A preliminary version of this paper appears in the proceedings of the *Cryptographer's Track at the RSA Conference 2020 (CT-RSA 2020)*, DOI: 10.1007/978-3-030-40186-3_4. This is the full version.

Modeling Memory Faults in Signature and Authenticated Encryption Schemes

Marc Fischlin[1] Felix Günther[2]

[1] Cryptoplexity, Technische Universität Darmstadt, Darmstadt, Germany
[2] Department of Computer Science, ETH Zürich, Zürich, Switzerland
marc.fischlin@cryptoplexity.de mail@felixguenther.info

January 16, 2020

Abstract. Memory fault attacks, inducing errors in computations, have been an ever-evolving threat to cryptographic schemes since their discovery for cryptography by Boneh et al. (Eurocrypt 1997). Initially requiring physical tampering with hardware, the software-based rowhammer attack put forward by Kim et al. (ISCA 2014) enabled fault attacks also through malicious software running on the same host machine. This led to concerning novel attack vectors, for example on deterministic signature schemes, whose approach to avoid dependency on (good) randomness renders them vulnerable to fault attacks. This has been demonstrated in realistic adversarial settings in a series of recent works. However, a unified formalism of different memory fault attacks, enabling also to argue the security of countermeasures, is missing yet.

In this work, we suggest a generic extension for existing security models that enables a game-based treatment of cryptographic fault resilience. Our modeling specifies exemplary memory fault attack types of different strength, ranging from random bit-flip faults to differential (rowhammer-style) faults to full adversarial control on indicated memory variables. We apply our model first to deterministic signatures to revisit known fault attacks as well as to establish provable guarantees of fault resilience for proposed fault-attack countermeasures. In a second application to nonce-misuse resistant authenticated encryption, we provide the first fault-attack treatment of the SIV mode of operation and give a provably secure fault-resilient variant.

Keywords. Fault attacks, security model, fault resilience, deterministic signatures, nonce-misuse resistant authenticated encryption

Contents

1 Introduction

Since their first treatment in the cryptographic realm by Boneh, DeMillo, and Lipton [BDL97] in 1997, fault attacks (i.e., attacks that induce unexpected disturbances during computations) have evolved as an important class of attacks to assess the strength of cryptographic systems. While the possibility of faults accidentally occurring in hardware chips was already known in the 1970s [MW78], the work by Boneh et al. as well as others [BDL97, JLQ99] demonstrated that faults can have devastating effects on the security of cryptographic systems, more specifically RSA and other signatures making use of the Chinese Remainder Theorem. The attack by Boneh, DeMillo, and Lipton inspired—beginning with Biham and Shamir introducing differential fault analysis [BS97]—a long line of research on different types of fault attacks challenging the security of cryptographic systems. These in particular encompass a wealth of different hardware tampering attacks, ranging from manipulation of the system's voltage, clock, or temperature to electromagnetic disturbances or laser irradiation (see, e.g., [BECN+06, BBKN12] for an overview).

For a long time, countermeasures against fault attacks focused on making the cryptographic hardware tamper-resilient (or tamper-proof). In 2014 however, a break-through research result by Kim et al. [KDK+14] demonstrated that faults can be remotely injected in modern hardware through software access only. More specifically, their attack leveraged that high-frequency repeated read/write operations to some memory address ("hammering") in DRAM memory may induce disturbance errors in other nearby addresses. Kim et al. described how in a so-called *rowhammer attack* a malicious process can induce controlled disturbances (i.e., bit flips as differential faults) in the memory of another process, circumventing the memory isolation security mechanisms of the computing system. In follow-up work, the rowhammer attack was refined further. Specifically, Razavi et al. [RGB+16] improved the attack in a way that enabled flipping individual bits in nearby memory in a fine-grained manner, even across the boundaries of virtual machines hosted on the same hardware.

It does not come as a surprise that software fault attacks like rowhammer can have critical security implications for cryptographic systems. Razavi et al. [RGB+16] already demonstrated how bit-flipping attacks in RSA public-keys stored by the SSH protocol for authentication [YL06] enable easy factorization and thereby break the authentication system. More recently, Poddebniak et al. [PSS+18] formalized rowhammer-style attacks that specifically target the setting of deterministic signature schemes, opening up a new type of attack vector in this area.

Deterministic signature schemes emerged from the insight that good randomness might not always be available in the signing process due to failures in the random number generation. This may be due to restricted hardware settings where no good randomness source is available or a result of badly implemented or flawed random number generators [GW96, GPR06, DGP07, CER08]. In such cases, signature schemes like DSA or ECDSA [Nat13] that crucially rely on good per-message randomness in the signing process will fail catastrophically. Prominent incident examples include the compromise of the ECDSA signature keys for Sony's Playstation 3 [fai10] or key leaks in cryptocurrencies [BR18, BH19].

To obviate the dependency on good randomness in the signing process, M'Raïhi et al. [MNPV99] put forward the concept of making signature schemes deterministic through what we call *de-randomization*. The idea is to replace the ephemeral randomness sampled in the signing process by the output of a random oracle [BR93] evaluated on the secret signing key and the message to be signed. This way, no genuine randomness source is required for signing while the used input remains uniformly random from the perspective of an adversary without knowledge of the secret signing key. The de-randomizing approach has been widely adopted, e.g., in the specification of deterministic versions of DSA and ECDSA through RFC 6979 [Por13] or upfront in the design of the EdDSA signature algorithm proposed by Bernstein et al. [BDL+11].

Poddebniak et al. [PSS+18] now show that the introduced determinism in such schemes enables new kinds of fault attacks. More specifically, they formalize how rowhammer-style attacks can be deployed to recover signing keys by injecting faults in the deterministic computation of ECDSA and EdDSA signatures. This is done in such a way that two signatures on two different messages are computed (one original, and one resulting from the memory fault attack), but with the signing algorithm (re-)using the same per-message random nonce. They then demonstrate the practical feasibility of their attacks on an EdDSA implementation in a realistic setting across virtual machines.

In their work, Poddebniak et al. [PSS+18, Section 9] touch upon a number of countermeasures. Notably, they specifically highlight that the commonly suggested countermeasure to verify the signature before releasing it in order to check correctness of the computation [BDL97, Len96] turns out to be ineffective in protecting against their attack: the resulting signature is actually valid for the message modified through the fault attack. They conclude that the only cryptographic mechanism that would render their attack infeasible is to re-integrate randomness in the signing process *in addition* to the deterministically derived per-message nonce. This supports the design of the XEdDSA signature scheme by Perrin [Per16] deployed in the Signal secure messaging protocol [Sig], which augments the EdDSA nonce derivation with an additional random value in order to protect against glitches in the computation, referring to an observation by Schmidt [Sch16].

In several works concurrent and closely related to that by Poddebniak et al. [PSS+18], Romailler and Pelissier [RP17], Ambrose et al. [ABF+18], as well as Samwel et al. [SBB+18, SB18] studied differential fault and side-channel attacks on deterministic signatures in general and the ECDSA and EdDSA schemes specifically, also revisiting a previous result by Barenghi and Pelosi [BP16]. Notably, all works agree that adding randomness back into the signing process is necessary in order to prevent the described fault attacks. Indeed, the lattice-based signature proposals qTesla[1] and Dilithium[2] for NIST's post-quantum standardization process both include now a randomized version in the second round update because of the attacks.

1.1 Contributions

At this point, the current state of understanding of memory fault attacks (on deterministic signatures and more generally) leaves us with the questions of how to formally capture different types of memory faults and relate their strength, and how to assess whether proposed attack countermeasures indeed provide security against certain classes of fault attacks. In this work, we approach an answer to these questions through establishing a generalized game-based security model capturing cryptographic fault resilience. We then apply this model to recapitulate the fault attacks discussed, establish provable security results for proposed countermeasures, and derive novel measures for the setting of nonce-based authenticated encryption.

Security model extension for fault resilience. We introduce, in Section 3, a game-based framework for extending existing security models in order to capture memory fault attacks resp. resilience against such attacks. Our approach generalizes fault attacks of different strength on memory variables through a modeling technique akin to callback functions in programming languages. The specific types we define range from full adversarial control to controlled (rowhammer-style) bit flips to random faults, both transient and persistent; further types of memory fault attacks can be easily captured in our formalism.

As a result, our security model on the one hand allows us to formalize weaknesses in a cryptographic scheme through describing memory fault attacks as an abstract set of adversarial interactions with the scheme. On the other hand, the model enables us to positively establish provable security results for the fault resilience of a scheme against well-defined classes of fault attacks. We will use our model in the

[1] https://qtesla.org/
[2] https://pq-crystals.org/

former way to demonstrate how known memory fault attacks are reflected in the model. In the latter way, we employ it to evaluate the provable security guarantees of potential countermeasures reconciling weak-randomness and fault-attack protection.

Fault resilience of signatures. We then apply our model (in Section 4) to assess the fault resilience of digital signature schemes. To this end, we first augment the classical notion of unforgeability with our security model extension to capture memory fault attacks. A key point in the augmented model is to attribute the signature to a message, because the adversary may alter the message content during the signing process. The extension enables us to formally restate the concept of above fault attacks on deterministic signatures [PSS+18, RP17, ABF+18, SBB+18] in terms of our security model, as a sanity check for our modeling so to speak.

More importantly, we then formalize the proposed countermeasure to include additional randomness in the signature generation process along with potential fault-attack vectors. One countermeasure, used in XEdDSA, is to derive the necessary randomness for signing by applying a pseudorandom function to the message, but also mixing in a random value in this pseudorandom function evaluation. An alternative countermeasure is to compute the exclusive-or of the pseudorandom value with the random string. We are able to formally establish that both approaches indeed achieve the desired goal of providing combined security: achieving fault resilience when good randomness is present while upholding regular security of a de-randomized scheme under arbitrarily weak randomness.

Fault resilience of authenticated encryption. Finally, we demonstrate the generality of our security model extension by applying it to another setting (in Section 5), namely that of nonce-based and nonce-misuse resistant authenticated encryption [Rog02, Rog04, RS06]. Somewhat similar to the setting of deterministic signatures, nonces were introduced to authenticated encryption schemes in order to obviate the need for randomness in the encryption process, again for (good) randomness not always being available.

There has been some preliminary work on fault attacks on nonce-based authenticated encryption (e.g., [DEK+16, DMMP19]). To the best of our knowledge, we however provide the first fault-attack treatment of the SIV mode of operation proposed by Rogaway and Shrimpton [RS06], aiming also at nonce-misuse resistance. Unfortunately, the SIV mode does not provide any fault resilience even under the weakest types of (random single-bit flip) fault attacks in our model. However, we can show that translating concepts similar to the additional-randomness countermeasure for deterministic signatures allows us to derive a randomness-augmented mode SIV$ which provides strong misuse-resistant authenticated encryption security while protecting against differential fault attacks.

1.2 Further Related Work

Faults in cryptographic schemes and formal ways of establishing fault resilience have been studied in different settings before. Ishai et al. [IPSW06] model faults in gate-wise computations in (conducting) circuits, focusing rather on hardware than on memory-based faults like rowhammer. Their approach ensures security through "self-destructing circuits," whereas our model aims at upholding functionality *and* security under a defined class of faults. Faults in (memory) variables of cryptographic schemes have been considered by Coron and Mandal [CM09] in their provable-security model tailored to random faults in RSA signatures. Barthe et al. [BDF+14] treated non-random fault attacks on RSA in a model generalizing attacks from [FGL+12]. Extending the principle idea of provable-security treatment of memory-variable faults, we provide a generic security model capturing general memory faults in arbitrary cryptographic primitives.

Memory-based fault attacks like rowhammer can also be used to modify the *control flow* of programs (through return addresses and the like). Similar tampering with program control flow is possible through

a range of hardware tampering, in cases enabling fine-grained instruction skipping [BECN+06, BBKN12]. This naturally also effects cryptographic implementations (see, e.g., attacks on elliptic curve cryptography [BMM00, BG15, TT19]) and could potentially be seen as an extreme, transient form of algorithm substitution attacks [BPR14]. It remains unclear how cryptographic schemes themselves can counter control-flow faults, and thus in this work we focus on faults modifying their *data* residing in memory.

Related-key attack (RKA) security [BK04, GLM+04] studies fault attacks in a setting where faults are restricted to the key material of cryptographic primitives, bound to a class of related-key deriving functions. While RKA security can be a building block for achieving strong fault resilience, our model more generally considers memory faults of various types that affect arbitrary memory variables. We leave studying the detailed relationship between RKA security notions and memory fault resilience as a possible avenue for future work.

As remarked above, one of the challenges for signature schemes is to link the signature to a message, because the message may change during the signing process. The notion of incremental cryptography [BGG94] faces a similar problem of attributing signature creations to messages in a setting where the adversary may tamper with the input. The idea of incremental signature schemes is to sign a message from scratch, and when the message is later slightly edited, one is able to update the signature fast by accessing only a few message blocks. In a strong notion for virus protection [BGG95], Bellare et al. consider the possibility that the adversary may alter the message before making an update call to create a new signature. Since the update algorithm can only access a bounded number of message blocks it cannot check validity of the entire message and potentially works on a substituted message. From a security viewpoint this too raises the question which message one assigns to the derived signature. Bellare et al. [BGG95] correlate the unaltered message which the signer would have expected to the signature.

Note that incremental signatures on the one hand touch a simpler problem than in our case. This is so because, there, the adversary can change the message only once, before calling the signature creation. In contrast, our adversary may continuously provide different values during the signing process, every time the data is accessed. At the same time our case does not deal with fast updates and may read the entire message. When adapting our fault-resilience model to the setting of signatures in Section 4.1, we will see that with introduced faults, the challenge message-signature pair to record turns out to be the (at most) one valid combination seen among all faulted variables.

