Security of Hedged Fiat–Shamir Signatures under Fault Attacks*

Diego F. Aranha[1], Claudio Orlandi[2], Akira Takahashi[2], and Greg Zaverucha[3]

[1]Department of Engineering, DIGIT, Aarhus University
dfaranha@eng.au.dk
[2]Department of Computer Science, DIGIT, Aarhus University
orlandi@cs.au.dk, takahashi@cs.au.dk
[3]Microsoft Research
gregz@microsoft.com

February 24, 2020

Abstract

Deterministic generation of per-signature randomness has been a widely accepted solution to mitigate the catastrophic risk of randomness failure in Fiat–Shamir type signature schemes. However, recent studies have practically demonstrated that such de-randomized schemes, including EdDSA, are vulnerable to differential fault attacks, which enable adversaries to recover the entire secret signing key, by artificially provoking randomness reuse or corrupting computation in other ways. In order to balance concerns of both randomness failures and the threat of fault injection, some signature designs are advocating a "hedged" derivation of the per-signature randomness, by hashing the secret key, message, and a nonce. Despite the growing popularity of the hedged paradigm in practical signature schemes, to the best of our knowledge, there has been no attempt to formally analyze the fault resilience of hedged signatures.

We perform a formal security analysis of the fault resilience of signature schemes constructed via the Fiat–Shamir transform. We propose a model to characterize bit-tampering fault attacks, and investigate their impact across different steps of the signing operation. We prove that, for some types of faults, attacks are mitigated by the hedged paradigm, while attacks remain possible for others. As concrete case studies, we then apply our results to XEdDSA, a hedged version of EdDSA used in the Signal messaging protocol, and to Picnic2, a hedged Fiat–Shamir signature scheme in Round 2 of the NIST Post-Quantum standardization process.

*An extended abstract appears at EUROCRYPT 2020. This is the full paper.

Contents

1 Introduction

Deterministic Signatures and Fault Attacks Some signature schemes require a fresh, secret random value per-signature, sometimes called a nonce. Nonce misuse is a devastating security threat intrinsic to these schemes, since the signing key can be computed after as few as two different messages are signed using the same value. The vulnerability can result from either programming mistakes attempting to implement non-trivial cryptographic standards, or faulty pseudo-random number generators. After multiple real-world implementations were found to be surprisingly vulnerable to this attack [fai10, BR18] researchers and practitioners proposed deterministic signature schemes, such as EdDSA [BDL+12], as a countermeasure, in which per-signature randomness is derived from the message and secret key as a defense-in-depth mechanism. However, it has been shown that simple low-cost fault attacks during the computation of the derandomized signing operation can leak the secret key by artificially provoking nonce reuse or by corrupting computation in other ways [Bae14, Sch16, BP16, ABF+18]. Recent papers have experimentally demonstrated the feasibility of these attacks [RP17, PSS+18, SB18]. Moreover, [BP18] and [RJH+19] extended such fault attacks to exploit deterministic lattice-based signature schemes among round two candidates of the NIST Post-Quantum Cryptography Standardization Process [AASA+19]. Despite these attacks, deterministic signature generation is still likely a positive outcome in improving security, since fault attacks are harder to mount.

In general, fault analysis consists of an invasive side-channel attack where the adversary corrupts computation to obtain some advantage in breaking a cryptographic implementation, such as forcing it to produce faulty results which reveal internal state or bits of a secret key. Faults can be either ephemeral, as in the case of the forced nonce reuse attacks described above, or persistently damage cryptographic implementations.

While (differential) fault analysis was already proposed as an attack methodology against symmetric-key cryptography [BS97] and some signature/identification schemes [BDL97], it became clear only more recently that it also poses a severe threat to modern signature schemes. This is especially relevant in the context of post-quantum signature schemes subject to standardization, in which side-channel resistance is listed as one of the main security requirements by NIST. In particular, the published evaluation criteria indicate that candidates that admit side-channel protection at minimal cost have a clear advantage.

