptions. Thus, we demonstrate that defending against such attacks is possible, even in an offline watchdog model. Unfortunately, this comes with a price. We require a few trusted operations and more fine-grained access to the building blocks. All of these assumptions seem unavoidable to us as long as no other or additional assumptions are being made. For a deeper discussion on necessary assumptions, consider Chapter 14. Random oracles are a valuable tool in this context, as subversion-resilient random oracles allow mapping adversarially chosen inputs to another domain without the risk of input triggers. However, as discussed in Section 2.3, there are limitations to random oracles. Our contributions show that subversion-resilience under complete subversion can be achieved, even *without* random oracles.

13 Subversion-Resilient Digital Signatures

After we designed subversion-resilient authenticated encryption in Chapter 12, the natural question becomes for which other primitives we can construct watchdogs under complete subversion. A natural choice are digital signature (DS). Digital signatures are often seen as the digital equivalent of handwritten signatures. They are used for verifying the authenticity of digital messages. Given a valid digital signature on some message, any other party (a.k.a. the verifier) can assure that the signature was indeed issued by a signer known to the verifier. Regarding security, it should be hard for other parties to forge signatures on behalf of a signer. To achieve this, digital signatures make use of public-key cryptography. A signer produces two keys: a private signing key and a public verification key. The signing key, only known by the signer, is used to compute signatures, while the verification key is publicly known and can be used by every party to verify signatures.

Digital signatures play a vital role in constructing modern-day cryptographic protocols. A prominent example is TLS, where they are primarily used for authentication. For example, if connecting to a website using HTTPS, the server provides a digital certificate, including the server's public key. This certificate is signed by a trusted Certificate Authority (CA). A client then uses the CA's public key to verify this signature. If verification is successful, the client can be sure it connects to its intended communication partner. Not only are digital signatures used in TLS, but they are also utilized in software distribution and other scenarios. Thus, as an essential building block in various modern applications, digital signatures are an attractive target for backdoor attacks. Unlike the previous parts of this thesis, there are several works available that already propose subversion-resilient signatures, even under complete subversion where *all* algorithms are provided by the adversary. We continue by highlighting prior work on subversion-resilient signatures in watchdog models to obtain a good understanding of the state-of-the-art constructions as well as their limitations. After we present our approach, we highlight previous works in other models.

Online Watchdogs. Russell et al. [RTYZ16] proposed the first subversion-resilient signature scheme in the online watchdog model with random oracles. In particular, they consider the *complete-subversion* setting where all cryptographic 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, Russell et al. [RTYZ16] also proposed subversion-resilient one-way permutations. They considered a stronger security setting, where the adversary can choose the function index (public parameters pp in Definition 22). This, in turn, makes the use of random oracles necessary. We observe that this stronger notion is not needed to construct subversion-resilient signatures, and we can thus remove the random oracle dependency.

Offline Watchdogs. Chow et al. [CRT+19] improved Russell et al.'s [RTYZ16] construction by presenting two schemes with offline watchdogs. They bypassed Russell et al.'s impossibility [RTYZ16] by using a more fine-grained split-program model adopted in [RTYZ17] 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 a domain-separation technique and a 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. [RTYZ18].

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 subversion-resilient signatures without random oracles in the offline watchdog model under complete subversion?

We can answer this question affirmatively. Similar to our approach for achieving subversion-resilient authenticated encryption, we utilize that we can build signatures from smaller building blocks. We see that classical results of cryptography can be adapted to grant subversion-resilient signatures under complete subversion. Notably, this time, indeed *all* algorithms are subject to subversion, although we rely on known methods to protect key generation. This captures the use of digital signatures in practice, where one party computes all key material and publishes its verification key. However, we want to emphasize that

we apply known techniques from prior work and focus on our contributions to protecting the verification algorithm.

Before we summarize our results and discuss other approaches to obtain subversion-resilient signatures, let us first highlight the challenges in designing subversion-resilient signatures in an offline watchdog model under complete subversion.

13.1 Technical Challenges

The difficulty of designing signature schemes that are secure against complete subversion mainly lies in the problem of restoring the security of the subverted verification algorithm. The main challenge is twofold:

The first reason is (yet again) the existence of *input trigger attacks*. Input trigger attacks on digital signatures are very similar to those on authenticated encryption, which we presented in the previous chapter. 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 polynomial time watchdog with only black-box access to the signatures scheme has only a negligible chance of detecting this, while the attacker can trivially break security. Intuitively, this attack is possible as *input distribution* to the verification algorithm is determined by the adversary during the run of the security experiment. Thus, for an offline watchdog that is executed *before* the security experiment, it is difficult to test the verification algorithm according to the same distribution and outrule the above attack. This holds unless we assume some strategy by the adversary. One example of this is subversion under random message attacks [AMV15], where the adversary is only allowed to ask for signatures on random messages. Thus, similar to our contributions from Section 12.8, more fine-grained access to the verification algorithm seems again necessary to defend against this simple and generic attack while not limiting the attack capabilities of the adversary.

The second big challenge is that a common technique to develop signature schemes based on symmetric primitive as first proposed by Merkle [Mer90], uses collision-resistant hash functions. But, due to the structure of the security experiment of collision-resistant hash functions, they are also highly vulnerable to input trigger attacks (see Section 13.9 for a more detailed discussion). Previous constructions thus needed heavy machinery such as the random oracle model [CRT+19, RTYZ18], sophisticated online watchdogs [AMV15, RTYZ18], or a trusted initialization phase [FM18]. Approaches using a collision-resistant hash function in the standard model with an offline watchdog thus seem somewhat futile.

Both issues around input triggers described above occur since the adversary can prepare its implementation with input triggers as the challenge computed by the adversary is directly input in the (possibly) subverted implementation.

To avoid this, in our constructions, we utilize building blocks, where the adversary needs to commit to its implementation before a challenge, and values within the construction are chosen by the security experiment. This way, the adversary can freely choose the inputs to our construction, but since the implementation is set up before the challenge, it is hard for the adversary to adapt. As mentioned earlier, Russell et al. [RTYZ18] 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 we show later 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 afterwards, the adversary tries to find a collision. The task thus becomes to construct target-collision-resistant hash functions in a subversion-resilient manner.

Practical Interpretation. The above approach of choosing challenges and random values after the implementation is provided is sound in our theoretical model. One might wonder whether such an approach is a theoretical artifact, similar to the abstraction of key generation for symmetric key primitives, and thus is little meaningful for practice. Since this is not the case, let us briefly discuss why this approach is also a meaningful model for practical use. The security experiment choosing a random challenge resembles the way the considered primitive is often used as a building block. If during the security experiment, a uniformly random input is chosen, a construction will also usually choose a uniformly random input to compute outputs in order to simulate the security experiment correctly. Concretely, we can choose fresh random coins in our construction by utilizing subversion-resilient randomness generators from prior or this work to remove possible biases and prevent input triggers. Thus, the security model captures how a building block will actually be used in a bigger construction. Then one might ask, whether this still is meaningful in a subversion setting. Even though a powerful adversary provides the implementation, we do not deem it that powerful that it is able to adjust its implementation for every random value chosen by a primitive during runtime. For example, changing the implementation of a popular Instant Messaging app might be within the scope of current real-life adversaries. However, we deem it unlikely for subversion attacks to be able to continuously update the implementation, potentially even every time after some random value is chosen while remaining undetected. Here, detection refers to noticing a change in the implementation for example via updates, and not the testing of the watchdog.

13.2 Our Contributions

Our overall approach is illustrated in Fig. 13.1 and our main contributions can be summarized as follows.

- We show that a standard one-way permutation (OWP) is a subversion-resilient one-way function (OWF) 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 [RTYZ16]. Our construction of OWFs does not rely on random oracles but rather only needs standard OWP as a building block.
- 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 target-collision-resistant (TCR) hash functions from a random target-collision-resistant (rTCR) hash function, where in the security experiment one input is chosen uniformly at random from the domain rather than provided by the adversary, can be used to obtain subversion-resilient TCR hash functions. This approach uses a trusted XOR and a random blinding value to randomize the input provided by the adversary.
- From subversion-resilient OTS, subversion-resilient target-collision resistant hash functions, and subversion-resilient PRFs, we build subversion-resilient signature via the classical Naor-Yung [NY89] construction of digital signatures.

Thus, similar to the work of Chow et al. [CRT+19], 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 seen in the previous chapter 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 critical insight of this thesis 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 other subversion-resilient signatures. For an overview, consider Table 13.1.

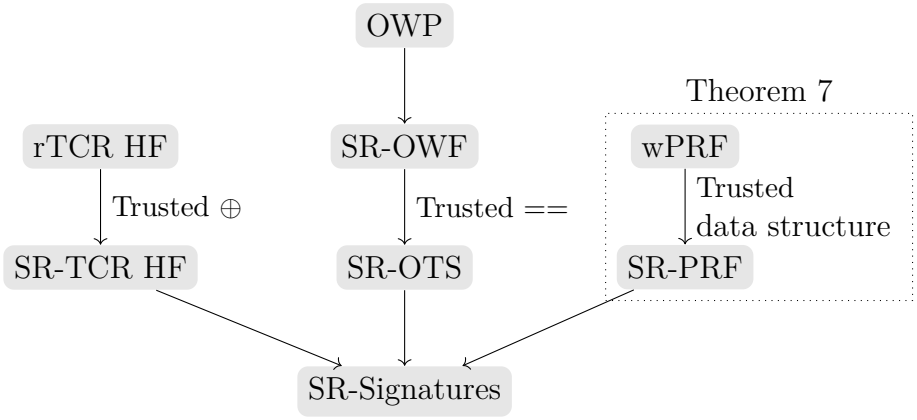


Figure 13.1: Overview of our construction. Here, SR denotes subversion-resilience, HF denotes hash function, OWP/OWF denotes one-way permutation/function, PRF denotes pseudo-random function, (r)TCR denotes (random) target collision-resistance, and OTS denotes one-time signature. Comments beside arrows highlight trusted operations.

13.3 Subversion-Resilient Signatures in Other 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 [AMV15], Ateniese et al. showed that signature schemes with unique signatures are subversion-resilient against all subversion attacks that meet the *verifiability condition*. This condition essentially requires that signatures produced by the subverted signature algorithm should almost always be verified correctly under the target verification key.[1] They adopted the reverse firewall model for constructing subversion-resilient signature schemes to relax such a strong requirement. Mironov and Stephens-Davidowitz [MS15] originally introduced the notion of RFs, 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

[1] In [AMV15], only subverted signing algorithms are considered while both key generation and verification algorithms are assumed to be trusted.