In the setting of hedged public-key encryption as introduced by Bellare et al. [BBN+09], similar combiner techniques are employed as in the countermeasures reconciling weak-randomness and fault-attack protection for deterministic signatures and authenticated encryption we discuss in Sections 4.3 and 5.3. We leave it as an open question for future work to study whether such techniques enable fault-resilient security for hedged public-key encryption, too.

1.2.1 Concurrent Work

In concurrent and independent work, Aranha et al. [AOTZ19] studied the security of hedged randomness derivation in Fiat-Shamir–type signatures under fault attacks. Focusing on the Fiat-Shamir transform, they treat tailored (memory) fault types occurring in such design and particularly study Schnorr signatures as well as the NIST post-quantum signature candidate Picnic2[3]. Their model considers a limited adversary capable of injecting (only) a single fault as setting or flipping a single bit in a function input or output. Our approach is more generic, introducing a generic extension to capture arbitrary and strong memory fault attacks in any cryptographic scheme. Beyond also studying signatures and their de-randomization and hedging as prime practical example, we exemplify this generality by furthermore treating nonce-misuse–resistant authenticated encryption in our framework.

[3] https://microsoft.github.io/Picnic/

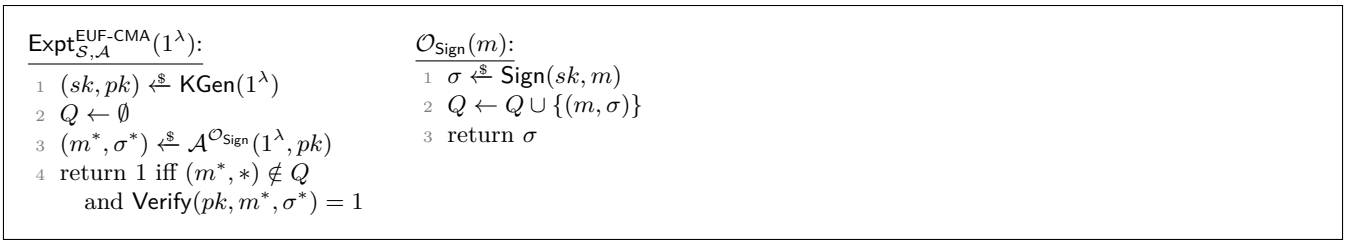

Figure 1: Security experiment for *existential unforgeability under chosen-message attacks* (EUF-CMA) for signature schemes. We write $(a, *) \notin Q$ if $\nexists\, b$ s.t. $(a, b) \in Q$.

2 Preliminaries

We briefly introduce some notation and recap the standard (security) definitions for digital signatures and authenticated encryption.

2.1 Notation

We denote by $\mathbb{N}$ the natural numbers and by $\lambda \in \mathbb{N}$ the security parameter. We write a bit as $b \in \{0,1\}$ and a (bit) string as $s \in \{0,1\}^*$ with $|s|$ indicating its (binary) length. By $s||t$ we denote concatenation of two bit strings s, t as well as vector concatenation of two vectors s, t. By $s \oplus t$ we denote the bitwise XOR of two bit strings s, t with $|s| = |t|$. We indicate the Hamming weight of a bit string s by $\mathsf{hw}(s)$.

2.2 Digital Signatures

We recap the syntax of digital signatures and their standard security notion of existential unforgeability under chosen-message attack [GMR88].

Definition 2.1 (Signature scheme)**.** *A* signature scheme $\mathcal{S} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ *consists of three efficient algorithms defined as follows.*

- $\mathsf{KGen}(1^\lambda) \xrightarrow{\$} (sk, pk)$. *On input the security parameter* 1^λ*, this probabilistic algorithm outputs a secret signing key* sk *and a public verification key* pk*.*
- $\mathsf{Sign}(sk, m; r) \xrightarrow{\$} \sigma$. *On input a signing key* sk*, a message* $m \in \{0,1\}^*$*, and (optionally) randomness* $r \in \{0,1\}^*$*, this (possibly) probabilistic algorithm outputs a signature* σ*. The randomness* r *is omitted if* Sign *is deterministic or randomness is handled internally.*
- $\mathsf{Verify}(K, m, \tau) \to \{0,1\}$. *On input a verification key* pk*, a message* m*, and a signature* σ*, this deterministic algorithm outputs* 1 *(indicating validity of the signature) or* 0 *(otherwise).*

Definition 2.2 (Existential unforgeability of signatures)**.** *Let* $\mathcal{S} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ *be a signature scheme and experiment* $\mathsf{Expt}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{S},\mathcal{A}}(1^\lambda)$ *for an adversary* $\mathcal{A}$ *be defined as in Figure 1.*

We say that $\mathcal{S}$ *provides* existential unforgeability under chosen-message attacks *(*EUF-CMA*) if for all PPT adversaries the following advantage function is negligible in the security parameter:*

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{S},\mathcal{A}} := \Pr\left[\mathsf{Expt}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{S},\mathcal{A}}(1^\lambda) = 1\right].$$

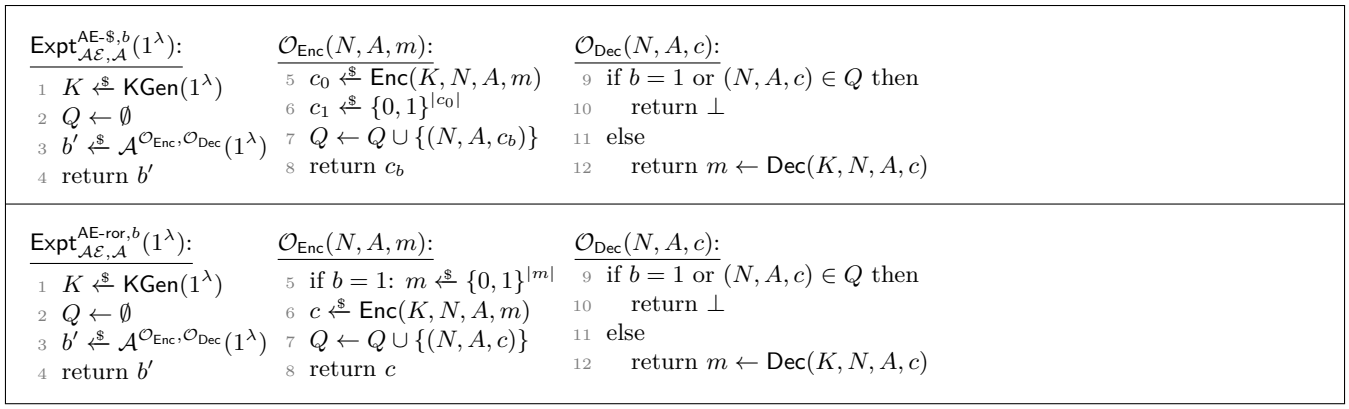

Figure 2: Security experiments for authenticated encryption schemes.

2.3 Authenticated Encryption

We also recap syntax and security of nonce-based authenticated encryption schemes with associated data [Rog02]. While the encryption algorithm of nonce-based authenticated encryption is generally considered to be deterministic, we liberally also allow probabilistic encryption here in order to accommodate fault-resilient constructions combining nonces and randomness under the same syntax. We call an authenticated encryption scheme deterministic if its encryption algorithm is deterministic.

Definition 2.3 (Authenticated encryption scheme)**.** *A nonce-based* authenticated encryption scheme *with associated data* $\mathcal{AE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *consists of three efficient algorithms defined as follows.*

- $\mathsf{KGen}(1^\lambda) \xrightarrow{\$} K$. *On input the security parameter* 1^λ*, this probabilistic algorithm outputs a secret key* $K \in \{0,1\}^\lambda$.
- $\mathsf{Enc}(K,N,A,m;r) \xrightarrow{\$} c$. *On input a key* K*, a nonce* $N \in \{0,1\}^*$*, an associated-data value* $A \in \{0,1\}^*$*, a message* $m \in \{0,1\}^*$*, and (optionally) randomness* $r \in \{0,1\}^*$*, this possibly probabilistic algorithm outputs a ciphertext* $c \in \{0,1\}^*$*. The randomness* r *is omitted if* Enc *is deterministic or randomness is handled internally.*
- $\mathsf{Dec}(K,N,A,c) \to m$. *On input a key* K*, a nonce* $N \in \{0,1\}^*$*, an associated-data value* $A \in \{0,1\}^*$*, and a ciphertext* $c \in \{0,1\}^*$*, this deterministic algorithm outputs either a message* m *or a distinct error symbol* $\perp$.

We formalize security for authenticated encryption via all-in-one definitions capturing both confidentiality and integrity (cf., e.g., [Shr04, RS06, NRS14]), considering two flavors: Randomness indistinguishability (AE-$) requires ciphertext to be indistinguishable from random strings. Real-or-random indistinguishability (AE-ror) is strictly weaker and only demands that ciphertexts be indistinguishable from encryptions of a random, equal-length message [BDJR97].

Definition 2.4 (Security of authenticated encryption)**.** *Let* $\mathcal{AE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *be an authenticated encryption scheme and experiments* $\mathsf{Expt}^{\mathsf{AE\text{-}\$},b}_{\mathcal{AE},\mathcal{A}}$ *and* $\mathsf{Expt}^{\mathsf{AE\text{-}ror},b}_{\mathcal{AE},\mathcal{A}}$ *for an adversary* $\mathcal{A}$ *and a bit* b *be defined as in Figure 2. We restrict* $\mathcal{A}$ *to ask any query* (N,A,m) *to* $\mathcal{O}_{\mathsf{Enc}}$ *at most once.*

We say that $\mathcal{AE}$ *is* AE-$-secure*, resp.* AE-ror-secure*, if for all PPT adversaries and* AE-SEC = AE-$*, resp.* AE-SEC = AE-ror*, the following advantage function is negligible in the security parameter:*

$$\mathsf{Adv}^{\mathsf{AE\text{-}SEC}}_{\mathcal{AE},\mathcal{A}} := \left| \Pr\left[\mathsf{Expt}^{\mathsf{AE\text{-}SEC},0}_{\mathcal{AE},\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{AE\text{-}SEC},1}_{\mathcal{AE},\mathcal{A}}(1^\lambda) = 1\right] \right|.$$

When $\mathcal{A}$ *never repeats the nonce value* N *between any two* $\mathcal{O}_{\mathsf{Enc}}$ *calls, we call it* nonce-respecting*; otherwise we say the scheme is* nonce-misuse resistant *[RS06].*

3 Modeling Fault Resilience

We begin with developing our generic security model extension for capturing memory fault attacks on cryptographic primitives. Such attacks arise through various means in practice and may range from single or few random bit-flips over rowhammer-style controlled flips of one or several bits to full control over the memory enabling injection of arbitrary values. Their effects may be transient and vanish after some subsequent memory access, or a persistent change to the affected bits in memory. In the security model extension we propose in the following, we capture all these different types of faults in a generic manner and formally relate their strength.

At the heart of our model is the observation that while memory fault attacks may be executed at arbitrary points during an execution, they come into effect only when variables are *read* from memory. We therefore capture the adversary's capability to induce faults (of various types) into memory by providing it with means to influence variable values when an algorithm reads them from memory (i.e., uses them). Technically, we model such influence by introducing *callbacks* to the adversary whenever a variable x is used within an algorithm. Resembling callback functions in programming languages, an adversary is then given the option to *alter* (i.e., fault) the value read/used for this variable.

The ways the adversary is allowed to alter the variable reflects the type of fault attack in consideration: In a *full fault* attack the adversary can provide an arbitrary value to be used. In a *differential fault* attack (flipping bits in a controlled way, as in the rowhammer attack [KDK+14, RGB+16]), the adversary instead provides a bitstring to be XORed to the variable it is used (while not learning the resulting value itself). In a fault attack introducing *random faults*, the adversary finally can merely choose how many bits to be flipped (with neither control over the position nor obtaining the resulting value). In all cases, the introduced fault can be either *transient*, applying only to the one read operation faulted, or *persistent*, in which case the variable is overwritten with the faulted value.

Our model does not fix one type of fault attack, but flexibly allows to consider different attack types for each individual memory variables in a scheme. This captures that some memory variables may be harder to fault than others, e.g., for being shorter (and thus more difficult to target with rowhammer-style bit flips) or residing in specially-protected memory. To enable this flexibility, we first of all explicitly indicate in syntax that some memory variable x is considered to be faultable by writing it as $\llcorner x \lrcorner$ with corner brackets when assigned. We then indicate positions where a variable x can be faulted, modeled through an adversarial callback, by writing its usage as $\langle x \rangle$ within angle brackets. This finally enables security statements that formalize individual fault attacks on each annotated variable. For example, we can that way capture an attacker injecting (in the same attack) differential fault attacks into some variable x and random fault attacks into some other variable y.

Applying our security model extension to existing game-based security definitions yields notions that capture the original type of security under the considered fault attacks. To this end, the cryptographic scheme under consideration is augmented by adding indications for faultable memory variables (e.g., $\llcorner x \lrcorner$) and callbacks (e.g., $\langle x \rangle$) in its algorithm descriptions. The actual security experiments remain syntactically largely unchanged, but now incorporate adversarial faulting access to memory variables as indicated by the scheme.

Observe that while the extended security model's dependency on the particular implementation and memory variable layout of a scheme might, at first glance, seem to yield a somewhat dedicated security result, such dependency is ultimately not surprising: the (non-)vulnerability of a scheme to memory fault attacks inevitably depends on the handling of memory variables. At the same time, abstract cryptographic algorithm representations are still reasonably close to their implementation in terms of memory variables, and our model captures strong and fine-grained adversarial faults on those variables.