Fault Resilience of Hedged Signatures In order to balance concerns of both nonce reuse and the threat of fault injection, some signature designs are advocating deriving the per-signature randomness from the secret key sk, message m, and a nonce n. The intention is to re-introduce some randomness as a countermeasure to fault injection attacks, and gracefully handle the case of poor quality randomness, to achieve a middle-ground between fully-deterministic and fully-probabilistic schemes. We call constructions following this paradigm *hedged signatures*. Despite the growing popularity of the *hedged* paradigm in practical signature schemes (such as in XEdDSA, VXEdDSA [Per16], qTESLA [BAA+19], and Picnic2 [ZCD+19]), to the best of our knowledge, there has been no attempt to formally analyze the fault resilience of hedged signatures in the literature. While the hedged construction intuitively mitigates some fault attacks that exploit the deterministic signatures, it does add a step where faults can be injected, and it has not been shown if faults to the hedging operation allow further attacks, potentially negating the benefit. Therefore, we set out to study the following question within the provable security methodology:

To what extent are hedged signatures secure against fault attacks?

Concretely, we study fault attacks in the context of signature schemes constructed from identification schemes using the Fiat–Shamir transform [FS87]. We propose a formal model to capture the internal functioning of signature schemes constructed in the hedged paradigm, and characterize faults to investigate their impact across different steps of the signature computation.

We prove that for some types of faults, attacks are mitigated by the hedged paradigm, while for others, attacks remain possible. This provides important information when designing fault-tolerant implementations. We then apply our results to hedged EdDSA (called XEdDSA) and the Picnic2 post-quantum signature scheme [ZCD+19], both designed using the hedged construction. The XEdDSA scheme is used in the Signal protocol [CCD+17] which is in turn used by instant messaging services such as WhatsApp, Facebook Messenger and Skype.

Threat Model We consider a weaker variant of the standard adversary assumed in the fault analysis literature [JT12], who is typically capable of injecting a fault into an arbitrary number of values. Our adversary is capable of injecting a single-bit fault each time a signature is computed. We further restrict the faults to be injected at the interfaces between the typical *commit, challenge, and response* phases of Fiat–Shamir signatures, i.e., only those function inputs and outputs can be faulted. This models transient faults injected into registers or memory cells, but does not fully capture persisting faults that permanently modify values in key storage, voltage glitches to skip instructions or micro-architectural attacks to modify executed instructions (such as RowHammer and variants [KDK+14]).

We argue that, even if our model does not capture *all* possible fault attacks, it provides a meaningful abstraction of a large class of fault attacks, and thus our analysis provides an important first step towards understanding the security of hedged signatures in the presence of faults. This way, designers and implementers can focus on protecting the portions of the attack surface that are detected as *most relevant* in practice. We observe that the effects of fault attacks found in the literature targeting deterministic signatures can be essentially characterized as simple bit-tampering faults on function input/output, even though some of actual experiments cause faults during computation [BP18]. This is a result of the lower budget, timing constraints and precision required for the attack to be effective [KSV13, JT12]. Moreover, an abstract model is needed to prove general results, and the general functions common to all Fiat–Shamir signatures are a natural candidate for abstraction.

We consider two single-bit tampering functions to set or flip individual bits, respectively: $\texttt{flip\_bit}_i(x)$ to perform a logical negation of the i-th bit of x, and $\texttt{set\_bit}_{i,b}(x)$ to set the i-th bit of x to b. This captures both stuck-at and bit-flip fault injection attacks [KSV13], introduced as data flows through the implementation. Such attacks are practically targeted at various components of the device, e.g., memory cells, processor registers, or data buses.

1.1 Our Contributions

The main results of this paper are summarized as follows.

A new security model for analyzing fault attacks. We establish a formal security model tailored to Fiat–Shamir type signatures (hedged, deterministic or fully probabilistic). We survey the literature on fault attacks, showing that our model captures many practical attacks. As a first step, we abstract real-world hedged signature schemes, basing our formalization on Bellare and Tackmann's nonce-based signatures [BT16] and Bellare, Poettering and Stebila's de-randomized signatures [BPS16]. We call this security notion *unforgeability under chosen message and nonce attacks* UF-CMNA. In this security experiment, when submitting a message to the signing oracle, the adversary may also choose the random input to the *hedged extractor*, a function that derives the per-signature randomness from a nonce, the secret key, and the message.