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 a self-destruct capability, which means that the RF can publicly check the validity 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 universal offline watchdog model.

Subversion-Resilient Signatures via Self-guarding. Fischlin and Mazaheri [FM18] proposed a novel defense mechanism called *self-guarding algorithms*, 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 bit-wise 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.

13.4 Outline

We continue by proving that one-way permutations constitute subversion-resilient one-way functions in Section 13.6. Afterwards, we provide a construction for subversion-resilient TCR hash functions in Section 13.7. Finally, in Section 13.8, we combine these ingredients to obtain subversion-resilient signatures.

13.5 Simplifying Assumptions

For our upcoming construction of subversion-resilient signatures, we will need access to uniformly random coins and subversion-resilient PRFs. Fortunately, for both primitives, there are already subversion-resilient schemes available. For randomness generation, we can either use our randomness generator from Chapter 5 or the randomness generator from Russell et al. [RTYZ17]. To help readability and focus on the challenge of protecting the verification algorithm, we will not explicitly state the use of a subversion-resilient randomness generator and simply assume access to uniformly random bits. Thus, our results can be extended by an additional game hop in our security proof and change the subverted randomness generator with a subversion-resilient randomness generator. Note that we will use these random coins, especially for key generation, and we assume the trusted amalgamation model includes the split-program model, and we are therefore able to test algorithms while using specified random coins. Thus, we can follow prior work by Russell et al. [RTYZ17] and guarantee that key generation can be efficiently tested utilizing Lemma 1, guaranteeing that the watchdog either detects subversion or keys are generated honestly. In the following, we will only cover this briefly for completeness and put more emphasis on our new contributions. For PRFs, we already showed in Section 12.5 how to construct subversion-resilient PRFs from weak PRFs. Thus, in the following, we assume that subversion-resilient PRFs are available, can be used in a black box fashion, and avoid adding additional notational overhead by restating the specific construction from Section 12.5.

13.6 One-Way Functions

As we have seen in Section 12.5, certain primitives are *inherently* subversion-resilient in our considered model. One such primitive is weak PRFs, which do not allow the attacker to input any value during the security game but rather choose random inputs for the oracle during the experiment. 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 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 use later to construct one-time signatures, particularly Lamport signatures [Lam79]. 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 is 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.

13.6.1 One-Way Functions and Permutations

A OWF is a function that is easy to compute on all inputs but is hard to invert given an image of a randomly chosen input. It is currently not known whether one-way functions, in fact, do exist. However, there are several candidates for which there are no algorithms that break the security of these candidates. To give an illustration, we briefly describe two popular candidates in cryptography. The first candidate is the multiplication (and factoring) of two large prime numbers. The function thus takes two primes p and q as inputs and returns the product $N = p \cdot q$. Inverting requires the factorization of N, for which currently no efficient algorithms are known. The famous RSA encryption scheme [RSA78] is based on the assumption that the above function is hard to invert. Another famous candidate is discrete exponentiation (and logarithm). Here, a finite abelian group $\mathbb{G}$ for order $n \in \mathbb{N}$ is given, together with a generator g. Given a random element k with $1 \leq k \leq n$, the function then computes $c = g^k$. While this evaluation can be computed efficiently, there are groups for which no efficient algorithms are known that compute the logarithm $k = \log_g c$. Typical choices for the group $\mathbb{G}$ are cyclic groups or cyclic subgroups of elliptic curves over finite fields.

In addition to one-way functions, we also use a primitive called OWP. These are one-way functions, which, in addition, are also permutations. Famous candidates for OWP are trapdoor functions like the RSA [RSA78] or Rabin cryptosystem [Rab79].

We recall the standard definition of OWFs and OWPs. These use families of one-way functions (and permutations) rather than addressing single functions. This is because most candidates' domain and range might differ depending on the security parameter. In the following, we assume inputs are bitstrings. Our following approach could also potentially be applied as long as there is a trusted

mapping between the range and domain of the one-way function and bistrings.

Definition 21. A (family of) *one-way functions* Π consists of two ppt algorithms KGen and Eval, where each algorithm is defined as follows:

- The randomized algorithm KGen returns the *public parameters* pp upon input the security parameter 1^λ.
- 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 $\mathsf{Eval}(\mathsf{pp}, \cdot)$ is a permutation on $\{0,1\}^\lambda$, we call Π a family of *one-way permutations*.

With this definition, we can define security for one-way functions and one-way permutations. In the case of one-way functions, the security experiment samples a uniformly random element from the domain and hands the adversary an evaluation of that element. In the case of one-way permutations, the adversary is simply handed a uniform element from the range of the function. In both cases, the adversary is afterwards asked to return a preimage of the value given to it.

Definition 22. Let $\Pi = (\mathsf{KGen}, \mathsf{Eval})$ be a *one-way function* or *one-way permutation* and let $\mathsf{Exp}^{\mathsf{Inv}}_{\mathcal{A},\Pi}$ be defined as shown in Fig. 13.2. We define

$$\mathsf{Adv}^{\mathsf{Inv}}_{\mathcal{A},\Pi}(1^\lambda) := \left|\Pr[\mathsf{Exp}^{\mathsf{Inv}}_{\mathcal{A},\Pi}(1^\lambda) = 1]\right|$$

and say that Π is *secure* if $\mathsf{Adv}^{\mathsf{Inv}}_{\mathcal{A},\Pi}$ is negligible for all ppt adversaries $\mathcal{A}$.

$\mathsf{Exp}^{\mathsf{Inv}}_{\mathcal{A},\Pi}(1^\lambda)$

$(\mathsf{pp}) \leftarrow \mathsf{KGen}(1^\lambda)$
if Π is a permutation
  $y^* \xleftarrow{\$} \{0,1\}^\lambda$
else
  $x \xleftarrow{\$} \{0,1\}^\lambda$
  $y^* = \mathsf{Eval}(\mathsf{pp}, x)$
$x^* \leftarrow \mathcal{A}(\mathsf{pp}, y^*)$
if $\mathsf{Eval}(\mathsf{pp}, x^*) == y^*$
  return 1
return 0

Figure 13.2: One-way function/ -permutation security experiment.

Note that other definitions exist in which x^* is chosen uniformly at random from the domain and $y^* = \mathsf{Eval}(\mathsf{pp}, x^*)$ 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 use this more straightforward way of defining security.

13.6.2 Subversion-Resilient One-Way Functions

We see that starting from any "standard" 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 11.

The idea is that by applying Lemma 1 and using that we use 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 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 17. *Let* $\Pi = (\mathsf{KGen}, \mathsf{Eval})$ *be a one-way permutation. Then, the trivial specification* $\widehat{\Pi} = (\mathsf{KGen}, \mathsf{Eval})$ *is a subversion-resilient secure one-way function in offline watchdog model with trusted amalgamation, assuming randomness generation is part of the trusted amalgamation.*

Proof. Let $\widehat{\Pi} = (\mathsf{KGen}, \mathsf{Eval})$ be the specification of a permutation, and $\widetilde{\Pi}$ be the implementation of $\widehat{\Pi}$ provided by $\mathcal{A}$. First, the watchdog simply tests $\widetilde{\mathsf{KGen}}$ polynomially many times using uniformly random coins. Afterwards, it runs KGen for pp, samples uniformly random x and compares $\widetilde{\mathsf{Eval}}(\mathsf{pp}, x)$ to $\widehat{\mathsf{Eval}}(\mathsf{pp}, x)$. Whenever a mismatch between the specifications and the implementation is found, the watchdog returns "true". Following Russell et al. [RTYZ17], we argue that either the watchdog detects subversion or pp is computed according to the specification. To prove the subversion-resilience, let $\mathcal{T} \subseteq \mathcal{PP} \times \{0,1\}^\lambda$ denote the *trigger set* such that $(\mathsf{pp}, x) \in \mathcal{T} \Leftrightarrow \widetilde{\mathsf{Eval}}(\mathsf{pp}, x) \neq \widehat{\mathsf{Eval}}(\mathsf{pp}, x)$ 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 $\widetilde{\mathsf{Eval}}(\mathsf{pp}, x^*) = y^*$. We now distinguish whether the adversary uses a trigger, i.e., whether $(\mathsf{pp}, x^*) \in \mathcal{T}$ or $(\mathsf{pp}, x^*) \notin \mathcal{T}$ holds. If $(\mathsf{pp}, x^*) \notin \mathcal{T}$, we know that $y^* = \widetilde{\mathsf{Eval}}(\mathsf{pp}, x^*) = \widehat{\mathsf{Eval}}(\mathsf{pp}, x^*)$. 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 not predict $\widetilde{\mathsf{Eval}}(\mathsf{pp}, x^*)$, however it can only redistribute weight within $\mathcal{T}$, as $\mathsf{Eval}(\mathsf{pp}, \cdot)$ 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 $\widetilde{\mathsf{Eval}}(\mathsf{pp}, x^*) = y^*$ exists is negligible. □

13.7 Hash Functions

Another crucial building block we use are hash functions. Generally speaking, a hash function is a function that maps inputs of arbitrary length to a fixed-size range. In this thesis, we are interested in cryptographic hash functions, which we simply call hash function (HF). These are hash functions with additional properties. The most commonly found in the literature is called *collision-resistance*. For a collision-resistant hash function, it is hard to find two different inputs that are mapped to the same output (and thus *collide*). Cryptographic hash functions are found in various applications as building blocks for other cryptographic primitives. Most notably, they are used in digital signatures and message authentication codes to reduce the input size before signing or tagging. However, hash functions are also used in other scenarios, such as checksums, to detect accidental data corruption.

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 Naor and Yung [NY89] work. Unfortunately, subversion-resilient collision-resistant hash functions seem impossible (without any additional assumptions), as we discuss in Section 13.9. On the positive side, just like in the case of standard signatures, subversion-resilient target collision-resistant hash functions are sufficient for our case, and we see that they can be constructed using a trusted XOR. So, let us begin by providing

the necessary (security) definitions.

Definition 23. A *family of hash functions* $\mathcal{H}$ is a pair of ppt algorithms $(\mathsf{Gen}, \mathsf{H})$ where each algorithm is defined as follows:

- The randomized algorithm Gen takes as input the security parameter 1^λ and outputs a (non-secret) key s.
- The deterministic algorithm 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, which is sufficient for our use case. We 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.

Keyed vs. Unkeyed. In practice, hash functions are not keyed. Following the approach of Rogaway [Rog06], our theorems could be rephrased such that breaking the security of some schemes using secure hash functions leads to finding collisions. We chose to use keyed functions instead, as the construction of the TCR hash function that we use explicitly introduced a blinding value, which is most conveniently modeled as part of a key. A similar approach was used by Russell et al. [RTYZ18] for their construction of subversion-resilient random oracles. For a deeper discussion on keyed and unkeyed hash functions, consider [KL14].