A noteworthy change in the augmented security experiment however may regularly be required in the evaluation of winning conditions and permissible queries. As the latter may rely on faultable variables, we

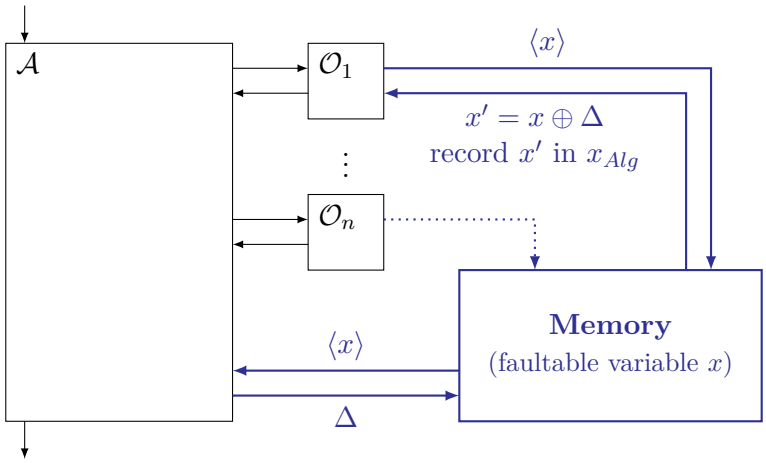

Figure 3: Illustration of how our proposed extension for fault resilience (on the right in blue) integrates through callbacks with the interaction of an adversary $\mathcal{A}$ and oracles $\mathcal{O}_1, \ldots, \mathcal{O}_n$ within some classical security experiment (on the left in black). As an example, we depict the callback query $\langle x \rangle$ and response for a transient differential fault on some variable x.

need to define which of possibly several values of the now changing variable to use when evaluating such conditions. For this purpose, our extension further provides access to the list of values that each faultable variable took within some algorithm: we write x_{Alg} for the sequential list of values that variable x took within some previously invoked algorithm Alg. The unforgeability experiment for signatures detailed in Section 4 is an example for such a modified winning condition. There, we will make use of the list m_{Sign} containing all values of the message variable m used within the signing algorithm to define the list of original signatures the adversary obtained through the signing oracle.

3.1 Fault Types

For our security model extension, we explicitly specify four different types of faults that an adversary may inject, and further distinguish between transient and persistent faults. We however stress that the model itself is generic and can be extended to encompass further fault types if desired.

On any read of a faultable variable x indicated by a callback $\langle x \rangle$, the adversary $\mathcal{A}$ is invoked with an identifier for the read variable, indicated by $\mathcal{A}(\langle x \rangle)$. ($\mathcal{A}$ implicitly keeps state between callbacks.) Note that this identifier is merely a handle in order for $\mathcal{A}$ to know *which variable* the callback is for, but without learning the *variable value* itself. Of course, the adversary knows the scheme's code itself; we furthermore let the handle for a variable also disclose the variable's bit-length to $\mathcal{A}$. In case of *transient* faults, the callback only temporarily modifies the value read for this variable for this specific read operation, but does not alter the variable itself beyond that. I.e., several transient-fault callbacks $\langle x \rangle$ on some variable x are always with respect to the original, non-faulted variable value of x. In contrast, for *persistent* faults, the callback modifies the variable in memory, which then also is used for the actual read operation.

Beyond the distinction between transient and persistent faults, the fault-injection callback $\langle x \rangle$ for some variable x behaves differently for each fault type as described in the following and formalized in Figure 4. We further illustrate the integration of our callback-based model with an existing security experiment at the example of a transient differential fault attack in Figure 3.

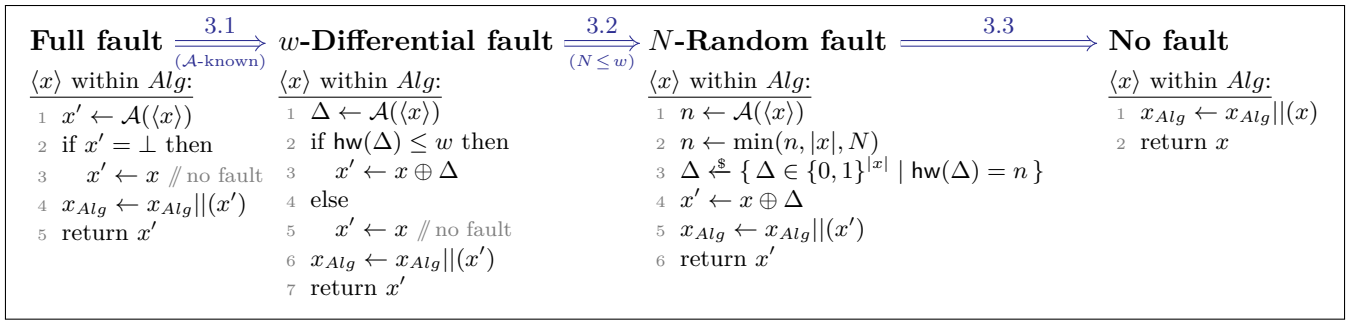

Figure 4: Specification of and implications between the four fault types: full faults, (w-)differential faults, (N-)random faults, and no faults. In case of a persistent fault, the returned value also overwrites the variable value. Implication arrows are annotated with the respective lemma (above) and conditions (below).

Full faults: In a full fault attack, the adversary is allowed to arbitrarily modify the faulted memory variable x.[4] This is modeled by giving the adversary full control over the variable whenever it is read.

Differential faults: In a differential-fault attack, the adversary can flip (up to) a certain number $w \in \mathbb{N}$ of bits in the faulted memory variable x in a controlled way. This is modeled by having the adversary supply a difference value Δ which is then XORed to the variable value whenever read, where the Hamming weight $\mathsf{hw}(\Delta)$ of the difference value must not exceed w. As a shorthand, whenever $w \ge |x|$, we omit w.

Random faults: In a random-fault attack, the adversary can flip (up to) a certain number $N \in \mathbb{N}$ of random bits in the faulted memory variable, without controlling which bits are flipped. This is modeled by letting the adversary specify a number $n \le N$ whenever the variable is read and then flip n randomly positioned bits of the variable value in the callback response. As a shorthand, whenever $N \ge |x|$, we omit N.

No faults: For completeness, we also specify a "no-fault" behavior of the variable callback (directly returning x), which enables formal comparisons of classical security notions within the same notational framework. In general, we omit annotating callbacks for non-faulted variables, though.

3.2 Relations

It is not surprising that full faults represent the strongest fault attacks in our model on memory variables known by the adversary at the time of the callback, e.g., some public parameter or a message input provided to a signing algorithm by $\mathcal{A}$. An adversary can capture any other fault behavior on such variables (which we call "$\mathcal{A}$-known") by providing the resulting faulted variable value directly. Note that this is not true for memory variables unknown to the adversary (e.g., the secret-key input to a signing algorithm): for such variables, the capability to flip bits is incomparable in power to overwriting the value with an adversarially-chosen one.

Furthermore, differential faults imply random faults for $N \le w$, as the adversary can sample a difference value Δ encoding $n \le N$ random bit flips on its own, which has permissible Hamming weight $\mathsf{hw}(\Delta) = n \le N \le w$. Finally, all fault types imply no faulting, as each allow the choice to leave the variable value unchanged.

[4] The adversary can opt to not modify the variable by returning a special symbol $\bot$.

Regarding the relations between transient and persistent faults, both variants are trivially equivalent for the full and no-fault types. In the case of differential and random faults, however, transient and persistent faults are indeed distinct adversarial capabilities, as the accumulation of persistent fault injections cannot be reproduced transiently if the number of bit flips or random bit faults on a variable is restricted (to less than $|x|$ for a differentially-faulted variable x).

We capture these expected relations between the different fault types in the following three lemmas, providing a brief formal argument in each case. The resulting implications are indicated by arrows in Figure 4.

Lemma 3.1 (Full faults $\overset{(\mathcal{A}\text{-known})}{\Longrightarrow}$ w-differential faults)**.** *For any security experiment, any PPT adversary $\mathcal{A}$, and any $w \in \mathbb{N}$, if $\mathcal{A}$ is successful in the experiment with (transient or persistent) w-differential faults on some variable x in algorithm Alg, with x being known by $\mathcal{A}$, then there exists an adversary $\mathcal{A}'$ successful in the experiment with (transient or persistent) full faults on x in Alg.*

Proof sketch. Since $\mathcal{A}$ knows x itself, an adversary $\mathcal{A}'$ can mimic $\mathcal{A}$'s behavior through full faults. Whenever $\mathcal{A}$ replies to a differential-fault callback $\langle x \rangle$ on x with a difference value Δ, $\mathcal{A}'$ replies to its full-fault callback with $x \oplus \Delta$ (accumulating persistent faults), resulting in the same variable value being used. □

Lemma 3.2 (w-differential faults $\overset{(N \leq w)}{\Longrightarrow}$ N-random faults)**.** *For any security experiment, any PPT adversary $\mathcal{A}$, and any $w, N \in \mathbb{N}$ with $N \leq w$, if $\mathcal{A}$ is successful in the experiment with transient (resp. persistent) N-random faults on some variable x in algorithm Alg, then there exists an adversary $\mathcal{A}'$ successful in the experiment with transient (resp. persistent) w-differential faults on x in Alg.*

Proof sketch. Observe that $\mathcal{A}'$ can mimic $\mathcal{A}$'s behavior as follows: whenever $\mathcal{A}$ replies with some value $n \leq N$ to a random-fault callback $\langle x \rangle$ on x, $\mathcal{A}'$ instead samples n distinct random positions $p_1, \ldots, p_n \xleftarrow{\$} \{1, \ldots, |x|\}$ and replies with a difference value $\Delta \in \{0,1\}^{|x|}$ which is the all-zero string except for bit positions $p_1, \ldots, p_n$. Such response results in the same variable value and is permissible as $\mathsf{hw}(\Delta) = n \leq N \leq w$. This strategy works both in the transient and in the persistent fault setting. In the persistent case, the differential faults of $\mathcal{A}'$ accumulate, correctly mimicking the accumulating random faults of $\mathcal{A}$. □

Lemma 3.3 (Full / w-differential / N-random faults $\implies$ no faults)**.** *For any security experiment, any PPT adversary $\mathcal{A}$, and any $w, N \in \mathbb{N}$, if $\mathcal{A}$ is successful in the experiment without faults on some variable x in algorithm Alg, then there exist adversaries $\mathcal{A}'$, $\mathcal{A}''$, and $\mathcal{A}'''$ successful in the experiment with (transient or persistent) full faults, w-differential faults, resp. N-random faults on x in Alg.*

Proof sketch. In the case of full faults, $\mathcal{A}'$ can mimic $\mathcal{A}$'s behavior by always returning the special symbol $\perp$ on a callback $\langle x \rangle$. In the case of differential faults, $\mathcal{A}'$ mimics the behavior by always replying with the zero-string $\Delta = 0^{|x|}$ to $\langle x \rangle$. In the case of random faults, $\mathcal{A}'$ does so by always replying 0 to $\langle x \rangle$. □

One can also argue that the notions form a strict hierarchy (i.e., that the reverse implications do not hold), if used to attack cryptographic schemes. One may for now consider some abstract scheme and x to be a random λ-bit string which is known to the adversary, and one lets the adversary win if it manages to bend (parts of) x to 0-bits (e.g., let the scheme then leak the secret key). If the adversary needs to return $x' = 0^\lambda$ then it can easily succeed in a full-fault attack, but for $w = \lambda/2$ it can change at most half of the bits of x such that the other bits would be 0 with probability at most $2^{-\lambda/2}$. Similarly, if the task of the adversary to flip $w = \lambda/2$ bits in x to 0, then it can easily succeed in a w-differential attack, but will fail to do so in an N-random fault attack with overwhelming probability.

4 Fault-Resilient Signatures

As the first application of our security model extension, we consider fault attacks against signature schemes and study the resilience of different designs against such attacks. We begin by augmenting the classical security notions for existential and strong unforgeability under chosen-message attacks for signatures with our extension to capture fault resilience, as described in Section 3. We then study the effects of faults specifically on a de-randomized (deterministic) signature schemes and analyze to which extent the proposed countermeasure to include additional randomness [PSS+18, RP17, ABF+18, SBB+18] provably provides fault resilience.

4.1 Fault-Resilient Signature Unforgeability

When augmenting the security notion for classical signature unforgeability, the essential question to answer is: which message–signature pairs did the adversary trivially learn through its signing oracle $\mathcal{O}_{\mathsf{Sign}}$ while tampering the message input *during* the signing process?

In the classical EUF-CMA security experiment without faults (cf. Figure 1), the adversary $\mathcal{A}$ obtains a signature σ on message m under secret key sk, and the oracle $\mathcal{O}_{\mathsf{Sign}}$ records (m, σ) in the set of oracle signatures Q. In the fault-resilience setting, the adversary however is now able to modify the message while the signing process is going on. As the simplest case, imagine $\mathcal{A}$ submitting some message m to the signing oracle, but then introducing a single-bit fault when the message is read once within the scheme's Sign algorithm, leading to the signature being produced on some $m' \neq m$. If the fault-resilient unforgeability experiment simply recorded (m, σ) in the oracle signature list Q, then $\mathcal{A}$ could trivially win against any signature scheme by outputting (m', σ) as its forgery.

The key observation for lifting the classical signature unforgeability experiment to the fault-resilience setting is hence that the list Q should record the signature σ together with the *actual* message it was generated on by the signing algorithm. With the adversary being able to potentially fault the message several times during the signing process (depending on the structure of the latter), it at first sight may seem unclear which of the messages in the set m_{Sign} of messages accessed during the signing process to record in Q. Our definition is based on the idea to include the messages which the signer "assumes to have signed correctly" during the attack, i.e., we restrict ourselves to the subset $m^{\mathsf{valid}}_{\mathsf{Sign}}$ of messages for which the output signature σ actually verifies under the challenge public key pk and which are not already included in Q. In other words, these are the new messages which the signer may have authenticated in the signing step. If there are two or more such valid messages in $m^{\mathsf{valid}}_{\mathsf{Sign}}$ then the signer cannot reliably identify the intended message. In this case we declare the adversary to win, captured via a flag clash which is set to `true` if there are multiple messages in $m^{\mathsf{valid}}_{\mathsf{Sign}}$ for any request.