Then we extend UF-CMNA to include resilience to fault attacks. In this security experiment the adversary plays a game similar to the UF-CMNA game, but the signing oracle also allows the attacker to specify a fault to be applied to a specific part of the signing algorithm. We identify eleven different fault types that the adversary can apply to the signing algorithm, and we denote them by by $f_0, \ldots, f_{10}$. For example, fault type f_1 applies `set_bit` or `flip_bit` to the secret key input to the hedged extractor. This notion is called *unforgeability under faults, chosen message and nonce attacks*, and is denoted F-UF-fCMNA where F is a set of fault types.

Fault resilience of hedged Fiat–Shamir signatures. We then prove that hedged Fiat–Shamir signature schemes are secure against attacks using certain fault types. Of the eleven fault types in our model, we found that the generic hedged Fiat–Shamir signature scheme is resilient to six of them (summarized in Fig. 1). As our model gives the attacker nearly full control of the RNG by default, our main results indicate that the hedged scheme can resist additional faults even in this (usually dire) scenario. The only constraint is that message-nonce pairs do not repeat as otherwise the scheme degenerates to a pure deterministic construction and attacks become trivial. When the underlying ID scheme has an additional property that we call *subset revealing*, the corresponding hedged signature scheme is secure against attacks that use eight of the eleven fault types. Informally, an ID scheme is subset revealing if the response to the challenge reveals a subset of

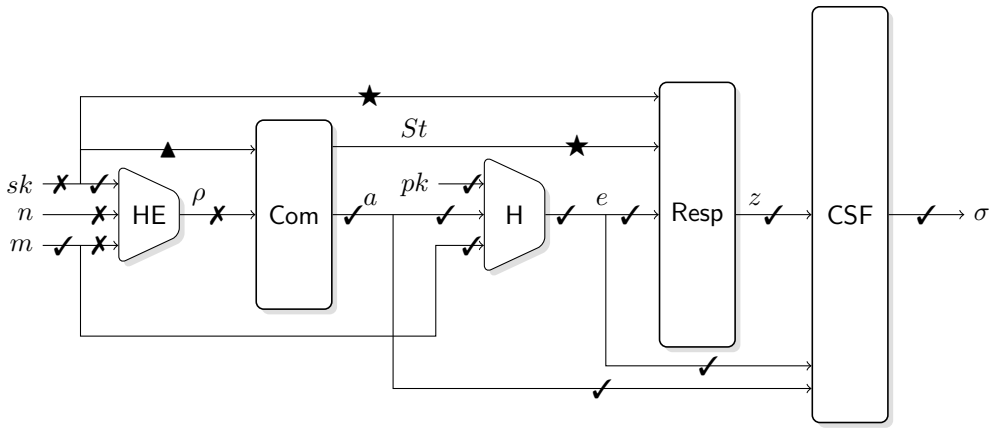

Figure 1: Overview of our results for hedged Fiat–Shamir type signature schemes. ✓ indicates security against 1-bit fault on the corresponding wire value, and ✗ indicates an attack or counterexample. A ★ (resp. ▲) indicates that security only holds for the schemes derived from subset-revealing ID (resp. input-delayed ID) protocols. The function components HE, Com, H, Resp, and CSF stand for hedged extractor, commitment, hash function, response, and canonical serialization function, respectively (see Sections 2 and 4 for the formal definitions).

the state computed during the preparation of the first message, and does not depend on the secret key (for instance, Schnorr is not subset-revealing while Picnic is).

Overall, our results give a full characterization of which fault attacks are mitigated as intended by the hedged construction, and which fault attacks remain. Our conclusion is that hedging is never worse than the deterministic construction with respect to faults, plus it has the additional benefit of hedging against poor randomness.