We consider two different but related security notions concerning hash functions in the following. *Target-collision-resistance* describes the notion that if an adversary outputs some x, it can not output a y such that x and y collide, given that the key of the hash function was chosen *after the adversary provided* x.

Definition 24. Let $\mathcal{H} = (\mathsf{Gen}, \mathsf{H})$ be a family of hash functions and let $\mathsf{Exp}^{\mathsf{TCR}}_{\mathcal{A},\mathcal{H}}$ be defined as shown in Fig. 13.3. We define

$$\mathsf{Adv}^{\mathsf{TCR}}_{\mathcal{A},\mathcal{H}}(1^\lambda) := \left|\Pr[\mathsf{Exp}^{\mathsf{TCR}}_{\mathcal{A},\mathcal{H}}(1^\lambda) = 1]\right|$$

and say that $\mathcal{H}$ is TCR if $\mathsf{Adv}^{\mathsf{TCR}}_{\mathcal{A},\mathcal{H}}$ is negligible for all ppt adversaries $\mathcal{A}$.

We see that the weaker notion of *random target-collision-resistance* is incredibly useful for our approach. This notion is the same as target-collision-resistance, but the adversary does not choose the value x but rather chooses uniformly at random from the domain of the hash function.

Definition 25. Let $\mathcal{H} = (\mathsf{Gen}, \mathsf{H})$ be a family of hash functions and let $\mathsf{Exp}^{\mathsf{rTCR}}_{\mathcal{A},\mathcal{H}}$ be defined as shown in Fig. 13.3. We define

$$\mathsf{Adv}^{\mathsf{rTCR}}_{\mathcal{A},\mathcal{H}}(1^\lambda) := \left|\Pr[\mathsf{Exp}^{\mathsf{rTCR}}_{\mathcal{A},\mathcal{H}}(1^\lambda) = 1]\right|$$

and say that $\mathcal{H}$ is rTCR if $\mathsf{Adv}^{\mathsf{rTCR}}_{\mathcal{A},\mathcal{H}}$ is negligible for all ppt adversaries $\mathcal{A}$.

$\mathsf{Exp}^{\mathsf{TCR}}_{\mathcal{A},\mathcal{H}}(1^\lambda)$	$\mathsf{Exp}^{\mathsf{rTCR}}_{\mathcal{A},\mathcal{H}}(1^\lambda)$
$(x, \mathsf{st}) \leftarrow \mathcal{A}_0(1^\lambda)$	$x \xleftarrow{\$} \mathcal{D}$
$s \leftarrow \mathsf{Gen}(1^\lambda)$	$s \leftarrow \mathsf{Gen}(1^\lambda)$
$y \leftarrow \mathcal{A}_1(s, x, \mathsf{st})$	$y \leftarrow \mathcal{A}(s, x)$
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

Figure 13.3: (random) Target-collision resistance security experiment for hash functions. In the left experiment, we use that $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ 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 inspired by the attack on one-way permutations by Russel et al. in [RTYZ16]. 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 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. We must run a watchdog for each input length individually to guarantee subversion-resilience. However, this seems unavoidable to prevent the above attack.

Construction. Similar to our construction of subversion-resilient one-way functions, we use the fact that the rTCR hash function has 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

XOR'ed to the input before evaluating the hash function. Our watchdog tests Gen for uniformly random coins to sanitize key generation. Thus, just as in [RTYZ17], 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 [RTYZ17] or our construction from Chapter 5 to produce random coins in a subversion-resilience manner that does not use random oracles.

We note that the XOR operation is 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. [RTYZ17]. Just like in the section about one-way function, 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 construction to obtain a TCR hash function from a rTCR hash function.[2]

In the proof, we use that target collision-resistant hash functions with small input domains 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.*

Thus, we can state our final result on hash functions. Here, the trusted amalgamation Am takes the input x and the second part of the key r applies the XOR operation to both and calls the underlying hash function.

Theorem 18. *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 specification $(\mathsf{Am}, \mathsf{Gen}', \mathsf{H}')$ with the family of hash functions $\mathcal{H}' = (\mathsf{Gen}', \mathsf{H}')$, with $\mathsf{H}' : \{0,1\}^\lambda \to \{0,1\}^{\lambda-1}$ as described above, is subversion-resilient target-collision-resistant in the offline watchdog model with trusted amalgamation, assuming the XOR operation and randomness generation 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 [RTYZ17], 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 queries 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

[2]Unfortunately, we could not find an explicit reference for this construction.

$\mathcal{A}$ 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 that $(s, y) \notin \mathcal{T}$. 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 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)[3].

The other case is $(s, y) \in \mathcal{T}$. But, as we 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 that 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 need a hash family $\mathcal{H}_\ell \colon \{0,1\}^\ell \to \{0,1\}^{\ell-1}$ for each $\ell = 2\lambda, 2\lambda - 1, \ldots, \lambda + 1$. Thus, we can combine polynomially many hash functions in an iterative process, to obtain a hash function that maps inputs to outputs half the input size.

13.8 Constructing Subversion-Resilient Signatures

Finally, we have all the ingredients to prove the signature scheme based on the Naor-Yung construction [NY89] subversion-resilient. As a necessary stepping stone, we 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.

13.8.1 Digital Signatures

Digital signatures are used to verify the authenticity and integrity of messages. A secure digital signature ensures that a message has not been altered during

[3]This resembles the "classical" security proof of the construction.

transmission and was indeed created by a specific sender. For this, it uses public-key cryptography. A sender has a secret singing key and a public verification key. As the names suggest, the signing key is used to sign messages, while everybody can use the verification key to verify signatures. We continue by recalling the standard definition of digital signatures.

Definition 26. A digital signature scheme $\Sigma = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ consists of three ppt algorithms, with each algorithm defined as follows:

- On input 1^λ the randomized *key generation algorithm* KGen outputs a pair of keys $(\mathsf{sk}, \mathsf{vk})$.
- The randomized *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 determinstic *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 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.

Note that here, we explicitly do not require perfect correctness as we did for the construction of subversion-resilient MACs and symmetric encryption presented in Chapter 12. As we discuss in Chapter 14, this is because our construction can not achieve perfect correctness and we suspect this notion can not be achieved. For a short further discussion on correctness, consider Section 13.9.

Next, we recall the standard definition of existential unforgeability for digital signatures. After generating a key pair, the adversary is given the verification key and access to a sign oracle. Thus, the adversary can ask for signatures on messages of its choice. The adversary wins if it can output a message signature pair (m, σ), such that σ is a valid signature for m. In order to rule out trivial attacks, the adversary does not win if it previously issued a query for m to its oracle.

Definition 27. Let $\Sigma = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ be a signature scheme and let $\mathsf{Exp}^{\mathsf{EUF}}_{\mathcal{A},\Sigma}$ be defined as shown in Fig. 13.4 where $\mathcal{A}$ 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. We define

$$\mathsf{Adv}^{\mathsf{EUF}}_{\mathcal{A},\Sigma}(1^\lambda) := \left|\Pr[\mathsf{Exp}^{\mathsf{EUF}}_{\mathcal{A},\Sigma}(1^\lambda) = 1]\right|$$

and say that Σ is *existentially unforgeable* if $\mathsf{Adv}^{\mathsf{EUF}}_{\mathcal{A},\Sigma}(1^\lambda)$ is negligible for all ppt adversaries $\mathcal{A}$.

$\mathsf{Exp}^{\mathsf{EUF}}_{\mathcal{A},\Sigma}(1^\lambda)$

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

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

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

  return 1

return 0

Figure 13.4: Unforgeability experiment for digital signatures.

A weaker form of security is *one-time* security. While existentially unforgeable signatures allow for multiple sign queries, one-time signatures, as the name suggests, only allow a single sign query to be issued by the adversary.

Definition 28. We say a signature is a *one-time* signature if Definition 27 holds *and* the attacker is only allowed to issue a *single* query to its signing oracle.

While this severely limits the adversary's capabilities, it allows the construction of one-time signatures from one-way functions, as discussed in the next section.

13.8.2 Lamport Signatures

Using the results of Section 13.6, we have access to subversion-resilient one-way functions and can directly obtain classical Lamport signatures [Lam79] 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 $(\mathsf{KGen}, \mathsf{Eval})$ as its main building block.

The key generation algorithm chooses ℓ many values $x_{i,0}, x_{i,1} \in \{0,1\}^\lambda$ uniformly at random as well as $\mathsf{pp} = \mathsf{KGen}(1^\lambda)$. Then compute $y_{i,0} = \mathsf{Eval}(\mathsf{pp}, x_{i,0})$ and $y_{i,1} = \mathsf{Eval}(\mathsf{pp}, x_{i,1})$. 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 $\mathsf{Eval}(\mathsf{pp}, x_i) = y_{i,M_i}$ 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 scheme'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. Similar to our results for subversion-resilient MACs, a trusted comparison seems unavoidable to obtain subversion-resilient signatures under complete subversion. 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 19. *Let* Π *be a subversion-resilient secure one-way function. Then Lamport signatures using* Π *as the one-way function are subversion-resilient existentially unforgeable one-time signatures in the offline watchdog model under with trusted amalgamation where the "==" operations is part of the trusted amalgamation .*

Our proof closely follows the classical proof for the security of Lamport signatures, as, for example, found in [KL14].

Proof. Following Russell et al. [RTYZ17], we argue that either the watchdog detects subversion or $(\mathsf{sk}, \mathsf{vk})$ is computed according to the specification. This is because the watchdog can test the underlying KGen algorithm and together with uniformly random bits, test F on a public input distribution.