The above definition in particular complies with the case that the adversary mounts a regular attack and does not tamper with the messages at all. In this case we would collect all signed messages in Q—one for each $\mathcal{O}_{\mathsf{Sign}}$ query (unless a message repeats and we do not extend Q)—as in the regular case, but clash would never become `true`. The fact that we declare the adversary to win if there are two messages in $m^{\mathsf{valid}}_{\mathsf{Sign}}$ immediately, without requiring the adversary to output the other (faulted) message as a forgery, releases the adversary from having to know the other message. This gives a stronger security guarantee, especially for random faults where bit flips may happen at unknown positions.

Put together, our signature unforgeability experiment adapted to the fault resilience setting allows the adversary to inject faults within the signature generation (as specified by the signature scheme in question). In its list of obtained signatures Q, it records the *first* value of the messages m used within the signing algorithm for which the generated signature σ verifies under the challenge public key. The augmented security definition for fault-resilient signature unforgeability is as follows; the according security experiment in Figure 5 highlights the changes from the classical experiment.

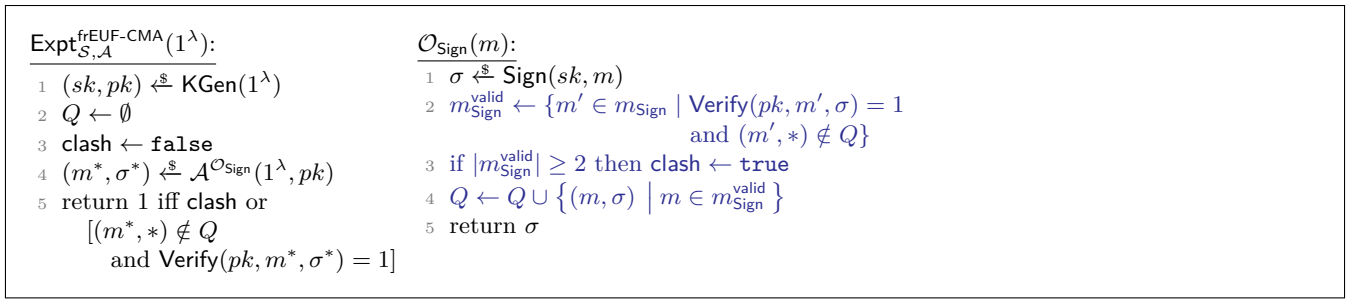

Figure 5: Security experiment for *fault-resilient existential unforgeability under chosen-message attacks* (frEUF-CMA) for signature schemes. We write $(a, *) \notin Q$ if $\nexists b$ s.t. $(a, b) \in Q$. The lines 2–4 in $\mathcal{O}_{\mathsf{Sign}}$ are changed compared to the classical EUF-CMA notion in Figure 1. Recall that m_{Sign} is the set of values the message variable m took during the signing process in line 1 due to fault callbacks.

Definition 4.1 (Fault-resilient existential unforgeability of signatures)**.** *Let $\mathcal{S} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme and experiment $\mathsf{Expt}^{\mathsf{frEUF\text{-}CMA}}_{\mathcal{S},\mathcal{A}}$ for an adversary $\mathcal{A}$ be defined as in Figure 5.*

We say that $\mathcal{S}$ provides fault-resilient existential unforgeability under chosen-message attacks *(*frEUF-CMA*) if for all PPT adversaries the following advantage function is negligible in the security parameter:*

$$\mathsf{Adv}^{\mathsf{frEUF\text{-}CMA}}_{\mathcal{S},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{Expt}^{\mathsf{frEUF\text{-}CMA}}_{\mathcal{S},\mathcal{A}}(1^\lambda) = 1\right].$$

4.2 De-randomized Signatures Are Not Fault-Resilient

We now exercise our fault-resilient unforgeability notion to establish that de-randomized schemes are vulnerable to the weakest fault injection attack of random one-bit flips. This in particular confirms the corresponding observations by Poddebniak et al. and others [PSS+18, RP17, ABF+18, SBB+18] in our formalism. To recap, de-randomization here refers to the approach to deterministically extract a per-message random value from the secret signing key and message input, replacing an otherwise needed true random sampling of a per-message nonce. This approach is employed, e.g., in the deterministic variants of the DSA and ECDSA signature schemes [Por13] and similarly in a more direct manner in the EdDSA signature scheme [BDL+11]. The latter scheme actually uses two pseudorandomly derived sub keys for signing and for nonce generation but this does not invalidate the attack.

We establish our result through the following abstractly de-randomized signature scheme $\mathcal{S}_{\mathsf{dr}}$ generalizing the above approach. The scheme $\mathcal{S}_{\mathsf{dr}} = (\mathsf{KGen}_{\mathsf{dr}}, \mathsf{Sign}_{\mathsf{dr}}, \mathsf{Verify}_{\mathsf{dr}})$ de-randomizes a randomized signature scheme $\mathcal{S} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$. In order to generate necessary randomness for $\mathcal{S}$'s signing algorithm, $\mathsf{Sign}_{\mathsf{dr}}$ invokes a cryptographic hash function $\mathsf{H}\colon \{0,1\}^* \to \{0,1\}^{\geq\lambda}$ (modeled as a random oracle [BR93]) on the scheme's secret signing key and the message to be signed. The key generation and verification algorithms $\mathsf{KGen}_{\mathsf{dr}}$ and $\mathsf{Verify}_{\mathsf{dr}}$ are as for the randomized scheme, the modified signing algorithm $\mathsf{Sign}_{\mathsf{dr}}$ is defined as follows:

$\mathsf{Sign}_{\mathsf{dr}}(sk, \llcorner m \lrcorner)$:
1 $r \leftarrow \mathsf{H}(sk, \langle m \rangle)$
2 $\sigma \leftarrow \mathsf{Sign}(sk, \langle m \rangle; r)$
3 return σ

In order to capture fault attacks, the definition of $\mathsf{Sign}_{\mathsf{dr}}$ defines the message m to be faultable (indicated through corner brackets $\llcorner m \lrcorner$ on definition and angle brackets $\langle m \rangle$ on reads).[5]

[5]For completeness, observe that the fault attack described in the following applies also when introducing faults into r

As required by the DSA and ECDSA standards [Nat13], the per-message random number (or *nonce*) r must be freshly generated for each message to be signed. If not, two signatures σ_0, σ_1 generated on distinct messages $m_0 \neq m_1$ using the *same* nonce r enable recovery of the secret signing key sk from the two signature equations [Vau03]. In the de-randomized versions of DSA and ECDSA, and likewise in the deterministic EdDSA scheme, this requirement is aimed to be satisfied through deterministically deriving the random nonce via a hash function from the secret signing key and input message.

However, as observed before [PSS+18, RP17, ABF+18, SBB+18], a fault introduced within the message memory variable m between reading m for deriving the nonce r and reading m again for computing the signature (with nonce r), recovers the nonce reuse scenario and, with it, a signing key extraction attack. In the following theorem, we formalize this observation in our generalized fault resilience setting. Let us call the underlying randomized signature scheme $\mathcal{S}$ *forgeable under nonce repetition* when given two distinct messages $m_0 \neq m_1$ and two valid corresponding signatures σ_0, σ_1 generated with the same random nonce r it is easy to produce an $\mathsf{EUF\text{-}CMA}$ forgery signature σ^* for some fresh message $m^* \notin (m_0, m_1)$. In particular, DSA, ECDSA, and the signing process underlying EdDSA are forgeable under nonce repetition.

Theorem 4.2. *Let $\mathcal{S}$ be a signature scheme forgeable under nonce repetition. Then the de-randomized signature scheme $\mathcal{S}_{\mathsf{dr}} = (\mathsf{KGen}_{\mathsf{dr}}, \mathsf{Sign}_{\mathsf{dr}}, \mathsf{Verify}_{\mathsf{dr}})$ derived as described above is not $\mathsf{frEUF\text{-}CMA}$-secure for any type of fault resilience.*

Proof. We show that $\mathcal{S}_{\mathsf{dr}}$ is not $\mathsf{frEUF\text{-}CMA}$-secure under the weakest form of fault attacks, namely (transient or persistent) 1-random faults (i.e., $N = 1$). This immediately also establishes the result under N-random faults with $N > 1$ and, through Lemmas 3.1 and 3.2, under differential and full faults.

The adversary $\mathcal{A}$ begins by calling the $\mathcal{O}_{\mathsf{Sign}}$ oracle on message $m_0 = 0^\lambda$. For the resulting two callbacks $\langle m \rangle$ on m (in lines 1, resp. 2, of the $\mathsf{Sign}_{\mathsf{dr}}$ algorithm) the adversary returns 0, i.e., introduces no faults. It obtains the resulting signature σ_0 (generated using some nonce r) which is valid for m_0.

The adversary then calls $\mathcal{O}_{\mathsf{Sign}}$ on message $m_0 = 0^\lambda$ again, this time returning 0 on the first callback to leave the message unchanged, but 1 on the second callback (line 2 of $\mathsf{Sign}_{\mathsf{dr}}$) to flip a message bit at a random position. This call results in a signature σ_1 generated using the same nonce r as in the first call which is valid on m_1, where by m_1 we denote the message value resulting from the single-bit random fault introduced through the second callback.

The adversary finally iterates over $i \in \{1, \ldots, \lambda\}$ to find the flipped bit position in m_1 (i.e., the single 1-entry in m_1) by invoking the $\mathsf{Verify}_{\mathsf{dr}}$ algorithm on an λ-bit message with the i-th bit set to 1, together with σ_1. As the underlying signature scheme $\mathcal{S}$ is forgeable under nonce repetition and $m_0 \neq m_1$, $\mathcal{A}$ can now use (m_0, σ_0) and (m_1, σ_1) to produce a valid $\mathsf{EUF\text{-}CMA}$ signature and win in the $\mathsf{frEUF\text{-}CMA}$ experiment. □

We note that similar attacks apply to other deterministic signature schemes such as RSA-FDH [BR96], showing that the additional property of uniqueness may not help to overcome fault attacks. If we describe the FDH scheme as a two-stage process $\llcorner h \lrcorner \leftarrow \mathsf{H}(\langle m \rangle)$, $\sigma \leftarrow \mathsf{Sign}(sk, \langle h \rangle)$, then the adversary can compute a hash value h^* of some message m^*, then call the signing oracle about some other message m, overwriting $\langle h \rangle$ with h^* in the signing process to get a signature for m^*. Even in case of a hash collision $h = h^*$ only m would be considered as used up, such that $\mathcal{A}$ would win the fault-resistance game. This works for full and differential faults but is unknown to work for random faults.

4.3 Combining Randomization and De-randomization

In seeking to overcome security failures due to weak randomness sources, de-randomized signature schemes forgo using any ephemeral randomness in the signing process. As discussed before, fault attacks can

instead of m. Due to the usually larger size of m, facilitating bit flips in m through row-hammer attacks, we focus on faulting m, but note that similar results apply for faulting r.

however revive these security failures by introducing nonce repetitions in the signing process. To insulate a signature scheme against both weak randomness and fault attacks—or, viewed differently, the de-randomization of a randomized signature scheme against fault attacks—, it is hence advisable to follow an approach that combines ephemeral randomness and de-randomization techniques. The agreed-upon only countermeasure effective against the previously described fault attacks [PSS+18, RP17, ABF+18, SBB+18] is to use an additional randomness value in the per-message nonce derivation. This is in support of the XEdDSA signature scheme design [Per16] deployed in the Signal protocol [Sig] for secure messaging, which combines deterministically generating a per-message nonce with an additional random value in order to derive the randomness used in the signing process.

We capture this combiner approach again through a generalized, abstract signature scheme $\mathcal{S}_{\mathsf{c}}$. The scheme $\mathcal{S}_{\mathsf{c}} = (\mathsf{KGen}_{\mathsf{c}}, \mathsf{Sign}_{\mathsf{c}}, \mathsf{Verify}_{\mathsf{c}})$ is based on a randomized signature scheme $\mathcal{S} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ for which it generates the randomness needed in $\mathcal{S}$'s signing algorithm in two steps: First, it samples an ephemeral random value r' (e.g., in the case of XEdDSA, r' is sampled as a random 512-bit string). Then, r' together with the signing key and input message enters a cryptographic hash function $\mathsf{H}\colon \{0,1\}^* \to \{0,1\}^{\geq\lambda}$ (again modeled as a random oracle) in order to derive the signing randomness r.[6] Key generation and verification are as for the randomized scheme, the modified signing algorithm $\mathsf{Sign}_{\mathsf{c}}$ is defined as follows:

$\underline{\mathsf{Sign}_{\mathsf{c}}(sk, \llcorner m \lrcorner)\text{:}}$
1 $\llcorner r' \lrcorner \xleftarrow{\$} \{0,1\}^{\lambda}$
2 $\llcorner r \lrcorner \leftarrow \mathsf{H}(sk, \langle m \rangle, \langle r' \rangle)$
3 $\sigma \leftarrow \mathsf{Sign}(sk, \langle m \rangle; \langle r \rangle)$
4 return σ

The definition of $\mathsf{Sign}_{\mathsf{c}}$ is accordingly annotated to capture fault attacks. This time, we consider faults not only for message m but also in the randomness variables r' and r. Note that the $\mathsf{Sign}_{\mathsf{c}}$ algorithm can furthermore be seen to tolerate (transient) faults in the secret signing key sk when used in the derivation of randomness through H; yet considering fault attacks on sk also in the signing process will require signature schemes secure against related-key attacks [BK04, GLM+04, BCM11], whose fault-resilience treatment we leave as an avenue for future work.