Fault resilience of XEdDSA and Picnic2. We use the Schnorr signature scheme throughout the paper as an example. As an application of our results, we show that hedged Schnorr resists attacks for six of the eleven fault types in our model. One implication is that the hedged scheme XEdDSA does provide better resistance to fault attacks than (deterministic) EdDSA. In particular, XEdDSA resists all fault injection attacks against EdDSA described in the literature that rely on nonce reuse without skipping nonce generation entirely [BP16, RP17, ABF+18, PSS+18, SB18]. We also show to what extent the Picnic2 signature scheme is secure against the fault attacks in our model. Because it is subset-revealing, resistance to eight of the eleven fault types is immediately established by our results for generic ID schemes. For the remaining three, we prove security for one (using specific details of Picnic2), and show attacks for the other two.

1.2 Related Work

To the best of our knowledge, ours is the first work considering fault attacks on hedged constructions. However, the modeling and construction of secure cryptographic schemes in the presence of faults or tampering attacks has received plenty of attention in recent years. We survey some of this work below. Related work on fault attacks to deterministic signature schemes is given in Section 3.

De-randomized and Hedged Constructions. Bellare and Tackmann [BT16] studied cryptography that is hedged against randomness failures. Signing is made deterministic, and takes a long-term secret called the seed, and a per-signature random value called a nonce. Security is assured if either (1) the seed is secret, or (2) the nonce is unpredictable (in addition to the signing key remaining secret, and the signature scheme must be UF-CMA secure to begin with). They also describe the "folklore construction", where instead of a separate secret seed, the signing key and message to be signed are used to derive the per-signature randomness, and additional randomness may or may not be included in the derivation. Some special cases have been

analyzed (for UF-CMA security): Schnorr signatures by M'Raihi et al. [MNPV99] and ECDSA by Koblitz and Menezes [KM15].

A generic version of the folklore derandomization construction was proven UF-CMA secure by Bellare, Pottering and Stebila [BPS16]. The proof is tight when the per-signature randomness is derived from a random oracle (independent of any other ROs used in the signature scheme). They also showed that the underlying randomized signature scheme need not even satisfy the traditional UF-CMA security, and it is only required to have a weaker property called "unique unforgeability" (abbreviated to UUF-CMA), meaning the scheme is unforgeable with the restriction that the adversary gets only one signature per message. This arises naturally when modeling derandomized signatures, where at most one signature is created per message.

Other works on hedged cryptography include [RY10] and [BBN+09, BH15, BPS17, HLC+18] when considering hedged public-key encryption in particular.

Fault Attacks and Tamper-Resilient Signatures. Tamper-resilient cryptography has received plenty of attention, both in the context of theoretical and practical cryptographic research, dating back at least to the early paper of Boneh, Demillo and Lipton [BDL97] considering fault attacks on RSA signatures (here it is noted that some attacks fail when a random padding is used, since it ensures that the same message is never signed twice). Later Coron and Mandal [CM09] proved that RSA-PSS is protected against random faults, and Barthe et al. [BDF+14] extends this to non-random faults as well. All of the above works contain examples of how randomization improves the security of signature schemes against fault attacks (in a provable way).

Other early work includes Gennaro et al. [GLM+04] that provides an early framework for proving tamper resilience, and Ishai et al. [IPSW06] which proposes generic transformation for tamper-resilient circuits. In a later work by Faust et al. [FPV11] a different and incomparable model was considered, which in particular guarantees security against tampering with *arbitrary* number of wires. We note that our model is similar to theirs since it also considers adversaries that are allowed to flip or reset each bit in the circuit. Similar ideas are also used in practice when considering fault resilient masking (e.g., [DAN+18]).

In our model the adversary is only allowed to tamper with part of the computation. Similar limitations have been considered before in the literature to circumvent impossibility results, in particular in the so called *split-state model* [DPW18]. Several constructions have been proposed in this model including: non-malleable codes (Dziembowski, Pietrzak and Wichs [DPW18]), signature schemes (Faonio et al. [FNSV18]), and more (Liu and Lysyanskaya [LL12]).