Given an adversary $\mathcal{A}$, which breaks the subversion-resilient existential unforgeability of Lamport Signatures with some non-negligible probability ε, we can then construct an adversary $\mathcal{B}$ which breaks the subversion-resilience of the underlying one-way function is defined as follows: It chooses $i^* \xleftarrow{\$} \{0,1\}^\ell$ and $b^* \xleftarrow{\$} \{0,1\}$. Given y as a challenge, $\mathcal{B}$ then sets $y_{i^*,b^*} := y$. $\mathcal{B}$ then continues by choosing $x_{i,b} \xleftarrow{\$} \{0,1\}^n$ for all $i \in [\ell]$ and $b \in \{0,1\}$ and setting $y_{i,b} := \widetilde{\mathsf{Eval}}(\mathsf{pp}, x_{i,b})$. Together with $y_{i,b}$, these values form the public key pk, which is given to $\mathcal{A}$. Note that due to Lemma 1, all $y_{i,b}$ values follow the specification but with negligible probability due to the testing of the watchdog. $\mathcal{A}$ requests a forgery on some message M'. If $M'_{i^*} = b^*$ then $\mathcal{B}$ aborts, and returns the correct signature $\sigma = (x_{\ell,M'_1}, \ldots x_{\ell,M'_\ell})$ otherwise. Finally, $\mathcal{A}$ outputs a forgery (M, σ) with $\sigma = (x_1, \ldots, x_\ell)$. If $\mathcal{A}$ outputs a forgery at (i^*, b^*), then $\mathcal{B}$ outputs x_{i^*}. Thus, $\mathcal{B}$ guesses at which index $\mathcal{A}$ makes it forgery and for which message bit. The probability for guessing both correct is at least $\varepsilon/2\ell$, where ε is the success probability of $\mathcal{A}$ because $\mathcal{B}$ made its guesses independently before $\mathcal{A}$ attempts its forgery or message query. Thus, $\mathcal{B}$ successfully breaks the subversion-resilience of the above construction, as ε is non-negligible by the assumption of this proof. □

13.8.3 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, where each node is associated with a key pair of a one-time signature scheme. Messages to be signed are then interpreted as a path in this tree. The signing key is used for each node to sign the keys of the two child nodes. Since the Lamport signature can not sign messages bigger than its public key, a hash function allows the signing of two verification

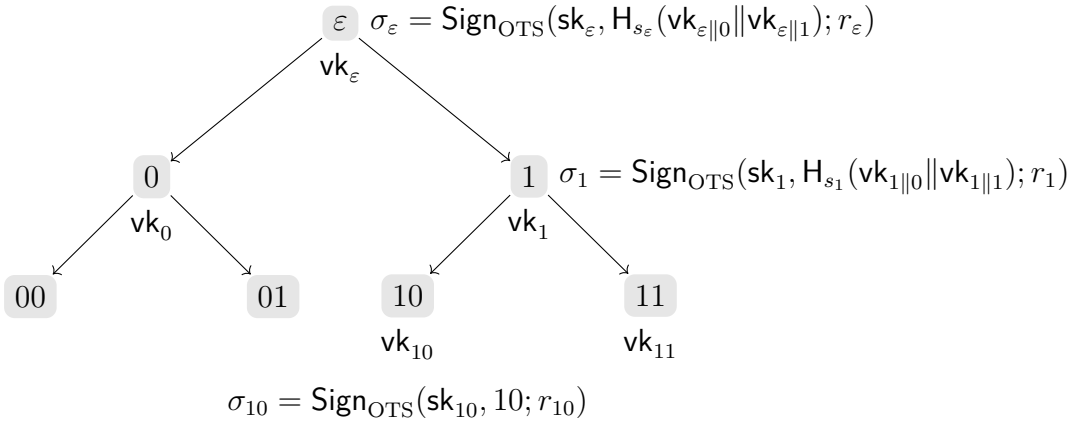


Figure 13.5: Simplified illustration of the Naor-Yung construction of digital signatures from one-times signatures for signing the message $M = 10$ where the derivation of randomness via PRFs is ignored.

keys. Here, a target-collision-resistant hash function is sufficient to guarantee security. While the original construction is stateful to keep track of which keys were used for the one-time signature scheme, it is known that it can be extended via PRFs and deterministically recomputation of keys to make the construction stateless. Note that the PRFs are only needed to sign messages and not for signature verification. A simplified example of the approach, not considering the PRFs, is shown in Fig. 13.5. 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 $\mathsf{H} = \{\mathsf{H}_s \colon \{0,1\}^{2\lambda} \to \{0,1\}^\lambda\}$, [4] 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.[5] 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 depicted in Fig. 13.6.

Thus, our amalgamation function here only follows the amalgamation functions of the underlying primitives while not introducing additional trusted operations besides handing in and outputs from one primitive to another.

[4] By applying the aforementioned iterative process for each value between 2λ and λ

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

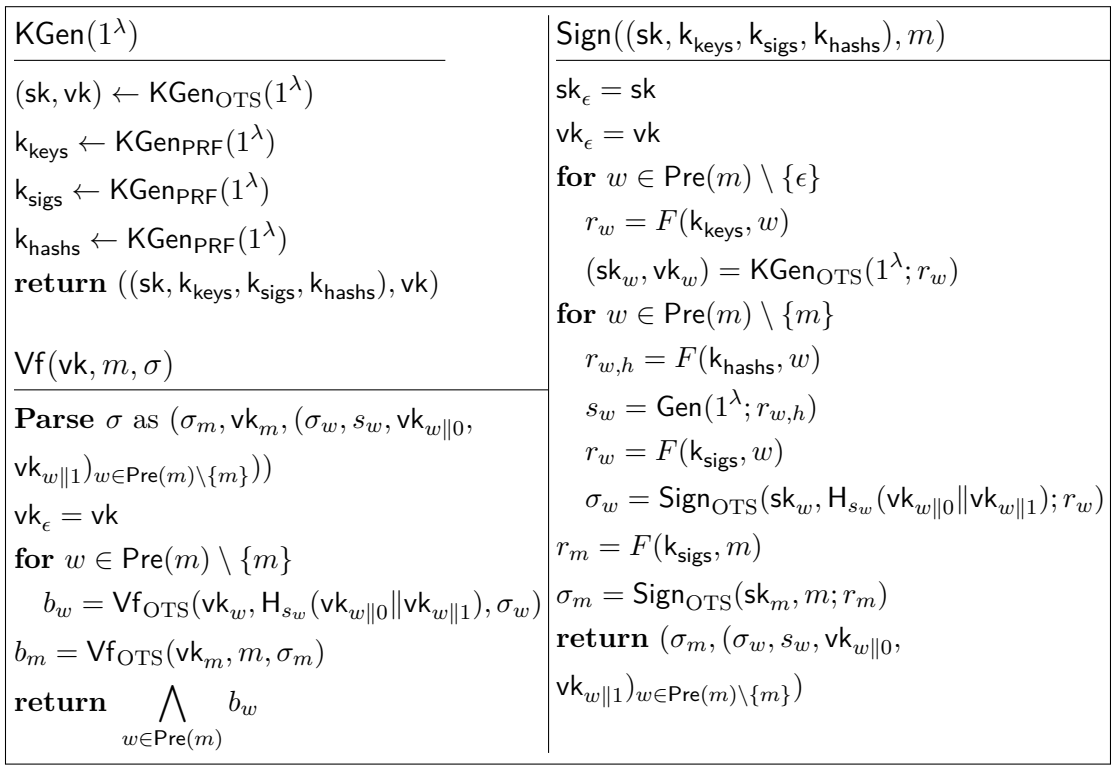


Figure 13.6: Our proposed signatures scheme.

Theorem 20. *Given a subversion-resilient one-time signature scheme* $\widehat{\Sigma} = (\mathsf{Am}_\Sigma, \mathsf{KGen}_{\mathrm{OTS}}, \mathsf{Sign}_{\mathrm{OTS}}, \mathsf{Vf}_{\mathrm{OTS}})$ *a subversion-resilient target-collision-resistant hash function family* $\widehat{\mathcal{H}} = (\mathsf{Am}_{\mathcal{H}}, \mathsf{Gen}, \mathsf{H})$*, and a subversion-resilient PRF* $\widehat{F} = (\mathsf{Am}_F, \mathsf{KGen}_{\mathsf{PRF}}, F)$*, then the specification* $(\mathsf{Am}, \widehat{\Sigma}, \widehat{\mathcal{H}}, \widehat{F})$ *where the amalgamation build the signature scheme as displayed in Fig. 13.6 is a stateless, subversion-resilient existentially unforgeable digital signature scheme in the offline watchdog with trusted amalgamation where* all *algorithms are subject to subversion, assuming randomness generation is part of the trusted amalgamation.*

In the following proof, we follow the proof by Naor and Yung [NY89], but we 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 must ensure that we reduce the security to the 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 all building blocks, i.e., of the one-time signature, the watchdog of the hash function, and the watchdog of the PRF. Note that subversion-resilience of key generation either follows from Lemma 1, or in the case of PRFs due to the access of uniformly random bits, which can directly be used as keys. If none

of these watchdogs detect a subversion, we follow an adaption of the proof by Naor and Yung [NY89].

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. Suppose this would be distinguishable from the setting where the PRF is used. In that case, we can easily build an attacker against the subversion-resilience of the PRF by simulating all other parts of the construction. We also ignore cases where some randomly chosen values (random strings or keys) collide, as this only happens with negligible probability.

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 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. □

13.9 Discussion

Efficiency. To provide a better evaluation of our results, we present an overview of subversion-resilient signature constructions from literature in Table 13.1. The table highlits 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 13.1: 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.

Construction	Model	RO	Complete Subv.	Signature size	Stateful subv.
[AMV15]	RF	✗	✗	σ	✓
[FM18]	SG	✗	✗	$\approx \lambda \cdot m + 2\lambda\sigma$	✗
[RTYZ16]	online WD	✓	✓	m	✓
[CRT+19]	offline WD	✓	✓	m	✗
[CRT+19]	offline WD	✗	✗	$2(m + \sigma)$	✗
This Thesis	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 discuss the construction of subversion-resilient 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 (see Section 12.5) and one-way permutations (see Section 13.6), 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[6] 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 teaching books for modern cryptography, such as [KL14], 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.

Subversion-Resilient Hash Functions via Combiners. One may wonder whether one can use a similar approach proposed in Part I and use cryptographic combiners to obtain a subversion-resilient hash function. However, this approach also seems futile. This is because even if we can guarantee that at least one instance (or even more instances) does not deviate from the specification, it seems hard to argue how this can be used to say that finding collisions is hard without assuming some pseudorandomness of hash values. This is which is why random oracles are helpful in this regard, as they fulfill this requirement.

Randomized Subversion. Following previous works, we assume that the subverted implementation of deterministic algorithms is also deterministic. Note that our proof for subversion-resilient one-way function can be adapted to allow the implementation of the one-way permutation to be potentially randomized. This is possible because the subverted algorithm is also executed once during the final check of the security game. In security games, where the subverted algorithm is executed multiple times, a careful analysis is required to determine whether this can also be allowed. We refrained from investigating this further as we focused on our main contributions, leaving this an exciting direction for future research.