We now establish that the combiner countermeasure captured in $\mathcal{S}_{\mathsf{c}}$ indeed provides security against either weak randomness sources or (differential) fault attacks. We do so by showing that the approach lifts EUF-CMA security of the underlying signature scheme to fault-resilient unforgeability frEUF-CMA for $\mathcal{S}_{\mathsf{c}}$, when H is modeled as a random oracle. Note that the security statement is closely linked to the description of the scheme: We move from a purely functional description of the signature scheme to a high-level procedural representation in which the adversary can now interfere with sub steps. Such an *algorithmic implementation* still treats some steps as atomic (or, monolithic) procedures in which the adversary can only tamper with the input, but not interact with intermediate steps. Examples of such atomic steps are basic operations like assignments but may also refer to cryptographic procedures. For instance, $\mathsf{Sign}_{\mathsf{c}}(sk, \llcorner m \lrcorner)$ treats the hashing with H and signing with the original signing algorithm Sign as atomic operations. One can thus view the algorithmic implementation as determining points in executions in which attacks can modify variables.

We make use of our strongest full fault attack type in order to capture that weak randomness samples r' may be fully controlled by the adversary. Let us stress that this first part of the result—full fault resilience in r'—is not meant as establishing resilience against strong faults targeted (only) at r', but really constitutes a baseline result showing that the combiner construction $\mathcal{S}_{\mathsf{c}}$ provides *at least* the security of $\mathcal{S}$ even if the added randomness r' is completely flawed. The second part then establishes differential-fault resilience—for any number w of faulted bits—if r' is indeed random.

[6] Note that we treat the underlying (randomized) signature scheme $\mathcal{S}$ as well as the hash function H in a black-box manner both for the positive fault resilience results here, as well as for the generic fault attacks on $\mathcal{S}_{\mathsf{dr}}$ before. Of course, studying the fault resilience of specific such constructions is a valuable target on its own, which we leave for future work.

A noteworthy fact in the proof is that it shows we can use the same secret key sk for the signing step and the hash evaluation, when assuming H behaves like a random oracle. Usually, the secret key consists of two (possibly pseudorandomly derived) portions, one used for signing and one in the hash evaluation. An example where the key splitting is done is the EdDSA signature algorithm [BDL+11]. Our proof, of course, could be adapted to capture this case as well.

Theorem 4.3. *Let $\mathcal{S}$ be a randomized* EUF-CMA*-secure signature scheme. Then, in the random oracle model, the algorithmic implementation of the combined signature scheme $\mathcal{S}_\mathsf{c} = (\mathsf{KGen}_\mathsf{c}, \mathsf{Sign}_\mathsf{c}, \mathsf{Verify}_\mathsf{c})$ given above is*

a) frEUF-CMA*-secure under full faults on variable r', with*

$$\mathsf{Adv}^{\mathsf{frEUF\text{-}CMA}}_{\mathcal{S}_\mathsf{c},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{S},\mathcal{A}'}(\lambda), \quad \textit{and}$$

b) frEUF-CMA*-secure under differential faults on variables m, r', and r, with*

$$\mathsf{Adv}^{\mathsf{frEUF\text{-}CMA}}_{\mathcal{S}_\mathsf{c},\mathcal{A}}(\lambda) \leq q_H \cdot q_S \cdot 2^{-\lambda} + \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{S},\mathcal{A}'}(\lambda),$$

for $\mathcal{A}'$ given in the proofs and q_H, q_S denoting the number of queries made to the random oracle and the signing oracle, respectively, by $\mathcal{A}$.

Let us stress again that the theorem refers to the actual algorithmic implementation of Sign_c, treating the underlying signature procedure Sign as atomic. There might still be fault attacks on this step if one fleshed out the algorithmic implementation of that signing procedure. But this would depend on the actual scheme and is not captured by our general theorem. Note that the de-randomized solution $\mathsf{Sign}_\mathsf{dr}$ in the previous section is indeed insecure even if the underlying scheme is atomic, as long as it breaks under nonce repetitions. In this sense the theorem here confirms that putting the randomness in the hashing helps.

Proof. We separately prove the two sub-cases.

Ad a). The first case models that r' is drawn from a weak randomness source. Here, the full-fault capabilities allow $\mathcal{A}$ to arbitrarily chose any value for r' through the callback in line 1 of the Sign_c algorithm, including repeating r' across different signatures. We will rely on the non-faultable secret key sk input to the hash function, unknown to the adversary, to establish that the derived value r (per message m) is indeed uniformly random as required. Since the message cannot be faulted in the case here, the adversary cannot win due to clash and we do not need to consider this attack option here.

To see the security in this case, we first exclude (by aborting the security experiment) the case that the adversary $\mathcal{A}$ ever queries the random oracle H on an input $(sk, \cdot, \cdot)$ including the scheme's secret key sk as the first component. This can reduce $\mathcal{A}$'s advantage $\mathsf{Adv}^{\mathsf{frEUF\text{-}CMA}}_{\mathcal{S}_\mathsf{c},\mathcal{A}}$ by at most the advantage of the following adversary $\mathcal{A}'$ against the EUF-CMA security of $\mathcal{S}$, which by assumption is negligible.

Adversary $\mathcal{A}'$ simulates $\mathsf{Expt}^{\mathsf{frEUF\text{-}CMA}}_{\mathcal{S}_\mathsf{c},\mathcal{A}}$ for $\mathcal{A}$, using its own signing oracle for computing the signature in line 3 of Sign_c as follows. At the outset of the experiment, $\mathcal{A}'$ initializes an empty list $\mathcal{L}$. Whenever Sign is to be invoked on some message m and randomness r in the simulation, $\mathcal{A}'$ first checks if $(m, r, \sigma) \in \mathcal{L}$ for some σ. If so, $\mathcal{A}'$ returns σ. Otherwise, $\mathcal{A}'$ invokes its signing oracle on m to obtain a signature σ, stores (m, r, σ) in a list $\mathcal{L}$, and returns σ. Furthermore, whenever $\mathcal{A}$ queries the random oracle H on some value $(x, \cdot, \cdot)$, adversary $\mathcal{A}'$ checks whether x equals the challenge secret key sk by computing $\sigma \leftarrow \mathsf{Sign}(x, m^*; r^*)$ for a fresh message and randomness m^*, r^* and checking whether $\mathsf{Verify}(pk, m^*, \sigma^*) = 1$. If so, $\mathcal{A}'$ outputs (m^*, σ^*) as its forgery and stops. Otherwise, $\mathcal{A}$ returns a random value as the answer for

the hash query (but obeying consistency across queries). Eventually, $\mathcal{A}'$ outputs the forgery of $\mathcal{A}$ as its own forgery when $\mathcal{A}$ stops.

Whenever $\mathsf{Expt}^{\mathsf{frEUF\text{-}CMA}}_{\mathcal{S}_c,\mathcal{A}}$ would abort due to $\mathcal{A}$ querying sk to the random oracle, $\mathcal{A}'$ wins in the $\mathsf{Expt}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{S},\mathcal{A}'}$ experiment through its valid forgery (m^*, σ^*). The probability of the first event occurring is hence bounded by the (negligible) advantage of $\mathcal{A}'$ in the latter experiment.

Otherwise, whenever $\mathcal{A}$ does not query sk to the random oracle, r is derived as a uniformly random value per message m which is secret to $\mathcal{A}$ in each of its $\mathcal{O}_{\mathsf{Sign}}$ queries. Observe that, by construction, $\mathsf{Sign}_{\mathsf{c}}$ is deterministic when fixing r' (and thus r), which is taken into account in the reduction through $\mathcal{A}'$ keeping the list $\mathcal{L}$ of signatures for each (m, r) pair seen. Adversary $\mathcal{A}'$ hence provides a sound simulation of the non-aborting $\mathsf{Expt}^{\mathsf{frEUF\text{-}CMA}}_{\mathcal{S}_c,\mathcal{A}}$ when implicitly setting r to the internal randomness choice of its signature oracle. As the trial signature computation under candidate secret keys x do not involve the signing oracle of $\mathcal{A}'$, a valid forgery by $\mathcal{A}$ in $\mathsf{Expt}^{\mathsf{frEUF\text{-}CMA}}_{\mathcal{S}_c,\mathcal{A}}$ also constitutes a valid forgery by $\mathcal{A}'$ in $\mathsf{Expt}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{S},\mathcal{A}'}$. This again is bounding the advantage of $\mathcal{A}$ in the former by the (negligible) advantage of $\mathcal{A}'$ in the latter.

Ad b). The second case models strong differential fault attacks (like rowhammer). This time, the adversary is allowed to inject arbitrary bit flips in the message variable m as well as the internal randomness variables r' and r. We will rely on the randomness of r' persisting through bit flips in r', the random oracle derivation, and the resulting r to establish that the derived value r is still uniformly random.

Consider the reduction $\mathcal{A}'$ of a successful $\mathcal{A}$ in $\mathsf{Expt}^{\mathsf{frEUF\text{-}CMA}}_{\mathcal{S}_c,\mathcal{A}}$ to the $\mathsf{EUF\text{-}CMA}$ security of $\mathcal{S}$, which simulates $\mathsf{Expt}^{\mathsf{frEUF\text{-}CMA}}_{\mathcal{S}_c,\mathcal{A}}$ by simply invoking its own $\mathcal{O}_{\mathsf{Sign}}$ oracle to compute the signature in line 3 of $\mathsf{Sign}_{\mathsf{c}}$. When $\mathcal{A}$ outputs its forgery, $\mathcal{A}'$ outputs the same forgery in its experiment $\mathsf{Expt}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{S},\mathcal{A}'}$.

We need to argue that the simulation provided to $\mathcal{A}$ is sound. In particular, this requires that the potentially faulted values r used to invoke the signing oracle $\mathcal{O}_{\mathsf{Sign}}$ are indeed uniformly random and secret to the adversary for each call as required for the $\mathsf{EUF\text{-}CMA}$ security of $\mathcal{S}$. To this end, let us trace the randomness used by $\mathcal{A}'$ in any invocation of $\mathsf{Sign}_{\mathsf{c}}$, originating from sampling r' to submitting (faulted) value r to the $\mathcal{O}_{\mathsf{Sign}}$ oracle.

- In line 1 of $\mathsf{Sign}_{\mathsf{c}}$, the value r' is sampled uniformly at random (and hidden from $\mathcal{A}$).
- In line 2, $\mathcal{A}$ is first invoked through the callback $\langle r' \rangle$ on r' and returns some difference value Δ_0. The callback returns the value $r'_{\Delta_0} = r' \oplus \Delta_0$ to be used in the hash function computation, which is still uniformly random distributed and unknown to $\mathcal{A}$ as r' was.

  Since H is a random oracle, the resulting value r is again uniformly random. Furthermore, the probability that $\mathcal{A}$ guesses r'_{Δ_0} in a query to the random oracle H is at most $2^{-\lambda}$, so r remains unknown to $\mathcal{A}$ with all but negligible probability over all random oracle queries. Note that we do not rely on the secrecy (nor integrity) of sk in this step since the unknown r'_{Δ_0} acts as an ephemeral key here.
- In line 3, $\mathcal{A}$ may again inject a differential fault Δ_1, this time on r. For the same reason as above, the resulting value $r_{\Delta_1} = r \oplus \Delta_1$ stays uniformly distributed and unknown to $\mathcal{A}$.

Using the faulted value r_{Δ_1} of r as the input to the $\mathcal{O}_{\mathsf{Sign}}$ oracle by $\mathcal{A}'$ is hence sound. Thus, if $\mathcal{A}$ wins in the original attack, either via a forgery or via a clash, then this also holds in the simulated attack with the (randomized) signing algorithm. For forgeries of fresh messages, the (negligible) advantage of $\mathcal{A}'$ against the $\mathsf{EUF\text{-}CMA}$ of $\mathcal{S}$ bounds the $\mathsf{frEUF\text{-}CMA}$ advantage of $\mathcal{A}$ against $\mathcal{S}_c$, as desired.

Finally, we have to account for $\mathcal{A}$ winning through a potential clash during the (now probabilistic) signing step. In each query there are at most two messages appearing during the signing process, the first one $m^{(1)}$ in the computation of $\llcorner r \lrcorner \leftarrow \mathsf{H}(sk, \langle m \rangle, \langle r' \rangle)$ in Line 2, the second one $m^{(2)}$ in the computation of

$\sigma \leftarrow \mathsf{Sign}(sk, \langle m \rangle; \langle r \rangle)$ in Line 3. The second one certainly verifies with σ under the public key. Now, if $\mathcal{A}$ triggers a clash, both messages must be included in the set $m^{\mathsf{valid}}_{\mathsf{Sign}}$. This means that the first message $m^{(1)}$, too, needs to verify, be different from the second one, and must not have been included in Q by any prior $\mathcal{O}_{\mathsf{Sign}}$ query. Hence, when detecting a clash, $\mathcal{A}'$ can immediately output $m^{(1)}$ together with σ as its own forgery. That forgery is valid, as $m^{(1)}$ was never asked to the signing oracle of $\mathcal{A}'$ before. Hence, the probability for this attack option of $\mathcal{A}$ to succeed can also be bounded by the EUF-CMA security of $\mathcal{S}$. □

4.3.1 An XOR Variant

For completeness, let us note that a variant of the combiner scheme $\mathcal{S}_{\mathsf{c}}$ above that merges the additional randomness via an XOR instead of including it under the hash function evaluation achieves similar security results. More precisely, such a variant $\mathcal{S}_{\oplus} = (\mathsf{KGen}_{\oplus}, \mathsf{Sign}_{\oplus}, \mathsf{Verify}_{\oplus})$ is defined as the $\mathcal{S}_{\mathsf{c}}$ scheme (for a hash function H with output length $\ell(\lambda)$), except for $\mathsf{Sign}_{\oplus}$ operating as follows:

$\underline{\mathsf{Sign}_{\oplus}(sk, \llcorner m \lrcorner):}$
1 $\llcorner r_0 \lrcorner \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}$
2 $\llcorner r_1 \lrcorner \leftarrow \mathsf{H}(sk, \langle m \rangle)$
3 $\llcorner r \lrcorner \leftarrow \langle r_0 \rangle \oplus \langle r_1 \rangle$
4 $\sigma \leftarrow \mathsf{Sign}(sk, \langle m \rangle; \langle r \rangle)$
5 return σ

Following the proof arguments for Theorem 4.3, uniform randomness and secrecy from $\mathcal{A}$ of either value r_0 or r_1 propagates through the XOR operation in line 3 of $\mathsf{Sign}_{\oplus}$, even under differential fault attacks. Hence, the $\mathcal{S}_{\oplus}$ variant achieves the same security guarantees as described in Theorem 4.3 for $\mathcal{S}_{\mathsf{c}}$.