Other related work on tamper resilient signature schemes include the work of: Faonio and Venturi [FV16] (that presented a generic way to construct bounded tamper resilient signatures in the standard model), Fujisaki and Xagawa [FX16] (who among other things proved impossibility of non-persistent tamper-resilient signatures against arbitrary class of tampering functions, even in the presence of self-destruct and key-updating mechanisms), Austrin et al. [ACM+17] (who considered p-tampering attacks that tamper with each bit of the random tapes of cryptographic algorithms) and Damgård et al. [DFMV17] (who proved the bounded tamper resilience of Σ-protocols against the arbitrary class of tampering functions, considering the faults on the witness and public parameters).

Most of this previous work has focused on *constructing* novel tamper resilient signature schemes, or understanding the limits of tamper resilience, in theory. Instead, we focus on analyzing the tamper resilience of a popular transformation used in practice. Moreover, some of the previous work must restrict the tampering to a bounded amount, since they allow for arbitrary polynomial time tampering functions. In our case, we restrict the tampering capabilities to realistic functions, motivated by surveying the state of the art in fault attacks against real-world signatures, but instead we do not assume any upper-bound on the number of faulty signing queries.

Related key attacks (RKA) can be seen as a special case of tampering, and we thus mention some related work. Bellare and Kohno [BK03] initiated the formal study of related-key attacks and several primitives where later studied: Bellare, Cash and Miller [BCM11] showed that RKA-secure signatures can be constructed from RKA-secure PRG; Patterson, Schuldt and Sibborn [PSS14] considered related randomness attacks against public key encryption; Morita et al. [MSM+16] analyzed RKA security of Schnorr signatures.

Ineffective Fault Attacks (IFA) and Countermeasures. In this paper we consider not only `flip_bit` fault attacks, but also `set_bit` faults for the following reason. Clavier [Cla07] proposed *ineffective fault attacks (IFA)*, in which the adversary forces a certain intermediate bit value to be stuck at 0 or 1, and tries to recover the secret internal state by observing whether the correct output is obtained (i.e., the injected fault was ineffective). IFA is very powerful, and works even if the target algorithm contains typical countermeasures against fault attacks, such as a correctness check after redundant operations [BCN+06] and the infective countermeasure [YJ00]. IFA has been recently superseded by *statistical* ineffective fault attacks (SIFA) [DEK+18, DEG+18], that use statistical analysis to enable mounting IFA with low-precision bit-fixing, random or bit-flip faults. Daemen et al. [DDE+19] provided several practical countermeasures against SIFA, and their abstract adversarial model is close to ours in the sense that the adversaries are allowed to flip or set a single bit wire value in the circuit per query, though their security argument does not follow the provable security methodology.

Formal Methods. Some previous works that analyze security of cryptography against fault attacks use formal methods. The promise of formal methods is that they allow many classes of faults to be considered with less manual effort. They could also allow implementers to easily test the effectiveness of countermeasures, without repeating the analysis manually. There can be many possible (fault type, fault location) pairs to consider, each potentially requiring a separate proof. So far, approaches based on formal methods have arguably not delivered this type of strong result.

In one study, Rauzy and Guilley [RG14] analyzed the security of multiple implementations of RSA-CRT against data faults at any point in the computation. Their tool proved the security of one implementation, and found attacks on the others. The security guarantee is weaker than a reduction-based proof, since it only rules out a certain class of attacks. Some drawbacks of [RG14] is that their tool is custom, and not general in that it only works for RSA-like cryptosystems, that are deterministic.

A different use of formal methods is by Barthe et al. [BDF+14], who prove the security of RSA-PSS against non-random faults. They use the EasyCrypt computer-aided framework to formally verify their (hand-written) security proof (and also find a mistake in the proof of [CM09]). Notably, this required significant changes to EasyCrypt itself, outside the scope of [BDF+14].

Moro et al. [MHER14] create a build-time transformation that replaces a program's machine instructions with an equivalent sequence that executes correctly even with instruction skip fault attacks. This countermeasure is formally proven to to be effective using a model-checking tool, and relies on the assumption that performing two faults on instructions separated by a small number of clock cycles is infeasible.