Correctness. Note that both of our signature construction satisfies our correctness definition, even under subversion, but not perfect correctness. This is because after 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 Section 12.7). We suspect that (polynomial time) offline watchdogs can not achieve perfect correctness, as even a single trigger is sufficient to violate perfect correctness. To the best of our knowledge, there

[6] As it is the case for tree-based signatures

are no constructions available that could make similar use of a trusted XOR as in Section 12.7 or other operations that allow to cancel terms out during verification.

Part III

Conclusion

We conclude this thesis by discussing our results and pointing out open problems for further research.

14 Discussion

Our results indicate that achieving stronger notions of subversion-resilience comes with the price of decreased efficiency, as in bigger keys, ciphertexts, or signatures. While this seems like a big downside, we must remind ourselves that we are dealing with a security model with arguably one of the most powerful adversaries in the context of cryptography. Achieving security at all against an adversary that is in control over the whole implementation is a big challenge. Hopefully, after showing the general feasibility, our results can enable future research to improve our schemes and develop schemes with better efficiency or fewer assumptions. Thus, we now discuss our results further and propose directions for future research before concluding this thesis.

Outline. In Section 14.1 we continue by comparing the offline watchdog model to other models found in the literature. Afterwards in Section 14.2, we discuss our results of the previous chapters with regard to their limitations, instantiations, and alternative approaches. Further, we discuss open problems for future research in Section 14.3. We propose approaches for further improving our results, unifying existing techniques and approaches, and directions for completely new approaches. Finally, we briefly summarize and conclude this thesis in Section 14.4.

14.1 Offline Watchdogs vs. Alternative Models

As mentioned before, all our constructions are proven secure in an offline watchdog model. Since the different models are based on completely different assumptions, we emphasize that neither model is strictly *better* than the other. We firmly believe that being aware of all these aspects and having various available tools can enable us to make better-informed decisions about which approach fits an application best.

Reverse Firewalls. If we compare watchdogs to reverse firewalls, we notice both models have advantages and disadvantages. *Usually*, RFs can handle public-key primitives well, while watchdogs handle symmetric primitives well. This does not mean that they cannot handle the other types of primitives. However, as demonstrated by various works (including this one), additional assumptions are usually required, or an efficiency loss occurs. In our model, we assume a trusted watchdog has fine-grained access to the used building blocks, and we require a trusted amalgamation. RFs on the other hand require that the reverse firewall itself is not subverted and potentially requires the RF to

self-destruct for input it cannot process. Then again, reverse firewalls can easily handle stateful subversion, while offline watchdogs need to depend on the adversary to defend against this type of attacks. Depending on the application, it is crucial to carefully evaluate which assumption can be better justified.

Self-Guarding Algorithms. Self-guarding algorithms [FM18] use a sampling phase, where the algorithm still computes outputs according to the specification. To employ this in practice, one would need to ensure that such a phase can indeed be guaranteed. We suspect that this would potentially be viable if somehow some guarded environment can be created to collect these backdoor-free samples. Self-guarding algorithms are similar to the watchdog model in the sense that both assume that some operations are exempt from subversion. In our model, these are all operations within the trusted amalgamation. For self-guarding algorithms, on the other hand, these are *all* algorithms, although only during the trusted sample phase. However, for some primitives, such as public-key encryption, the number of sanitized outputs is limited by the number of collected samples, which reduces the practicality of the self-guarding algorithms.

Immunization. In the immunization model of Dodis et al. [DGG+15] some immunization function that is exempt from subversion is applied to the output of pseudorandom generators. Thus, they use the leverage that the construction can rely on the act that the immunization function indeed behaves as specified. Thus, they differentiate to which degree the adversary is aware of the immunization function. In contrast, our approach aims to guarantee that the behavior of some implementations is in accordance with a specification without applying operations after the implementation produces an output.

14.2 Our Results

Limitations of Our Approach. While advancing the state-of-the-art in subversion-resilient cryptography, our results come with a few limitations. All our results are in the trusted amalgamation model, which requires more fine-grained testing access beyond black-box access. Additionally, we make use of several trusted operations, although these seem necessary in the presence of generic attacks. Also, we can not defend against stateful subversion attacks as we only consider universal watchdogs which are independent of the adversary. That being said, all these assumptions seem necessary to achieve our obtained security guarantees. If we remove the trusted amalgamation model or the trusted operations, generic attacks such as input triggers on the receiving algorithm are possible. Thus, we aimed to minimize the necessary trust assumptions as best as possible while avoiding using "cryptographic operations", such as hashing, as a trusted operation.

Concrete Instantiations and Minimal Assumptions. In both parts of this thesis, we start from generic building blocks (one-way KEMs, weak PRFs, one-wy permutations) and build more complex primitives with them as a foundation. This allows us to plug in any concrete instantiation with the required security guarantees. The way our watchdogs are defined, the exact details of the used scheme do not matter for testing as long as the input and output behavior is specified. So, the natural question is whether watchdogs for specific schemes would, for example, allow for schemes with fewer assumptions. However, the problem arises regarding to which degree operations can be usefully modeled as trusted operations. We argue that trusted operations, such as an XOR or a comparison, are "simple enough" such that it can be assured that they are not subverted. If one would consider specific schemes using, for example, cyclic groups, one could argue that one could apply the trusted amalgamation model and require such fine-grained access that the watchdog can even test group operations. If such an approach would lead to a secure scheme, this would most likely only grant very few meaningful results, as the adversary's power is severely limited. We believe that operations like an XOR or a comparison, which are "non-cryptographic", form a useful middle ground between requiring more fine-grained access than only black-box access and minimizing the required access as best as possible.

Code Verification vs Watchdogs. While some may suggest that verifying the software against specifications can resolve the issue of backdoors, it's worth noting that the complete code of some widespread schemes may not be accessible as they are closed-source. Also, obfuscating the code could also make this approach more challenging than it initially appears. Additionally, our watchdog model actually does not require access to a *complete* description of the in- and output behavior. It suffices to obtain information on the values sampled by the watchdog during its testing phase. This is captured by our models, as the watchdog is only given oracle access to the specification.

Stego-Freeness vs Subversion-Resilience. In the security proofs for both our constructions in Part II, we always prove the subversion resilience of a building block to then argue for the subversion resilience of the overall scheme. Russell et al. [RTYZ17] proposed the notion of stego-freeness. This notion describes a scenario where an adversary cannot distinguish between an algorithm's implementation and specification. The authors presented various versions of stego-freeness, differing on whether the adversary can choose the algorithm's inputs freely or if they're chosen based on a known distribution. Their approach to achieving subversion-resilient CPA secure encryption involves replacing all building blocks of an encryption scheme with their specification and then playing the security game with the amalgamated specification. Security is guaranteed by the security of the specification, and their analysis is based on their asymptotic framework and conditioned on the implementation passing the watchdog's testing. However, in our concrete security model of Part I, this

approach creates issues with security notions that involve indistinguishability. After the watchdog's tests, we lack a similar "baseline" to argue for security. Our construction allows the adversary to always distinguish between the implementation and specification in our model. Nevertheless, when only considering one-way security, we do not need a similar notion of stego-freeness in our model. For our analysis, the event that one instance follows the specification is sufficient. On the other hand, we want to mention that parts of our results from Part II could be reworked to utilize stego-freeness instead of subversion-resilience. Basically, the proofs would change because if one could show that certain building blocks could be replaced with their specification, the original security proofs hold without redoing the security proof with the subversion-resilient property. That being said, the above does not apply to all our used building blocks, for example, the decryption algorithm of the symmetric encryption scheme. Overall, proving our results via stego-freeness would introduce notational overhead to correctly model our use of subversion-resilience, as we aimed to show the subversion resilience of all building blocks anyway.

14.3 Open Problems

While this thesis proposes advancements in the field of subversion, many interesting open problems remain for the future. Here, we want to discuss some of these possible research directions.

Building on Our Results. A natural question is if it is possible to further improve our constructions. Improvement can, for example, mean achieving the same security notions while using fewer assumptions or improving the schemes' efficiency to match modern schemes' efficiency better. In case our work already proposes the minimal amount of trusted operations or the "least complex" trusted operations, the task would be to formally prove this, requiring adequate formalizations of the above statements.

We already observed in Part II, that we used a very similar approach and techniques for both our constructions of authenticate encryption and digital signatures. The question becomes whether we can achieve general theorems of the form: "If we take a construction for a cryptographic scheme and prove all building blocks subversion-resilient, then the resulting construction is also subversion-resilient". Similar results were shown by Chakraborty, Magri, Nielsen, and Venturi [CMNV22] in the UC (Universal Composability) model for constructions in the reverse firewall model. A theorem of the above form could be incredibly useful in proving TLS-like protocols subversion-resilient. Speaking of which, this thesis provides the necessary building blocks for a TLS-like protocol: KEMs for exchanging key material digital signatures and MACs for guaranteeing integrity and authenticity, as well as authenticated encryption for guaranteeing confidentiality. However, when considering the subversion-resilience of TLS, the question of how to handle states under subversion arises. This is important not

only when dealing with stateful subversion but also when dealing with notions like forward security, a standard security property of TLS. Roughly speaking, forward security means that given access to the current state of a party, past communication remains secure. Updating the secret key material in a "save" manner while deleting old key material can lead to forward security. A simple subversion attack could be that an adversary does not erase old key material. Additionally, as demonstrated by Berndt et al. [BWP+22], forward secrecy implies ASAs. One possible approach to prevent these attacks would be to limit the space available to the adversary. Thus, it can use the disk space reserved by the non-subverted scheme in any way it wishes but does not get access to save more data. This could be motivated by operating systems that prevent applications from arbitrarily allocating disk additional disk space. Thus, the adversary could store *different* keys then computed by the specification. This could, in turn, lead to detection, as the adversary might be unable to continue the protocol correctly. The detection model would thus need to formally incorporate this. Other possible approaches would be to adjust the state-leaking oracle so that the adversary cannot access the saved old key material. However, it is not clear how this could be achieved or motivated.

A similar question arises if we consider the subversion of modern instant messaging protocols. So far, prior work only considered subversion attacks on these protocols [BWP+22, CEJ23], and possible countermeasures against their specific attacks. However, to the best of our knowledge, no Instant Messaging protocol has yet proven to be subversion-resilient. It's an interesting and important question: Which properties of modern messaging protocols can be guaranteed under subversion? Can existing protocols achieve this property, can they be adapted to achieve them, or are entirely new protocols needed? We hope that the building blocks proposed in this thesis can help as a foundation for this line of research.

Complete Subversion with Practical Watchdogs. Our results of Part I do not require random oracles while also allowing the adversary to provide the implementation for all algorithms relevant to the security experiment (thus excluding decapsulation/decryption). Thus, one could potentially argue that also the results Part I achieve subversion-resilience under complete subversion without random oracles. However, as prior results by Russell et al. [RTYZ17] could also be interpreted this way, we focus only on the practical aspects of the watchdog.