5 Fault-Resilient Authenticated Encryption

We now turn to studying the effects of fault attacks on authenticated encryption schemes and how to enable fault resilience in this setting. In an effort to obviate the need for strong randomness in the encryption process, the understanding of modern authenticated encryption switched to a nonce-based syntax, in which a non-repeating nonce value enters encryption in replacement of fresh per-message randomness. Regularly, authenticated encryption schemes then indeed rely on the nonce not to repeat and generally do not uphold any security guarantees if this condition is violated. A prominent example is the widely adopted Galois/Counter mode (GCM) [Dwo07], combined, e.g., with the AES block cipher. While being secure as an authenticated encryption scheme [MV04], authentication guarantees are immediately lost in case of nonce repetitions [Jou06].

A strengthened security notion introduced by Rogaway and Shrimpton [RS06] augments authenticated-encryption security with resistance against *nonce misuse*: it demands that security is upheld even if nonces repeat, such that an adversary may only learn when a full triple (N, A, m) of nonce, associated data, and message is repeated, but ciphertexts otherwise look random. Our basic security definitions for authenticated encryption in Definition 2.4 capture nonce-misuse resistance when the adversary is allowed to repeat nonces.Since its introduction, nonce-misuse resistance has become a design target for authenticated encryption schemes, put forth, e.g., in the CAESAR competition for authenticated encryption ciphers [CAE].

5.1 Fault-Resilient Security of Authenticated Encryption

In order to study the effects of fault attacks on authenticated encryption schemes based on our generic model, we first lift the security notions for authenticated encryption (cf. Definition 2.4) to the fault resilience setting. We focus on faults in the encryption process here, as it is encryption where different variants

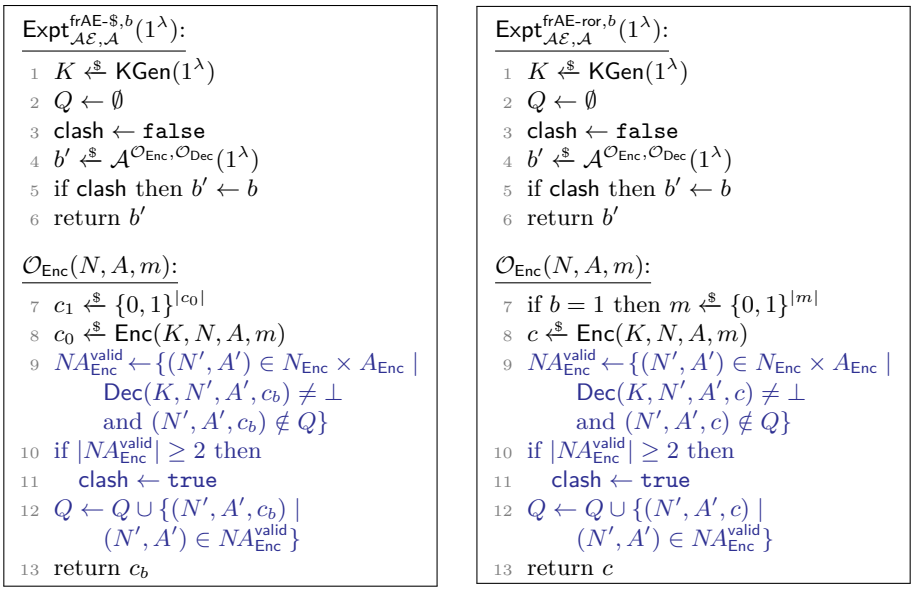

Figure 6: Security experiments for *fault-resilient* authenticated encryption schemes. Lines 9–12 in $\mathcal{O}_{\mathsf{Enc}}$ are changed compared to the classical notions in Figure 2; the definition of $\mathcal{O}_{\mathsf{Dec}}$ in both experiments is as for the classical notions in Figure 2. Recall that N_{Enc} and A_{Enc} are the set of values the nonce, resp. AD, variable N, resp. A, took during the encryption process in line 8 due to fault callbacks.

for avoiding ephemeral randomness and nonce glitches are implemented. Our notions can however be extended to also consider faults attacks on the decryption process.

As the major change from regular security definitions, we need to define how to rule out trivial queries decrypting the response of an encryption query. We do so analogously to the signature setting described in Section 4.1, namely by considering, through a list $NA^{\mathsf{valid}}_{\mathsf{Enc}}$, the new combinations of (N,A,c) which decrypt successfully, taking candidate values for N and A from N_{Enc}, resp. A_{Enc}, the lists of values taken by N, resp. A, within Enc. Intuitively, these are the new tuples which the encryption algorithm can be considered to have produced. If there is just one such combination this gets added to Q as the single resulting challenge ciphertext to be prohibited for the decryption oracle. If there are however multiple combinations, we declare the adversary to win by setting the clash flag. One can think of this saying that the adversary has managed to produce multiple (valid) encryption tuples from a single, faulted encryption call. We again then declare the adversary to win immediately.

We again consider both randomness and real-or-random indistinguishability under fault attacks, with the latter being weaker than the former.

Definition 5.1 (Fault-resilient security of authenticated encryption)**.** *Let $\mathcal{AE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ be an authenticated encryption scheme and experiments $\mathsf{Expt}^{\mathsf{frAE\text{-}\$},b}_{\mathcal{AE},\mathcal{A}}$ and $\mathsf{Expt}^{\mathsf{frAE\text{-}ror},b}_{\mathcal{AE},\mathcal{A}}$ for an adversary $\mathcal{A}$ and a bit b be defined as in Figure 6. We restrict $\mathcal{A}$ to ask any query (N,A,m) to $\mathcal{O}_{\mathsf{Enc}}$ at most once.*

We say that $\mathcal{AE}$ is AE-$-secure with fault resilience*, resp.* AE-ror-secure with fault resilience*, if for all PPT adversaries and* AE-SEC = AE-$*, resp.* AE-SEC = AE-ror*, the following advantage function is negligible in the security parameter:*

$$\mathsf{Adv}^{\mathsf{frAE\text{-}SEC}}_{\mathcal{AE},\mathcal{A}} := \left|\Pr\left[\mathsf{Expt}^{\mathsf{frAE\text{-}SEC},0}_{\mathcal{AE},\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{frAE\text{-}SEC},1}_{\mathcal{AE},\mathcal{A}}(1^\lambda) = 1\right]\right|.$$

When $\mathcal{A}$ never repeats the nonce value N between any two $\mathcal{O}_{\mathsf{Enc}}$ calls, we call it nonce-respecting*; otherwise we say the scheme is* nonce-misuse resistant*.*

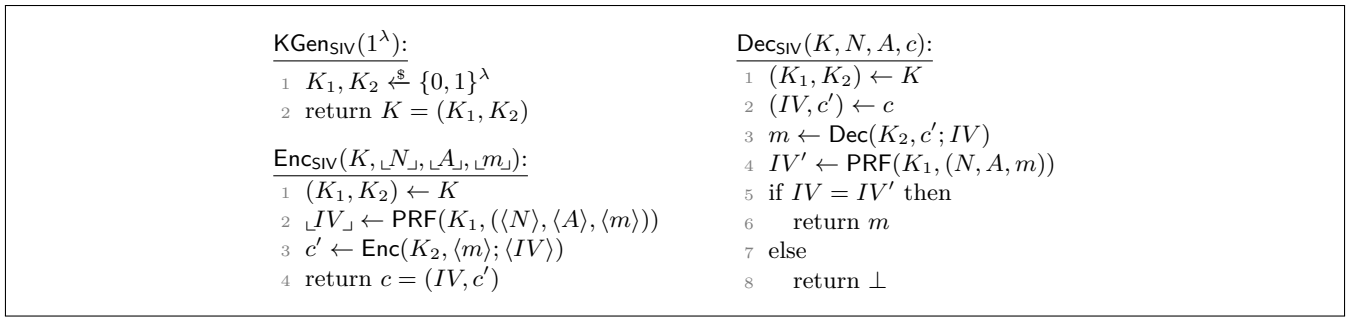

Figure 7: The synthetic initialization vector (SIV) mode of operation based on a pseudorandom function PRF and an IV-based encryption scheme $\mathcal{E}$.

5.2 SIV Is Not Fault-Resilient

As an example for a nonce-misuse resistant authenticated encryption scheme, we will study the SIV (for "synthetic initialization vector") mode of operation introduced by Rogaway and Shrimpton [RS06]. It achieves classical, misuse-resistant randomness indistinguishability AE-$ by combining a pseudorandom function and an IND$-CPA-secure IV-based encryption scheme [RS06]. SIV was also considered for generic composition in a work together with Namprempre [NRS14] and optimized through combination with GCM by Gueron and Lindell [GL15].

SIV is defined as in Figure 7 based on a pseudorandom function $\mathsf{PRF}\colon \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^\lambda$ and a conventional IV-based encryption scheme $\mathcal{E} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ with initialization vectors from $\{0,1\}^\lambda$. We write the IV-based encryption algorithm Enc as $c \leftarrow \mathsf{Enc}(K, m; IV)$ for encrypting a message m under key K and initialization vector IV into a ciphertext c. Analogously, we write IV-based decryption as $m \leftarrow \mathsf{Dec}(K, c; IV)$ for decrypting a ciphertext c under key K and initialization vector IV into a message m.

In our definition of SIV, we consider potential fault attacks on the nonce N, associated data A, message m, and synthetic initialization vector IV within the encryption algorithm (cf. the according annotation in Figure 7). Our following result shows that SIV does not achieve fault-resilient security, even in the weaker AE-ror sense. More specifically, assuming pseudorandomness of the deployed PRF, AE-ror security of SIV breaks under (transient or persistent) single-bit random faults (i.e., the weakest form of fault attacks in our model) on either of the adversarially-provided values N, A, or m for encryption. As for AE-$ security, it is easy to see that faults can induce collisions in the IV computation, which then are easy to distinguish from randomly sampled values.

Theorem 5.2. *Let* PRF *be a pseudorandom function. Then the SIV authenticated encryption mode* $\mathcal{AE}_{\mathsf{SIV}} = (\mathsf{KGen}_{\mathsf{SIV}}, \mathsf{Enc}_{\mathsf{SIV}}, \mathsf{Dec}_{\mathsf{SIV}})$ *from Figure 7 is not* frAE-ror*-secure against any type of faults on the encryption inputs* N*,* A*, and* m*.*

Proof. We show insecurity against the weakest type of (transient or persistent) 1-random faults in the nonce input N. Insecurity under stronger fault attacks follows by implication, and the cases for associated data A and message m are analogous.

First of all, it is convenient to replace the pseudorandom function PRF deployed in $\mathcal{AE}_{\mathsf{SIV}}$ in $\mathsf{Expt}^{\mathsf{frAE\text{-}ror},b}_{\mathcal{AE}_{\mathsf{SIV}},\mathcal{A}}$ by a truly random function. The advantage difference of any adversary $\mathcal{A}$ introduced by this step can be upper bounded by the advantage in breaking the pseudorandomness of PRF.

Now, consider an adversary $\mathcal{A}$ that first queries $\mathcal{O}_{\mathsf{Enc}}$ on zero-valued nonce, associated data, and message $(N_0, A, m) = (0^\lambda, 0^\lambda, 0^\lambda)$. When queried for the faulting callback $\langle N \rangle$ in line 2, then $\mathcal{A}$ replies with 0 (no fault). It hence obtains a ciphertext output $c_0 = (IV_0, c'_0)$ which consists of the outputs of

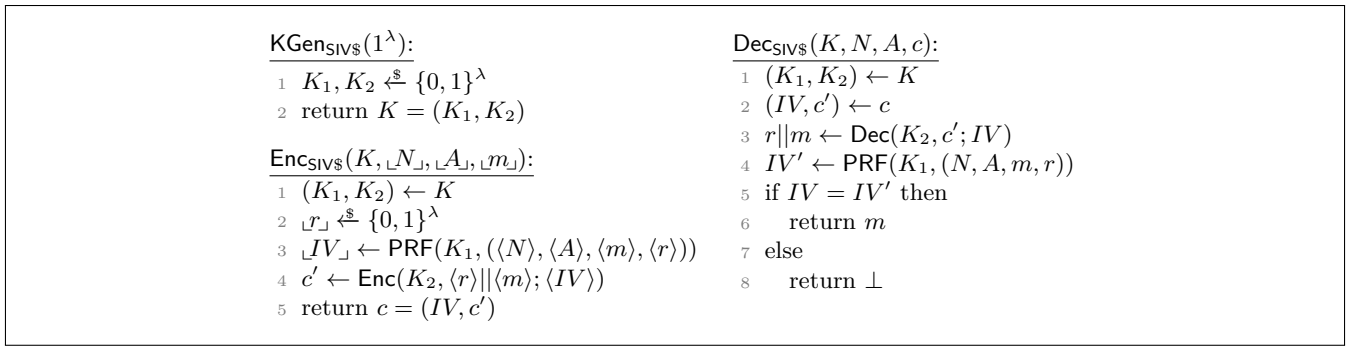

Figure 8: The randomness-augmented synthetic initialization vector mode SIV$ based on a pseudorandom function PRF and an IV-based encryption scheme $\mathcal{E}$.

the now random function PRF and of the encryption scheme on either the real message m if $b = 0$, i.e., $IV_0 = \mathsf{PRF}(K_1, (0^\lambda, 0^\lambda, 0^\lambda))$ and $c'_0 = \mathsf{Enc}(K_2, 0^\lambda; IV_0)$, or the outputs on a random message $m' \xleftarrow{\$} \{0,1\}^\lambda$ if $b = 1$, i.e., $IV_0 = \mathsf{PRF}(K_1, (0^\lambda, 0^\lambda, m'))$ and $c'_0 = \mathsf{Enc}(K_2, m'; IV_0)$.