Concurrent Work. An independent work by Fischlin and Günther [FG20] proposes a memory fault model for digital signatures and authenticated encryption. Their main result about a generic hedged signature scheme is two-fold: it is provably secure when the nonce is fully faulted, or when the message, nonce, and hedged extractor output are all differentially faulted in each signing query. The former essentially coincides with our Lemma 3, but with a different proof technique. For the latter, the outcome diverges because the adversarial power in our model is different in the following ways: 1. the adversary can locally inject a fault into sk as a hedged extractor input, 2. the adversary can inject a bit-fixing fault, not only a bit-flip (i.e., differential) fault, 3. the adversary has nearly full control over the nonce, instead of assuming nonces are randomly generated and subject to bit flips later on, but 4. the adversary cannot inject multi-bit faults into multiple variables in a query. We additionally consider fault attacks on other various intermediate values inside the signing operation. Our treatment is then more fine-grained and successfully captures typical existing attacks on deployed deterministic schemes (like attacks that fault the challenge hash), while [FG20] does not. The upside of the generic approach in [FG20] is that the result applies to more signature schemes.

1.3 Paper Organization

In Section 2 we provide basic definitions related to digital signature schemes derived from the Fiat–Shamir transform. Section 3 surveys recent fault attacks against deterministic Fiat–Shamir type signature schemes, to classify and characterize types of fault adversaries found in the literature. We then formally define hedged signature schemes in Section 4 together with their security notion against fault adversaries. In Section 5, we present our main result with formal security proofs, and clarify against which kind of fault attacks hedged

Fiat–Shamir signature schemes remain secure. Then Sections 6 and 7 apply our results to concrete instances of hedged signature schemes, the XEdDSA scheme (§ 6) and Picnic2 (§ 7). Finally, Section 8 discusses some fault attacks that are not covered by our model, to illustrate the limitations of our analysis, and indicate an interesting direction for future work.

2 Preliminaries

Notation The notation $|\cdot|$ denotes two quantities depending on the context: $|S|$ denotes the cardinality of a set S, and $|s|$ denotes the length of a bit string s. The notation $x \leftarrow\!\!\$\ X$ means that an element x is sampled from the set X uniformly at random. We often use the notation $[n]$ as a short hand for a set $\{1, \ldots, n\}$ where $n \in \mathbb{N}$. When we explicitly mention that an algorithm A is randomized, we use the notation $\mathsf{A}(x; \rho)$ meaning that it is executed on input x with random tape ρ.

2.1 Fiat–Shamir type Signature Schemes

This paper studies the robustness of Fiat–Shamir type signature schemes against fault attacks. Some concrete examples of schemes in this category that we consider are given here. A relevant signature algorithm that is not constructed with the Fiat–Shamir transform is ECDSA, but RFC 6979 [Por13, Section 3.6] describes a derandomized implementation and includes a variant that matches the hedged construction we study in this paper.

Schnorr and EdDSA Signatures The Schnorr signature scheme [Sch91] is one of the most well-known signature schemes using the Fiat–Shamir transform. The original paper instantiates the scheme in a finite field, but easily generalizes to other groups where the discrete logarithm problem is hard. Later work used the group of points on an elliptic curve, and a recently standardized [JL17] variant gaining popularity is EdDSA [BDL+12]. Appendix B gives the details of these algorithms. One notable feature of EdDSA is that it is deterministic, signatures are derandomized by deriving the per-signature random value from a secret key and the message to be signed. Applying the hedging strategy from this paper to EdDSA is simple, and it has been already employed in the XEdDSA signing algorithm used in the Signal protocol [Per16]. The only drawback (that we are aware of) is that the test vectors of [JL17] are no longer applicable.