Countermeasures against Stateful Subversion for Offline Watchdogs. More generally, finding new countermeasures against stateful subversion in a universal offline watchdog model would also be incredibly useful, even at the cost of some other, additional assumption. So far, only reverse firewalls and online watchdogs with access to the state can effectively defend against stateful subversion. Russell [RTYZ16] et al. proposed a way that offline watchdogs can also defend against stateful attacks if the runtime of the watchdog is allowed

to depend on the adversary. However, then one would consider non-universal watchdogs, which we aimed to avoid. Defending against stateful subversion with a universal offline watchdog whose running time is independent of the adversary seems impossible without further assumptions. This is because of the aforementioned time-bomb attacks. If the watchdog runs for time t, but the implementation of the adversary only diverts from the specification at time $t + 1$, this attack can not be detected. Thus, the question is how this simple but generic attack can be circumvented by additional assumptions, such as a regular state reset done in a trusted manner. Such an approach is known from the field of hardware trojans. Dziembowski, Faust, and Standaert [DFS16] guard deterministic functions by proposing a semi-online watchdog (which they did not refer to as such) that regularly tests tokens against the specification. It's interesting whether similar or completely different techniques could be used in a formal watchdog model to prove randomized schemes subversion-resilient.

Combining Watchdogs and Reverse Firewalls. A different approach to tackle the subversion-resilience of, for example, TLS-like protocols would be to combine a reverse firewall and a watchdog approach. The usual intuition is that reverse firewalls work well for public-key primitives that are rerandomizeable. At the same time, watchdogs generally fit a symmetric and non-rerandomizeable scheme better ("well" and "better" referring to either fewer assumptions or better efficiency). Thus, instead of forcing a complete protocol to use only one of the two, it would seem natural to combine the approaches. The reverse firewall handles the key exchange/agreement while the watchdog handles the primitives utilizing these keys. Again, generic results similar to UC-style results would be preferable, with an appropriate model combining reverse firewalls and watchdogs. Ultimately, comparing this approach to constructions using solely a watchdog or reverse firewall approach is an interesting direction for future research.

Combiners. In Part I, we saw that cryptographic combiners can achieve subversion-resilient one-way secure KEMs. Thus, the question becomes whether a similar approach can be used for other primitives with security properties modeled as search problems. Obvious candidates are the primitives considered in Part II. It is known from previous works that combiners exist for one-way functions as presented by Harnik, Kilian, Naor, Reingold, and Rosen [HKN+05] and hash functions as proposed by Fischlin, Lehmann, and Pietrzak [FLP14], although both approaches double the output size of the primitive by simply concatenating the outputs of two instances. Another important question is whether repeated oracle access introduces further complications, as discussed in Chapter 9. Thus, it needs to be investigated whether similar results to our results of Part II can also be achieved in a concrete security setting with efficient watchdogs using the aforementioned combiners.

Post-Quantum. There are several works that outline attacks on post-quantum schemes [YCL+20, GG19, JHZ+23]. Yang, Chen, Li, Qu, and Yang [YCL+20]

mention in their work that existing approaches [MS15, RTYZ17, RTYZ16], can be used to prevent their attack. While our and prior work allows for current post-quantum schemes to be plugged in as a building block as we build upon generic building blocks, investigating whether existing schemes can already achieve subversion-resilience with fewer assumptions as compared to generic approaches is a possible direction for future work.

MPC. So far, several works [MS15, CGPS21, CDN20, CMNV22] proposed approaches multi-party computation (MPC) protocols with reverse firewalls exist. Investigating whether similar results can be achieved in a watchdog model would be interesting. For this, an adequate model is required that clearly defines the adversaries' capabilities. Some form of subversion is already built in some MPC models. This is because it makes little difference whether the adversary completely controls a party or that party uses a subverted implementation. Additionally, security against *covert security* as introduced by Aumann and Lindell [AL07] is already an established security notion. Covert security captures that honest parties are guaranteed to detect any misbehavior by a malicious party with a constant probability. Thus, the honest parties participating in such an MPC protocol could be viewed as online watchdogs. Investigating relations to online and offline watchdog models, as well as whether covert security directly translates to subversion resilience is an interesting open question.

More Parameters for Detection. In our models and constructions, the watchdogs compare the input and output behavior of the specification and the implementation. Although we require fine-grained access to the used building block to achieve strong security notions, we do not need more intrusive access, as, for example, direct access to the code. Note, however, that other parameters can be considered when modeling a watchdog. Since the watchdog can access a building block's input and outputs, it might not be far-fetched that other information is also available. This could, for example, include the runtime of an algorithm or the power consumption of an implementation. Taking the example of the von Neumann extractor or subversion attacks via rejection sampling, the runtime of the subverted algorithm might be much higher than the runtime of the specification. In an offline watchdog model, this information might be easy to obtain. On the other hand, an online watchdog that requires constant monitoring of the runtime of algorithms and power consumption is much more intrusive. Fishlin and Mazaheri [FM18] used the runtime of subverted algorithms to argue that weak PRFs can not be subverted without being detected. However, their security model does not formally capture this. Thus, it is an exciting problem to formally model these additional parameters, give the watchdog access to this new information, and investigate whether this allows simpler constructions with a stronger watchdog.

Moving to the Real World. While our research grants further theoretical insights, seeing how our constructions perform in the real world would be desirable. Especially how our assumptions on trusted operations could, for instance,

be realized in software or hardware. Overall, our constructions are much less efficient than current state-of-the-art constructions. The question is whether this loss is acceptable for certain applications. For instance, we would suspect that investigative journalists or whistleblowers would be willing to accept if their communication takes longer but can be certain that their means of communication is subversion-resilient.

14.4 Conclusion

In this thesis, we identify key aspects that we consider critical for subversion-resilient schemes to be practical. In Part I we focus on efficient testing in offline watchdog models. We propose a security model in a concrete security setting. Within this model, we construct a subversion-resilient randomness generator with constant testing time and a subversion-resilient one-way public-key encryption scheme with linear testing time. As our primary technique, we use combiners for KEMs to construct subversion-resilient one-way KEMs using a trusted XOR, which also allows the construction of subversion-resilient one-way public-key encryption. Thus, we propose the first cryptographic constructions with efficient watchdogs. We also showed that the claimed results on indistinguishability of a previous version of our results were impossible to achieve with our proposed approach. In Part II, we change the focus to designing subversion-resilient schemes, where all practically relevant algorithms are subject to subversion while not using idealized primitives as building blocks. Due to the limitations of our model in Part I, we use an asymptotic security setting and focus on constructing subversion-resilient schemes under complete subversion without random oracles. Ultimately, we propose the first construction of authenticated encryption where implementations for both the encryption and decryption algorithm are provided by the adversary and digital signatures where all algorithms are subject to subversion, both without random oracles. Our constructions follow the same approach, building the primitives from smaller, symmetric primitives. We observe weak PRFs and one-way-permutation are naturally subversion-resilient (with one-way-permutation granting subversion-resilient one-way functions) in our used models. This allows us to revisit classical results from cryptography in the context of subversion. In detail, we see that the classical Encrypt-then-MAC approach grants subversion-resilient authenticated encryption and Naor-Yung signatures can be constructed in a subversion-resilient manner. Both times, we prove various other important building blocks such as symmetric encryption, MACs, and target-collision-resistant hash functions subversion-resilient along the way. All our results are in the trusted amalgamation and split-program model and only require a few simple operations, such as an XOR, trusted comparison, or a trusted data structure, but do not rely on random oracles. This thesis makes steps toward more practical subversion-resilient schemes without random oracles.

While we deem it unlikely that our proposed construction will be rolled out on a grander scale due to the increased key and output size, we hope that our contributions can help enable future research to design *truly* practical subversion-resilient schemes that find widespread use in practice.

Bibliography

[ABK+07] Dakshi Agrawal, Selçuk Baktir, Deniz Karakoyunlu, Pankaj Rohatgi, and Berk Sunar. Trojan detection using IC fingerprinting. In *2007 IEEE Symposium on Security and Privacy*, pages 296–310, Oakland, CA, USA, May 20–23, 2007. IEEE Computer Society Press. doi:10.1109/SP.2007.36.

[AFMV19] Giuseppe Ateniese, Danilo Francati, Bernardo Magri, and Daniele Venturi. Public immunization against complete subversion without random oracles. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *ACNS 19*, volume 11464 of *LNCS*, pages 465–485, Bogota, Colombia, June 5–7, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-21568-2_23.

[AL07] Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 137–156, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-70936-7_8.

[ALSZ21] Behzad Abdolmaleki, Helger Lipmaa, Janno Siim, and Michal Zajac. On subversion-resistant SNARKs. *Journal of Cryptology*, 34(3):17, July 2021. doi:10.1007/s00145-021-09379-y.

[AMV15] Giuseppe Ateniese, Bernardo Magri, and Daniele Venturi. Subversion-resilient signature schemes. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 364–375, Denver, CO, USA, October 12–16, 2015. ACM Press. doi:10.1145/2810103.2813635.

[AP19] Marcel Armour and Bertram Poettering. Subverting decryption in AEAD. In Martin Albrecht, editor, *17th IMA International Conference on Cryptography and Coding*, volume 11929 of *LNCS*, pages 22–41, Oxford, UK, December 16–18, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-35199-1_2.

[AP22] Marcel Armour and Bertram Poettering. Algorithm substitution attacks against receivers. *Int. J. Inf. Sec.*, 21(5):1027–1050, 2022.

[AS21] Behzad Abdolmaleki and Daniel Slamanig. Subversion-resistant quasi-adaptive NIZK and applications to modular zk-SNARKs. In Mauro Conti, Marc Stevens, and Stephan Krenn, editors, *CANS 21*, volume 13099 of *LNCS*, pages 492–512, Vienna, Austria, December 13–15, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-92548-2_26.

[Bag19] Karim Baghery. Subversion-resistant simulation (knowledge) sound nizks. In Martin Albrecht, editor, *Cryptography and Coding - 17th IMA International Conference, IMACC 2019, Oxford, UK, December 16-18, 2019, Proceedings*, volume 11929 of *Lecture Notes in Computer Science*, pages 42–63. Springer, 2019. doi:10.1007/978-3-030-35199-1_3.