As the second step, $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Enc}}$ on a nonce with one leading 1-bit, and zero-valued associated data and message $(N_1, A, m) = (10^{\lambda-1}, 0^\lambda, 0^\lambda)$. In response to the callback $\langle N\rangle$, $\mathcal{A}$ now replies with 1, which leads to one random bit being flipped in N_1 resulting in some value N'_1. The resulting ciphertext $c_1 = (IV_1, c'_1)$ is thus computed on (N'_1, A, m) if $b = 0$, or on (N'_1, A, m'') for some random $m'' \xleftarrow{\$} \{0,1\}^\lambda$ if $b = 1$.

The adversary $\mathcal{A}$ finally checks if $IV_0 = IV_1$ and outputs 0 if this is the case, otherwise 1. If indeed $b = 0$, then with (non-negligible) probability $1/\lambda$ the first bit of N_1 is flipped through the induced fault and thus $N_0 = N'_1$, resulting in $IV_0 = IV_1$ due to all inputs to g being equal. If however $b = 1$, then the inputs to PRF are distinct and $IV_0 = IV_1$ happens only with (negligible) probability $2^{-\lambda+1}$ when either $m' = m''$ or the distinct inputs collide under PRF. In summary, $\mathcal{A}$ hence distinguishes the two cases with non-negligible probability and thus breaks the frAE-ror security of $\mathcal{AE}_{\mathsf{SIV}}$ under 1-random faults in the nonce input N. □

5.3 SIV$: Randomness-augmented SIV

In order to overcome SIV's vulnerability to fault attacks in the encryption inputs, we propose and discuss an approach of augmenting the encryption process with ephemeral randomness in order to protect against faults. This approach translates concepts employed in the setting of signature schemes (e.g., in the XEdDSA scheme [Per16], cf. Section 4.3) to the realm of authenticated encryption which, to the best of our knowledge, have not been previously considered in this setting before.

Observe that the reason for SIV falling short of protecting against fault attacks is that such attacks can force the synthetic IV value to collide for different inputs (N, A, m) of nonce, associated data, and message. This resembles the setting for de-randomized deterministic signatures, where fault attacks may lead to the random per-message nonce being repeated. We show that an analogous combiner approach to derive the synthetic IV from both the values N, A, and m as well as an *additional ephemeral random* input provides strong combined security against either weak randomness sources or fault attacks.

We denote the randomness-augmented synthetic initialization vector mode as SIV$, described in Figure 8. Like SIV, the scheme $\mathsf{SIV\$} = (\mathsf{KGen}_{\mathsf{SIV\$}}, \mathsf{Enc}_{\mathsf{SIV\$}}, \mathsf{Dec}_{\mathsf{SIV\$}})$ is based on a pseudorandom function $\mathsf{PRF}\colon \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^\lambda$ and an IV-based encryption scheme $\mathcal{E} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$. In contrast to SIV, the encryption operation of SIV$ is now randomized. Prudently including the ephemeral randomness value r as a λ-bit prefix to the encrypted message, we ensure that SIV$ maintains the same

outer ciphertext format as SIV, including its strong randomness indistinguishability.[7] The ciphertext size increases by one block.

As we show next, SIV$ indeed protects against (either) weak randomness sources (modeled as full-fault attacks on the ephemeral randomness r)[8] or strong differential fault attacks (for any number w of faulted bits) on all adversarial encryption inputs N, A, and m as well as the internal randomness r and synthetic initialization vector IV. Under the same assumptions needed to establish regular security for SIV [RS06], namely PRF being a pseudorandom function and $\mathcal{E}$ being IND$-CPA-secure, we show that SIV$ upholds strong randomness indistinguishability (frAE-$) under such faults. Again, considering fault attacks also on the PRF and encryption keys requires schemes secure against related-key attacks [BK04, GLM+04, BC10] and is left for future work.

Theorem 5.3. *Let* PRF *be a pseudorandom function and* $\mathcal{E}$ *an* IND$-CPA*-secure IV-based encryption scheme. Then the algorithmic implementation of the randomness-augmented SIV mode* $\mathsf{SIV\$} = (\mathsf{KGen}_{\mathsf{SIV\$}}, \mathsf{Enc}_{\mathsf{SIV\$}}, \mathsf{Dec}_{\mathsf{SIV\$}})$ *from Figure 8 is, in a nonce-misuse resistant manner,*

a) frAE-$*-secure under full faults on variable* r*, with*

$$\mathsf{Adv}^{\mathsf{frAE\text{-}\$}}_{\mathsf{SIV\$},\mathcal{A}}(\lambda) \leq 2 \cdot \left(\mathsf{Adv}^{\mathsf{PRF\text{-}sec}}_{\mathsf{PRF},\mathcal{A}'}(\lambda) + q_D \cdot 2^{-\lambda} + \mathsf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathcal{E},\mathcal{A}''}(\lambda)\right), \quad \textit{and}$$

b) frAE-$*-secure under differential faults on all of the variables* N*,* A*,* m*,* r*, and* IV*, with*

$$\mathsf{Adv}^{\mathsf{frAE\text{-}\$}}_{\mathsf{SIV\$},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{PRF\text{-}sec}}_{\mathsf{PRF},\mathcal{A}'}(\lambda) + \mathsf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathcal{E},\mathcal{A}''}(\lambda),$$

for $\mathcal{A}'$*,* $\mathcal{A}''$ *given in the proofs and* q_D *denoting the number of queries made to the decryption oracle by* $\mathcal{A}$*.*

Proof. We separately consider the two sub-cases.

Ad a). The first case models weak randomness sources through full faults in the ephemeral randomness value r. Note that this means that the data N and A are not faulted and are thus unique for each encryption call. We therefore do not need to take into account successful attacks due to clash.

Consider an adversary $\mathcal{A}$ in the experiment $\mathsf{Expt}^{\mathsf{frAE\text{-}\$},b}_{\mathsf{SIV\$},\mathcal{A}}$. We modify the experiment into $\mathsf{Expt}^{\mathsf{frAE\text{-}\$',b}}_{\mathsf{SIV\$},\mathcal{A}}$ by replacing the pseudorandom function PRF with a truly random function g. Observe that any advantage difference for $\mathcal{A}$ (for either value of b) introduced by this step can be upper bounded by the distinguishing advantage of a reduction $\mathcal{A}'$ to the pseudorandom function security of PRF, which by assumption is negligible.

Next recall that the tuple (N, A, m) is unique for each $\mathcal{O}_{\mathsf{Enc}}$ oracle call of $\mathcal{A}$ and that none of its components can be faulted in this sub-case. Hence, in each $\mathcal{O}_{\mathsf{Enc}}$ oracle invocation in $\mathsf{Expt}^{\mathsf{frAE\text{-}\$',b}}_{\mathsf{SIV\$},\mathcal{A}}$, the truly random function g yields an independent and uniformly random value IV, which is unpredictable from $\mathcal{A}$'s perspective. In particular, analogous to the security analysis of the original SIV mode [RS06], the decryption oracle $\mathcal{O}_{\mathsf{Dec}}$ using the truly random function g produces a non-error output only with negligible probability of $2^{-}\lambda$ for each decryption oracle call.

We can hence in a second step alter the experiment into $\mathsf{Expt}^{\mathsf{frAE\text{-}\$'',b}}_{\mathsf{SIV\$},\mathcal{A}}$, where the ciphertext output c' of the encryption step in the $\mathcal{O}_{\mathsf{Enc}}$ oracle is replaced by a uniformly random value $\tilde{c}' \xleftarrow{\$} \{0,1\}^{|c'|}$. Independent of b, any advantage difference for $\mathcal{A}$ introduced by this change can be bounded by the (negligible) advantage

[7] Alternatively, one may include r as additional component in the ciphertext. This however degrades security to real-or-random indistinguishability in case of weak randomness values r.

[8] Analogous to the signature case in Theorem 4.3, the first part of the statement again only serves as a baseline result. It shows that SIV$ provides at least the security of SIV even if the added randomness r' is completely flawed.

of a reduction $\mathcal{A}''$ in breaking the IND$-CPA security of the encryption scheme $\mathcal{E}$. As we are now ensured to be using independent and unpredictable random IV values, $\mathcal{A}''$ can freely replace (IV, c') with the output of the IND$-CPA encryption oracle.

Observe that $\mathcal{O}_{\mathsf{Enc}}$ in $\mathsf{Expt}^{\text{frAE-\$}'',b}_{\text{SIV\$},\mathcal{A}}$ is independent of b. As the modifications apply for either value of b, $\mathcal{A}$'s advantage $\mathsf{Adv}^{\text{frAE-\$}}_{\mathcal{AE},\mathcal{A}}$ in the original frAE-$ experiment is upper bounded by twice the advantage against breaking pseudorandomness of PRF or IND$-CPA security of $\mathcal{E}$, which by assumption is negligible.

Ad b). In the second case we are concerned with an adversary exercising differential fault attacks (for difference values of arbitrary Hamming weight) on variables N, A, m, r, and IV. Since the scheme accesses N and A only once, we do not need to consider event clash; it simply cannot occur.

We begin by considering an adversary $\mathcal{A}$ in the experiment $\mathsf{Expt}^{\text{frAE-\$},0}_{\text{SIV\$},\mathcal{A}}$ (i.e., for bit $b = 0$). First of all, observe that r is initialized as random value unknown to $\mathcal{A}$ which persists through the differential-fault callback $\langle r \rangle$ in line 3 of the SIV$ scheme.

As in Case a), we now first replace PRF with a truly random function g, bounding the advantage difference by the PRF security advantange of a reduction $\mathcal{A}'$ and yielding a modified experiment $\mathsf{Expt}^{\text{frAE-\$}'}_{\text{SIV\$},\mathcal{A}}$. For r being a uniformly random input to g, the derived initialization vector IV is also uniformly random and unpredictable for $\mathcal{A}$, a property which persists through the fault callback $\langle IV \rangle$ in line 4.

Therefore, we can as in Case a) modify the experiment into $\mathsf{Expt}^{\text{frAE-\$}''}_{\text{SIV\$},\mathcal{A}}$ by replacing the ciphertext c' in the $\mathcal{O}_{\mathsf{Enc}}$ oracle by a uniformly random value $\tilde{c}' \xleftarrow{\$} \{0,1\}^{|c'|}$. This modification can again be bounded by the (negligible) advantage of a reduction $\mathcal{A}''$ in breaking IND$-CPA security of $\mathcal{E}$, where $\mathcal{A}''$ queries $\mathcal{A}$'s response to the callbacks $\langle r \rangle$ and $\langle m \rangle$ in line 4 to its encryption oracle.

As a result, in $\mathsf{Expt}^{\text{frAE-\$}''}_{\text{SIV\$},\mathcal{A}}$ now r, IV, and $\tilde{c}'$ in each $\mathcal{O}_{\mathsf{Enc}}$ call are all independent and uniformly random and thus $\mathsf{Expt}^{\text{frAE-\$}''}_{\text{SIV\$},\mathcal{A}} = \mathsf{Expt}^{\text{frAE-\$},1}_{\text{SIV\$},\mathcal{A}}$. Therefore, $\mathcal{A}$'s advantage $\mathsf{Adv}^{\text{frAE-\$}}_{\text{SIV\$},\mathcal{A}}$ is negligible. □

6 Conclusion

We introduced a game-based treatment of cryptographic fault resilience which enables generic extensions of existing security notions to capture memory fault attacks. Our model exemplifies how different attack types can be captured through a hierarchy of callback-style adversarial interactions within accordingly augmented security notion. Applying our modeling technique to deterministic signature schemes, we revisit known fault attacks on deterministic signature schemes. Moreover, we can, for the first time, give provable security guarantees for proposed countermeasures in the realm of signatures and translate both attacks and provably-secure countermeasures to the setting of nonce-misuse resistant authenticated encryption.

Potential future research questions arise both in modeling and applications. Applying the modeling of fault resilience to other security notions possibly yields new insights into fault attacks and protection for other cryptographic primitives. Security against related-key attacks targeting partial effects of memory faults lends itself to be a viable building block here. Another worthwhile effort is to look beyond our strict monolithic treatment of the cryptographic primitives and investigate in how far the structure of the primitive, say, iterative hashing as in SHA-2 or SHA-3, affects memory fault attacks. Of course, such a treatment could be performed all the way down to the lower implementation level. Finally, while our modeling provides a general way to capture memory faults, capturing control-flow fault attacks in a meaningful way for game-based, cryptographic security notions remains a challenging open problem.

Acknowledgments

We thank the anonymous reviewers for valuable comments. Felix Günther is supported in part by Research Fellowship grant GU 1859/1-1 of the German Research Foundation (DFG) and National Science Foundation (NSF) grants CNS-1526801 and CNS-1717640. This work has been co-funded by the DFG as part of project P2 within the CRC 1119 CROSSING. Most of the work on this paper was done while Felix Günther was at UC San Diego.

References

[ABF+18] Christopher Ambrose, Joppe W. Bos, Björn Fay, Marc Joye, Manfred Lochter, and Bruce Murray. Differential attacks on deterministic signatures. In Nigel P. Smart, editor, *Topics in Cryptology – CT-RSA 2018*, volume 10808 of *Lecture Notes in Computer Science*, pages 339–353. Springer, Heidelberg, April 2018. (Cited on pages 4, 5, 13, 14, 15, and 16.)