Picnic Signatures Picnic [CDG+17] is a signature scheme that is designed to provide security against attacks by quantum computers (in addition to attacks by classical computers). The scheme is a second round candidate for the NIST Post-Quantum Cryptography project [AASA+19]. Here we provide a brief description based on the design document [Pic19b], and in Appendix C we give more details. The scheme uses a zero-knowledge proof system and is based on symmetric-key primitives like hash functions and block ciphers with conjectured post-quantum security. In particular, Picnic does not rely on number-theoretic or algebraic hardness assumptions.

The proof system is one of ZKBoo [GMO16], ZKB++ [CDG+17] or KKW [KKW18]; all designed for zero-knowledge proofs on arbitrary circuits. These proof systems build on the MPC-in-the-head paradigm of Ishai et al. [IKOS07], that we describe only informally here. The multiparty computation protocol (MPC) will implement the relation, and the input is the witness. For example, the MPC could compute $y = \text{SHA-256}(x)$ where players each have a share of x and y is public. The idea is to have the prover simulate a multiparty computation protocol "in their head", commit to the state and transcripts of all parties, then have the verifier "corrupt" a random subset of the simulated parties by seeing their complete state. The verifier then checks that the opened parties performed the correct computation, and if so, he has some assurance that the output is correct. Iterating this for many rounds then gives the verifier high assurance that the prover knows the witness. To make this non-interactive, the rounds are done in parallel, and the challenge is computed with the Fiat–Shamir transform. In the Picnic signature scheme, the secret key sk (the witness) is the key to a block cipher, and the public key (the relation) is a plaintext-ciphertext pair created by sk.

Clearly, while Picnic's design is based on Fiat–Shamir, it is much different from Schnorr signatures. We chose to analyze it in part to validate the generality of our results. It is also a relatively new scheme that has had little analysis with respect to fault attacks and we hope that our results will guide future analysis, by ruling out some classes of attacks.

$\mathsf{Exp}^{\mathsf{UF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{A})$	$\mathsf{OSign}(m)$	$\mathsf{H}(x)$
$M \leftarrow \emptyset; \mathrm{HT} \leftarrow \emptyset$	$\rho \leftarrow\!\!\$\ D_\rho$	If $\mathrm{HT}[x] = \perp$:
$(sk, pk) \leftarrow \mathsf{Gen}(1^\lambda)$	$\sigma \leftarrow \mathsf{Sign}(sk, m; \rho)$	$\mathrm{HT}[x] \leftarrow\!\!\$\ D_H$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{OSign},\mathsf{H}}(pk)$	$M \leftarrow M \cup \{m\}$	**return** $\mathrm{HT}[x]$
$v \leftarrow \mathsf{Verify}(pk, m^*, \sigma^*)$	**return** σ	
return $(v = 1) \wedge m^* \notin M$		

Figure 2: Standard UF-CMA experiment in the random oracle model

$\mathsf{Prover}(sk; \rho)$		$\mathsf{Verifier}(pk)$
$(a, St) \leftarrow \mathsf{Com}(sk; \rho)$	$\xrightarrow{a}$	
	$\xleftarrow{e}$	$e \leftarrow\!\!\$\ D_H$
$z \leftarrow \mathsf{Resp}(sk, e, St)$	$\xrightarrow{z}$	$0/1 \leftarrow \mathsf{V}(a, e, z, pk)$

Figure 3: Canonical identification protocol ID, executed with random tape $\rho \in D_\rho$. The set D_H may be bitstrings, or have more structure (e.g., a list of integers).

2.2 Definitions

In this subsection we recall several basic definitions related to digital signatures constructed from the identification protocols. Since this paper deals with Fiat–Shamir signatures, we always assume that the signing algorithm of digital signature schemes takes some randomness as input.

Definition 1 (Digital Signature Scheme)**.** A *digital signature scheme*, denoted by a tuple of algorithms $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$, is defined as follows.