[Bag20] Karim Baghery. Subversion-resistant commitment schemes: Definitions and constructions. In Konstantinos Markantonakis and Marinella Petrocchi, editors, *Security and Trust Management - 16th International Workshop, STM 2020, Guildford, UK, September 17-18, 2020, Proceedings*, volume 12386 of *Lecture Notes in Computer Science*, pages 106–122. Springer, 2020. doi:10.1007/978-3-030-59817-4_7.

[BBB+17] Pascal Bemmann, Felix Biermeier, Jan Bürmann, Arne Kemper, Till Knollmann, Steffen Knorr, Nils Kothe, Alexander Mäcker, Manuel Malatyali, Friedhelm Meyer auf der Heide, Sören Riechers, Johannes Schaefer, and Jannik Sundermeier. Monitoring of domain-related problems in distributed data streams. In *Structural Information and Communication Complexity - 24th International Colloquium, SIROCCO 2017, Porquerolles, France, June 19-22, 2017, Revised Selected Papers*, volume 10641 of *Lecture Notes in Computer Science*, pages 212–226. Springer, 2017.

[BBC24] Pascal Bemmann, Sebastian Berndt, and Rongmao Chen. Subversion-resilient signatures without random oracles. In *Applied Cryptography and Network Security - 22nd International Conference, ACNS 2024, Abu Dhabi, United Arab Emirates*, 2024. To appear..

[BBCJ23] Pascal Bemmann, Sebastian Berndt, Rongmao Chen, and Tibor Jager. Subversion-resilient public key encryption with practical watchdogs. Cryptology ePrint Archive, Paper 2021/230, 2023. URL: https://eprint.iacr.org/archive/2021/230/20231011:061958.

[BBD+23] Pascal Bemmann, Sebastian Berndt, Denis Diemert, Thomas Eisenbarth, and Tibor Jager. Subversion-resilient authenticated encryp-

tion without random oracles. In *Applied Cryptography and Network Security - 21st International Conference, ACNS 2023, Kyoto, Japan, June 19-22, 2023, Proceedings, Part II*, pages 460–483, 2023. doi:10.1007/978-3-031-33491-7_17.

[BBF+20] Angèle Bossuat, Xavier Bultel, Pierre-Alain Fouque, Cristina Onete, and Thyla van der Merwe. Designing reverse firewalls for the real world. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *ESORICS 2020, Part I*, volume 12308 of *LNCS*, pages 193–213, Guildford, UK, September 14–18, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-58951-6_10.

[BC10] Bazara I. A. Barry and H. Anthony Chan. Intrusion detection systems. In Peter P. Stavroulakis and Mark Stamp, editors, *Handbook of Information and Communication Security*, pages 193–205. Springer, 2010. doi:10.1007/978-3-642-04117-4_10.

[BCJ21] Pascal Bemmann, Rongmao Chen, and Tibor Jager. Subversion-resilient public key encryption with practical watchdogs. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 627–658, Virtual Event, May 10–13, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-75245-3_23.

[BDJR97] Mihir Bellare, Anand Desai, Eric Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *38th FOCS*, pages 394–403, Miami Beach, Florida, October 19–22, 1997. IEEE Computer Society Press. doi:10.1109/SFCS.1997.646128.

[BFS16] Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. NIZKs with an untrusted CRS: Security in the face of parameter subversion. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 777–804, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-53890-6_26.

[BG13] James Ball Julian Borger and Glenn Greenwald. Revealed: how us and uk spy agencies defeat internet privacy and security, 2013. URL: https://www.theguardian.com/world/2013/sep/05/nsa-gchq-encryption-codes-security [cited 29.10.2023].

[BJK15] Mihir Bellare, Joseph Jaeger, and Daniel Kane. Mass-surveillance without the state: Strongly undetectable algorithm-substitution attacks. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 1431–1440, Denver, CO, USA, October 12–16, 2015. ACM Press. doi:10.1145/2810103.2813681.

[BL17] Sebastian Berndt and Maciej Liskiewicz. Algorithm substitution attacks from a steganographic perspective. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1649–1660, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press. doi:10.1145/3133956.3133981.

[BLR90] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. In *22nd ACM STOC*, pages 73–83, Baltimore, MD, USA, May 14–16, 1990. ACM Press. doi:10.1145/100216.100225.

[BN00] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 531–545, Kyoto, Japan, December 3–7, 2000. Springer, Heidelberg, Germany. doi:10.1007/3-540-44448-3_41.

[BN08] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *Journal of Cryptology*, 21(4):469–491, October 2008. doi:10.1007/s00145-008-9026-x.

[Boo47] George Boole. *The mathematical analysis of logic*. Philosophical Library, 1847.

[BPR14] Mihir Bellare, Kenneth G. Paterson, and Phillip Rogaway. Security of symmetric encryption against mass surveillance. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 1–19, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-44371-2_1.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press. doi:10.1145/168588.168596.

[BR17] Andrej Bogdanov and Alon Rosen. Pseudorandom functions: Three decades later. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography*, pages 79–158. Springer International Publishing, 2017. doi:10.1007/978-3-319-57048-8_3.

[BWP+22] Sebastian Berndt, Jan Wichelmann, Claudius Pott, Tim-Henrik Traving, and Thomas Eisenbarth. ASAP: Algorithm substitution

attacks on cryptographic protocols. In Yuji Suga, Kouichi Sakurai, Xuhua Ding, and Kazue Sako, editors, *ASIACCS 22*, pages 712–726, Nagasaki, Japan, May 30 – June 3, 2022. ACM Press. doi:10.1145/3488932.3517387.

[CDN20] Suvradip Chakraborty, Stefan Dziembowski, and Jesper Buus Nielsen. Reverse firewalls for actively secure MPCs. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 732–762, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-56880-1_26.

[CEJ23] Benoît Cogliati, Jordan Ethan, and Ashwin Jha. Subverting telegram's end-to-end encryption. *IACR Trans. Symmetric Cryptol.*, 2023(1):5–40, 2023. URL: https://doi.org/10.46586/tosc.v2023.i1.5-40, doi:10.46586/TOSC.V2023.I1.5-40.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004. doi:10.1145/1008731.1008734.

[CGPS21] Suvradip Chakraborty, Chaya Ganesh, Mahak Pancholi, and Pratik Sarkar. Reverse firewalls for adaptively secure MPC without setup. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part II*, volume 13091 of *LNCS*, pages 335–364, Singapore, December 6–10, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-92075-3_12.

[CHY20] Rongmao Chen, Xinyi Huang, and Moti Yung. Subvert KEM to break DEM: Practical algorithm-substitution attacks on public-key encryption. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 98–128, Daejeon, South Korea, December 7–11, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-64834-3_4.

[CMG+16] Stephen Checkoway, Jacob Maskiewicz, Christina Garman, Joshua Fried, Shaanan Cohney, Matthew Green, Nadia Heninger, Ralf-Philipp Weinmann, Eric Rescorla, and Hovav Shacham. A systematic analysis of the juniper dual EC incident. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 468–479, Vienna, Austria, October 24–28, 2016. ACM Press. doi:10.1145/2976749.2978395.

[CMNV22] Suvradip Chakraborty, Bernardo Magri, Jesper Buus Nielsen, and Daniele Venturi. Universally composable subversion-resilient cryp-

tography. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 272–302, Trondheim, Norway, May 30 – June 3, 2022. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-06944-4_10.

[CMY+16] Rongmao Chen, Yi Mu, Guomin Yang, Willy Susilo, Fuchun Guo, and Mingwu Zhang. Cryptographic reverse firewall via malleable smooth projective hash functions. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 844–876, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-53887-6_31.

[CRT+19] Sherman S. M. Chow, Alexander Russell, Qiang Tang, Moti Yung, Yongjun Zhao, and Hong-Sheng Zhou. Let a non-barking watchdog bite: Cliptographic signatures with an offline watchdog. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part I*, volume 11442 of *LNCS*, pages 221–251, Beijing, China, April 14–17, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-17253-4_8.

[DCM+19] Emma Dauterman, Henry Corrigan-Gibbs, David Mazières, Dan Boneh, and Dominic Rizzo. True2f: Backdoor-resistant authentication tokens. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 398–416. IEEE, 2019. doi:10.1109/SP.2019.00048.

[DFP15] Jean Paul Degabriele, Pooya Farshim, and Bertram Poettering. A more cautious approach to security against mass surveillance. In Gregor Leander, editor, *FSE 2015*, volume 9054 of *LNCS*, pages 579–598, Istanbul, Turkey, March 8–11, 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-48116-5_28.

[DFS16] Stefan Dziembowski, Sebastian Faust, and François-Xavier Standaert. Private circuits III: Hardware trojan-resilience via testing amplification. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 142–153, Vienna, Austria, October 24–28, 2016. ACM Press. doi:10.1145/2976749.2978419.

[DGG+15] Yevgeniy Dodis, Chaya Ganesh, Alexander Golovnev, Ari Juels, and Thomas Ristenpart. A formal treatment of backdoored pseudorandom generators. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 101–126, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-46800-5_5.

[DIJK09] Yevgeniy Dodis, Russell Impagliazzo, Ragesh Jaiswal, and Valentine Kabanets. Security amplification for interactive cryptographic primitives. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 128–145. Springer, Heidelberg, Germany, March 15–17, 2009. doi:10.1007/978-3-642-00457-5_9.

[DMS16] Yevgeniy Dodis, Ilya Mironov, and Noah Stephens-Davidowitz. Message transmission with reverse firewalls—secure communication on corrupted machines. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 341–372, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-53018-4_13.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18, Santa Barbara, CA, USA, August 19–23, 1984. Springer, Heidelberg, Germany.

[FJM18] Marc Fischlin, Christian Janson, and Sogol Mazaheri. Backdoored hash functions: Immunizing HMAC and HKDF. In Steve Chong and Stephanie Delaune, editors, *CSF 2018 Computer Security Foundations Symposium*, pages 105–118, Oxford, UK, July 9–12, 2018. IEEE Computer Society Press. doi:10.1109/CSF.2018.00015.

[FLP14] Marc Fischlin, Anja Lehmann, and Krzysztof Pietrzak. Robust multi-property combiners for hash functions. *Journal of Cryptology*, 27(3):397–428, July 2014. doi:10.1007/s00145-013-9148-7.

[FM18] Marc Fischlin and Sogol Mazaheri. Self-guarding cryptographic protocols against algorithm substitution attacks. In Steve Chong and Stephanie Delaune, editors, *CSF 2018 Computer Security Foundations Symposium*, pages 76–90, Oxford, UK, July 9–12, 2018. IEEE Computer Society Press. doi:10.1109/CSF.2018.00013.

[Fuc18] Georg Fuchsbauer. Subversion-zero-knowledge SNARKs. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 315–347, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-76578-5_11.