[AOTZ19] Diego F. Aranha, Claudio Orlandi, Akira Takahashi, and Greg Zaverucha. Security of hedged Fiat-Shamir signatures under fault attacks. Cryptology ePrint Archive, Report 2019/956, 2019. https://eprint.iacr.org/2019/956. (Cited on page 6.)

[BBKN12] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache. Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures. *Proceedings of the IEEE*, 100(11):3056–3076, Nov 2012. (Cited on pages 3 and 6.)

[BBN+09] Mihir Bellare, Zvika Brakerski, Moni Naor, Thomas Ristenpart, Gil Segev, Hovav Shacham, and Scott Yilek. Hedged public-key encryption: How to protect against bad randomness. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 232–249. Springer, Heidelberg, December 2009. (Cited on page 6.)

[BC10] Mihir Bellare and David Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 666–684. Springer, Heidelberg, August 2010. (Cited on page 23.)

[BCM11] Mihir Bellare, David Cash, and Rachel Miller. Cryptography secure against related-key attacks and tampering. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 486–503. Springer, Heidelberg, December 2011. (Cited on page 16.)

[BDF+14] Gilles Barthe, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Mehdi Tibouchi, and Jean-Christophe Zapalowicz. Making RSA-PSS provably secure against non-random faults. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems – CHES 2014*, volume 8731 of *Lecture Notes in Computer Science*, pages 206–222. Springer, Heidelberg, September 2014. (Cited on page 5.)

[BDJR97] Mihir Bellare, Anand Desai, Eric Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *38th Annual Symposium on Foundations of Computer Science*, pages 394–403. IEEE Computer Society Press, October 1997. (Cited on page 8.)

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In Walter Fumy, editor, *Advances in*

Cryptology – EUROCRYPT'97, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer, Heidelberg, May 1997. (Cited on pages 3 and 4.)

[BDL+11] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 124–142. Springer, Heidelberg, September / October 2011. (Cited on pages 3, 14, and 17.)

[BECN+06] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan. The sorcerer's apprentice guide to fault attacks. *Proceedings of the IEEE*, 94(2):370–382, Feb 2006. (Cited on pages 3 and 6.)

[BG15] Johannes Blömer and Peter Günther. Singular curve point decompression attack. In *2015 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 71–84, 2015. (Cited on page 6.)

[BGG94] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Incremental cryptography: The case of hashing and signing. In Yvo Desmedt, editor, *Advances in Cryptology – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 216–233. Springer, Heidelberg, August 1994. (Cited on page 6.)

[BGG95] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Incremental cryptography and application to virus protection. In *27th Annual ACM Symposium on Theory of Computing*, pages 45–56. ACM Press, May / June 1995. (Cited on page 6.)

[BH19] Joachim Breitner and Nadia Heninger. Biased nonce sense: Lattice attacks against weak ECDSA signatures in cryptocurrencies. In Ian Goldberg and Tyler Moore, editors, *FC 2019: 23rd International Conference on Financial Cryptography and Data Security*, volume 11598 of *Lecture Notes in Computer Science*, pages 3–20. Springer, Heidelberg, February 2019. (Cited on page 3.)

[BK04] Mihir Bellare and Tadayoshi Kohno. Hash function balance and its impact on birthday attacks. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 401–418. Springer, Heidelberg, May 2004. (Cited on pages 6, 16, and 23.)

[BMM00] Ingrid Biehl, Bernd Meyer, and Volker Müller. Differential fault attacks on elliptic curve cryptosystems. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 131–146. Springer, Heidelberg, August 2000. (Cited on page 6.)

[BP16] Alessandro Barenghi and Gerardo Pelosi. A note on fault attacks against deterministic signature schemes. In Kazuto Ogawa and Katsunari Yoshioka, editors, *IWSEC 16: 11th International Workshop on Security, Advances in Information and Computer Security*, volume 9836 of *Lecture Notes in Computer Science*, pages 182–192. Springer, Heidelberg, September 2016. (Cited on page 4.)

[BPR14] Mihir Bellare, Kenneth G. Paterson, and Phillip Rogaway. Security of symmetric encryption against mass surveillance. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 1–19. Springer, Heidelberg, August 2014. (Cited on page 6.)

[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: 1st Conference on Computer and Communications Security*, pages 62–73. ACM Press, November 1993. (Cited on pages 3 and 14.)

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In Ueli M. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer, Heidelberg, May 1996. (Cited on page 15.)

[BR18] Michael Brengel and Christian Rossow. Identifying key leakage of bitcoin users. In Michael Bailey, Thorsten Holz, Manolis Stamatogiannakis, and Sotiris Ioannidis, editors, *Research in Attacks, Intrusions, and Defenses*, pages 623–643, Cham, 2018. Springer. (Cited on page 3.)

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer, Heidelberg, August 1997. (Cited on page 3.)

[CAE] CAESAR: Competition for authenticated encryption: Security, applicability, and robustness. https://competitions.cr.yp.to/caesar.html. (Cited on page 19.)

[CER08] CERT Vulnerability Notes Database. Vulnerability note VU#925211: Debian and Ubuntu OpenSSL packages contain a predictable random number generator. https://www.kb.cert.org/vuls/id/925211, 2008. (Cited on page 3.)

[CM09] Jean-Sébastien Coron and Avradip Mandal. PSS is secure against random fault attacks. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 653–666. Springer, Heidelberg, December 2009. (Cited on page 5.)

[DEK+16] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Victor Lomné, and Florian Mendel. Statistical fault attacks on nonce-based authenticated encryption schemes. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 369–395. Springer, Heidelberg, December 2016. (Cited on page 5.)

[DGP07] Leo Dorrendorf, Zvi Gutterman, and Benny Pinkas. Cryptanalysis of the windows random number generator. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM CCS 2007: 14th Conference on Computer and Communications Security*, pages 476–485. ACM Press, October 2007. (Cited on page 3.)

[DMMP19] Christoph Dobraunig, Stefan Mangard, Florian Mendel, and Robert Primas. Fault attacks on nonce-based authenticated encryption: Application to keyak and ketje. In Carlos Cid and Michael J. Jacobson Jr:, editors, *SAC 2018: 25th Annual International Workshop on Selected Areas in Cryptography*, volume 11349 of *Lecture Notes in Computer Science*, pages 257–277. Springer, Heidelberg, August 2019. (Cited on page 5.)

[Dwo07] Morris Dworkin. Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC, November 2007. NIST Special Publication 800-38D. (Cited on page 19.)

[fai10] fail0verflow. Console hacking 2010: PS3 epic fail. In *27th Chaos Communication Congress*. Chaos Computer Club, 2010. (Cited on page 3.)

[FGL+12] Pierre-Alain Fouque, Nicolas Guillermin, Delphine Leresteux, Mehdi Tibouchi, and Jean-Christophe Zapalowicz. Attacking RSA-CRT signatures with faults on montgomery multiplication. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems – CHES 2012*, volume 7428 of *Lecture Notes in Computer Science*, pages 447–462. Springer, Heidelberg, September 2012. (Cited on page 5.)

[GL15] Shay Gueron and Yehuda Lindell. GCM-SIV: Full nonce misuse-resistant authenticated encryption at under one cycle per byte. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015: 22nd Conference on Computer and Communications Security*, pages 109–119. ACM Press, October 2015. (Cited on page 21.)

[GLM+04] Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 258–277. Springer, Heidelberg, February 2004. (Cited on pages 6, 16, and 23.)

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988. (Cited on page 7.)

[GPR06] Zvi Gutterman, Benny Pinkas, and Tzachy Reinman. Analysis of the linux random number generator. In *2006 IEEE Symposium on Security and Privacy*, pages 371–385. IEEE Computer Society Press, May 2006. (Cited on page 3.)

[GW96] Ian Goldberg and David Wagner. Randomness and the Netscape browser. *Dr. Dobb's Journal*, 1996. (Cited on page 3.)

[IPSW06] Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private circuits II: Keeping secrets in tamperable circuits. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 308–327. Springer, Heidelberg, May / June 2006. (Cited on page 5.)

[JLQ99] Marc Joye, Arjen K. Lenstra, and Jean-Jacques Quisquater. Chinese remaindering based cryptosystems in the presence of faults. *Journal of Cryptology*, 12(4):241–245, September 1999. (Cited on page 3.)

[Jou06] Antoine Joux. Authentication failures in nist version of gcm. `http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/Joux_comments.pdf`, 2006. (Cited on page 19.)

[KDK+14] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. In *Proceeding of the 41st Annual International Symposium on Computer Architecuture*, ISCA '14, pages 361–372, Piscataway, NJ, USA, 2014. IEEE Press. (Cited on pages 3 and 9.)

[Len96] Arjen K Lenstra. Memo on RSA signature generation in the presence of faults, 1996. (Cited on page 4.)

[MNPV99] David M'Raïhi, David Naccache, David Pointcheval, and Serge Vaudenay. Computational alternatives to random number generators. In Stafford E. Tavares and Henk Meijer, editors, *SAC 1998: 5th Annual International Workshop on Selected Areas in Cryptography*, volume

1556 of *Lecture Notes in Computer Science*, pages 72–80. Springer, Heidelberg, August 1999. (Cited on page 3.)

[MV04] David A. McGrew and John Viega. The security and performance of the Galois/counter mode (GCM) of operation. In Anne Canteaut and Kapalee Viswanathan, editors, *Progress in Cryptology - INDOCRYPT 2004: 5th International Conference in Cryptology in India*, volume 3348 of *Lecture Notes in Computer Science*, pages 343–355. Springer, Heidelberg, December 2004. (Cited on page 19.)

[MW78] T. C. May and M. H. Woods. A new physical mechanism for soft errors in dynamic memories. In *16th International Reliability Physics Symposium*, pages 33–40, April 1978. (Cited on page 3.)

[Nat13] National Institute of Standards and Technology. Digital Signature Standard (DSS) (FIPS PUB 186-4), July 2013. (Cited on pages 3 and 15.)

[NRS14] Chanathip Namprempre, Phillip Rogaway, and Thomas Shrimpton. Reconsidering generic composition. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 257–274. Springer, Heidelberg, May 2014. (Cited on pages 8 and 21.)

[Per16] Trevor Perrin. The XEdDSA and VXEdDSA signature schemes. https://signal.org/docs/specifications/xeddsa/, 2016. (Cited on pages 4, 16, and 22.)

[Por13] T. Pornin. Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA). RFC 6979 (Informational), August 2013. (Cited on pages 3 and 14.)

[PSS+18] D. Poddebniak, J. Somorovsky, S. Schinzel, M. Lochter, and P. Rösler. Attacking deterministic signature schemes using fault attacks. In *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018*, pages 338–352. IEEE, April 2018. (Cited on pages 3, 4, 5, 13, 14, 15, and 16.)

[RGB+16] Kaveh Razavi, Ben Gras, Erik Bosman, Bart Preneel, Cristiano Giuffrida, and Herbert Bos. Flip feng shui: Hammering a needle in the software stack. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016: 25th USENIX Security Symposium*, pages 1–18. USENIX Association, August 2016. (Cited on pages 3 and 9.)

[Rog02] Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, *ACM CCS 2002: 9th Conference on Computer and Communications Security*, pages 98–107. ACM Press, November 2002. (Cited on pages 5 and 8.)

[Rog04] Phillip Rogaway. Nonce-based symmetric encryption. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption – FSE 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 348–359. Springer, Heidelberg, February 2004. (Cited on page 5.)

[RP17] Y. Romailler and S. Pelissier. Practical fault attack against the ed25519 and eddsa signature schemes. In *2017 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 17–24, 2017. (Cited on pages 4, 5, 13, 14, 15, and 16.)

[RS06] Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 373–390. Springer, Heidelberg, May / June 2006. (Cited on pages 5, 8, 19, 21, and 23.)

[SB18] Niels Samwel and Lejla Batina. Practical fault injection on deterministic signatures: The case of EdDSA. In Antoine Joux, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *AFRICACRYPT 18: 10th International Conference on Cryptology in Africa*, volume 10831 of *Lecture Notes in Computer Science*, pages 306–321. Springer, Heidelberg, May 2018. (Cited on page 4.)

[SBB+18] Niels Samwel, Lejla Batina, Guido Bertoni, Joan Daemen, and Ruggero Susella. Breaking Ed25519 in WolfSSL. In Nigel P. Smart, editor, *Topics in Cryptology – CT-RSA 2018*, volume 10808 of *Lecture Notes in Computer Science*, pages 1–20. Springer, Heidelberg, April 2018. (Cited on pages 4, 5, 13, 14, 15, and 16.)

[Sch16] Benedikt Schmidt. [curves] EdDSA specification. https://moderncrypto.org/mail-archive/curves/2016/000768.html, 2016. (Cited on page 4.)

[Shr04] Tom Shrimpton. A characterization of authenticated-encryption as a form of chosen-ciphertext security. Cryptology ePrint Archive, Report 2004/272, 2004. http://eprint.iacr.org/2004/272. (Cited on page 8.)

[Sig] Signal: Technical documentation. https://whispersystems.org/docs/. (Cited on pages 4 and 16.)

[TT19] Akira Takahashi and Mehdi Tibouchi. Degenerate fault attacks on elliptic curve parameters in OpenSSL. In *2019 IEEE European Symposium on Security and Privacy, EuroS&P 2019*. IEEE, June 2019. (to appear). (Cited on page 6.)

[Vau03] Serge Vaudenay. The security of DSA and ECDSA. In Yvo Desmedt, editor, *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 309–323. Springer, Heidelberg, January 2003. (Cited on page 15.)

[YL06] T. Ylonen and C. Lonvick (Ed.). The Secure Shell (SSH) Authentication Protocol. RFC 4252 (Proposed Standard), January 2006. Updated by RFCs 8308, 8332. (Cited on page 3.)