[Gel] Barton Gellman. Edward snowden, after months of nsa revelations, says his mission's accomplished. URL: https://www.washingtonpost.com/world/national-security/edward-snowden-after-

months-of-nsa-revelations-says-his-missions-accomplished/2013/12/23/49fc36de-6c1c-11e3-a523-fe73f0ff6b8d_story.html [cited 15.10.2023].

[GG19] Herman Galteland and Kristian Gjøsteen. Subliminal channels in post-quantum digital signature schemes. Cryptology ePrint Archive, Report 2019/574, 2019. https://eprint.iacr.org/2019/574.

[GGM84a] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th FOCS*, pages 464–479, Singer Island, Florida, October 24–26, 1984. IEEE Computer Society Press. doi:10.1109/SFCS.1984.715949.

[GGM84b] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 276–288, Santa Barbara, CA, USA, August 19–23, 1984. Springer, Heidelberg, Germany.

[GHP18] Federico Giacon, Felix Heuer, and Bertram Poettering. KEM combiners. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 190–218, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-76578-5_7.

[GMV20] Chaya Ganesh, Bernardo Magri, and Daniele Venturi. Cryptographic reverse firewalls for interactive proof systems. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 55:1–55:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. URL: https://doi.org/10.4230/LIPIcs.ICALP.2020.55, doi:10.4230/LIPICS.ICALP.2020.55.

[Gre] Glenn Greenwald. Nsa collecting phone records of millions of verizon customers daily. URL: https://www.theguardian.com/world/2013/jun/06/nsa-phone-records-verizon-court-order [cited 15.10.2023].

[HFK+10] Matthew Hicks, Murph Finnicum, Samuel T. King, Milo M. K. Martin, and Jonathan M. Smith. Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically. In *2010 IEEE Symposium on Security and Privacy*, pages

159–172, Berkeley/Oakland, CA, USA, May 16–19, 2010. IEEE Computer Society Press. doi:10.1109/SP.2010.18.

[HKN+05] Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. On robust combiners for oblivious transfer and other primitives. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 96–113, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany. doi:10.1007/11426639_6.

[HR05] Thomas Holenstein and Renato Renner. One-way secret-key agreement and applications to circuit polarization and immunization of public-key encryption. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 478–493, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Heidelberg, Germany. doi:10.1007/11535218_29.

[JHZ+23] Haodong Jiang, Jiang Han, Zhenfeng Zhang, Zhi Ma, and Hong Wang. Practical algorithm substitution attacks on real-world public-key cryptosystems. *IEEE Trans. Inf. Forensics Secur.*, 18:5069–5081, 2023. doi:10.1109/TIFS.2023.3304124.

[JKMS20] Aayush Jain, Alexis Korb, Nathan Manohar, and Amit Sahai. Amplifying the security of functional encryption, unconditionally. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 717–746, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-56784-2_24.

[JMS20] Aayush Jain, Nathan Manohar, and Amit Sahai. Combiners for functional encryption, unconditionally. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 141–168, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-45721-1_6.

[Kar] Paul Karp. Australia's war on encryption: the sweeping new powers rushed into law. URL: https://www.theguardian.com/technology/2018/dec/08/australias-war-on-encryption-the-sweeping-new-powers-rushed-into-law [cited 14.10.2023].

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014.

[Lam79] Leslie Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, October 1979.

[LCWW18] Chi Liu, Rongmao Chen, Yi Wang, and Yongjun Wang. Asymmetric subversion attacks on signature schemes. In Willy Susilo and Guomin Yang, editors, *ACISP 18*, volume 10946 of *LNCS*, pages 376–395, Wollongong, NSW, Australia, July 11–13, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-93638-3_22.

[Mer90] Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 218–238, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany. doi:10.1007/0-387-34805-0_21.

[Mila] Milieu. Study on the retention of electronic communications non-content data for law enforcement purposes. URL: https://www.statewatch.org/media/1453/eu-com-study-data-retention-10-20.pdf [cited 12.10.2023].

[Milb] Greg Miller. The intelligence coup of the century. URL: https://www.washingtonpost.com/graphics/2020/world/national-security/cia-crypto-encryption-machines-espionage/ [cited 12.10.2023].

[MS07] Ueli M. Maurer and Johan Sjödin. A fast and key-efficient reduction of chosen-ciphertext to known-plaintext security. In Moni Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 498–516, Barcelona, Spain, May 20–24, 2007. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-72540-4_29.

[MS15] Ilya Mironov and Noah Stephens-Davidowitz. Cryptographic reverse firewalls. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 657–686, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-46803-6_22.

[MT08] Ueli M. Maurer and Stefano Tessaro. Basing PRFs on constant-query weak PRFs: Minimizing assumptions for efficient symmetric cryptography. In Josef Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 161–178, Melbourne, Australia, December 7–11, 2008. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-89255-7_11.

[NR99] Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. *J. Comput. Syst. Sci.*, 58(2):336–375, 1999.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *21st ACM STOC*, pages 33–43,

Seattle, WA, USA, May 15–17, 1989. ACM Press. doi:10.1145/73007.73011.

[PLS13] Nicole Perlroth, Jeff Larson, and Scott Shane. Secret documents reveal nsa campaign against encryption, 2013. URL: https://archive.nytimes.com/www.nytimes.com/interactive/2013/09/05/us/documents-reveal-nsa-campaign-against-encryption.html [cited 29.10.2023].

[PR20] Bertram Poettering and Paul Rösler. Combiners for AEAD. *IACR Trans. Symmetric Cryptol.*, 2020(1):121–143, 2020. URL: https://doi.org/10.13154/tosc.v2020.i1.121-143, doi:10.13154/TOSC.V2020.I1.121-143.

[Pri15] Emily Price. Juniper networks security flaw may have exposed us government data, 23.12.2015. URL: https://www.theguardian.com/technology/2015/dec/22/juniper-networks-flaw-vpn-government-data [cited 29.10.2023].

[PT67] Harold E. Petersen and Rein Turn. System implications of information privacy. In *AFIPS Spring Joint Computing Conference*, volume 30 of *AFIPS Conference Proceedings*, pages 291–300. AFIPS / ACM / Thomson Book Company, Washington D.C., 1967.

[Rab79] Michael O. Rabin. *Digitalized signatures and public-key functions as intractable as factorization.* MIT Laboratory for Computer Science, 1979. Technical Report 212.

[RBBK01] Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. In Michael K. Reiter and Pierangela Samarati, editors, *ACM CCS 2001*, pages 196–205, Philadelphia, PA, USA, November 5–8, 2001. ACM Press. doi:10.1145/501983.502011.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press. doi:10.1145/1060590.1060603.

[Res18] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, August 2018. URL: https://www.rfc-editor.org/info/rfc8446, doi:10.17487/RFC8446.

[Rog02] Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, *ACM CCS 2002*, pages 98–107, Washington, DC, USA, November 18–22, 2002. ACM Press. doi:10.1145/586110.586125.

[Rog06] Phillip Rogaway. Formalizing human ignorance. In Phong Q. Nguyen, editor, *Progress in Cryptology - VIETCRYPT 06*, volume 4341 of *LNCS*, pages 211–228, Hanoi, Vietnam, September 25–28, 2006. Springer, Heidelberg, Germany.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[RTYZ16] Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Cliptography: Clipping the power of kleptographic attacks. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 34–64, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-53890-6_2.

[RTYZ17] Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Generic semantic security against a kleptographic adversary. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 907–922, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press. doi:10.1145/3133956.3133993.

[RTYZ18] Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Correcting subverted random oracles. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 241–271, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-96881-0_9.

[Sha49a] Claude E. Shannon. Communication theory of secrecy systems. *Bell Syst. Tech. J.*, 28(4):656–715, 1949.

[Sha49b] Claude E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656–715, 1949.

[SMC08] Joseph A. Salowey, David McGrew, and Abhijit Choudhury. AES Galois Counter Mode (GCM) Cipher Suites for TLS. RFC 5288, August 2008. URL: https://www.rfc-editor.org/info/rfc5288, doi:10.17487/RFC5288.

[von51] John von Neumann. Various techniques used in connection with random digits. In A.S. Householder, G.E. Forsythe, and H.H. Germond, editors, *Monte Carlo Method*, pages 36–38. National Bureau of Standards Applied Mathematics Series, 12, Washington, D.C.: U.S. Government Printing Office, 1951.

[Vor] Arbeitskreis Vorratsdatenspeicherung. Serious criminal offences, as defined in sect. 100a stpo, in germany according to police crime statistics. URL: http://www.vorratsdatenspeicherung.de/images/data_retention_effectiveness_report_2011-01-26.pdf [cited 12.10.2023].

[Wat] Human Rights Watch. New evidence that biometric data systems imperil afghans. URL: https://www.hrw.org/news/2022/03/30/new-evidence-biometric-data-systems-imperil-afghans [cited 12.10.2023].

[WC81] Mark N. Wegman and Larry Carter. New hash functions and their use in authentication and set equality. *J. Comput. Syst. Sci.*, 22(3):265–279, 1981.

[WCL+22] Yi Wang, Rongmao Chen, Chi Liu, Baosheng Wang, and Yongjun Wang. Asymmetric subversion attacks on signature and identification schemes. *Pers. Ubiquitous Comput.*, 26(3):849–862, 2022.

[WHF03] Doug Whiting, Russ Housley, and Niels Ferguson. Counter with CBC-MAC (CCM). RFC 3610, September 2003. URL: https://www.rfc-editor.org/info/rfc3610, doi:10.17487/RFC3610.

[WS10] Adam Waksman and Simha Sethumadhavan. Tamper evident microprocessors. In *2010 IEEE Symposium on Security and Privacy*, pages 173–188, Berkeley/Oakland, CA, USA, May 16–19, 2010. IEEE Computer Society Press. doi:10.1109/SP.2010.19.

[WS11] Adam Waksman and Simha Sethumadhavan. Silencing hardware backdoors. In *2011 IEEE Symposium on Security and Privacy*, pages 49–63, Berkeley, CA, USA, May 22–25, 2011. IEEE Computer Society Press. doi:10.1109/SP.2011.27.

[YCL+20] Zhichao Yang, Rongmao Chen, Chao Li, Longjiang Qu, and Guomin Yang. On the security of LWE cryptosystem against subversion attacks. *Comput. J.*, 63(4):495–507, 2020.

[YY97] Adam Young and Moti Yung. Kleptography: Using cryptography against cryptography. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 62–74, Konstanz, Germany, May 11–15, 1997. Springer, Heidelberg, Germany. doi:10.1007/3-540-69053-0_6.