- $\mathsf{Gen}(1^\lambda)$, where λ is a security parameter, outputs a key pair (sk, pk). We assume that a key pair defines the set $D_\rho \subseteq \{0,1\}^{\ell_\rho}$ from which a randomness ρ of length ℓ_ρ-bit is sampled.
- $\mathsf{Sign}(sk, m; \rho)$, where sk is a signing key, m is a message, and $\rho \in D_\rho$ is a random value, outputs a signature σ.
- $\mathsf{Verify}(pk, m, \sigma)$, where pk is a public key, outputs 1 (i.e., accept if the signature is valid) or 0 (i.e., reject if the signature is invalid).

We say that SIG is *correct* if for every key pair (pk, sk) output by Gen, and for every $m \in \{0,1\}^*$ and $\rho \in D_\rho$, the corresponding signature $\sigma = \mathsf{Sign}(sk, m; \rho)$ satisfies

$$\Pr[\mathsf{Verify}(pk, m, \sigma) = 1] = 1.$$

We tailor our definition of UF-CMA security to the random oracle model since our analysis will always model the hash function used in Sign as a random oracle, and all Fiat–Shamir-type signature schemes in this paper use a hash function.

Definition 2 (UF-CMA Security in the Random Oracle Model)**.** A signature scheme $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ is said to be UF-CMA (unforgeability against chosen message attacks) secure in the random oracle model, if for any probabilistic polynomial time adversary $\mathcal{A}$, its advantage

$$\mathbf{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{A}) \coloneqq \Pr\left[\mathsf{Exp}^{\mathsf{UF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{A}) = 1\right]$$

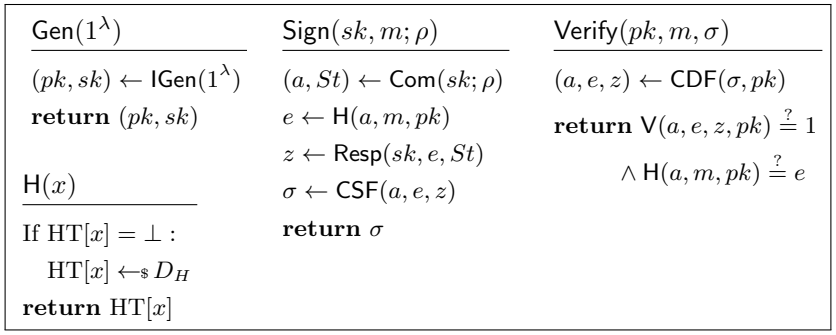

Figure 4: The Fiat–Shamir transform applied to canonical ID with serialization CSF, to construct the signature scheme $\mathbf{FS}[\mathsf{ID}, \mathsf{CSF}] = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$. The function $\mathsf{H} : \{0,1\}^* \to D_H$ is constructed with a cryptographic hash function which we model as a random oracle.

is negligible in λ, where $\mathsf{Exp}^{\mathsf{UF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{A})$ is described in Fig. 2. Moreover, SIG is said to be UF-KOA (unforgeability against key-only attack) secure if $\mathcal{A}$ makes no signing queries to OSign in $\mathsf{Exp}^{\mathsf{UF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{A})$.

We now define a three-round public-coin identification protocol, the basis of Fiat–Shamir-type signatures. The definition below essentially follows the formalization of [KLS18] unless explicitly stated.

Definition 3 (Canonical Identification Protocol)**.** A *canonical identification protocol*, denoted by a tuple of algorithms $\mathsf{ID} = (\mathsf{IGen}, \mathsf{Com}, \mathsf{Resp}, \mathsf{V})$and depicted in Fig. 3, is a three-round protocol defined as follows:

- IGen(1^λ), where λ is a security parameter, outputs a key pair (sk, pk). In the context of identification protocols, pk and sk are sometimes called *statement* and *witness*. We assume that IGen defines a *hard-relation*, and that pk defines the parameters of the scheme including: *randomness space* D_ρ, *commitment space* A, *challenge space* D_H and *response space* Z.
- Prover invokes a committing algorithm Com on a secret key sk and randomness $\rho \in D_\rho$ as input, and outputs a *commitment* $a \in A$ and *state* St.
- Verifier samples a *challenge* e from the challenge space $D_H \subseteq \{0,1\}^*$.
- Prover executes a response algorithm Resp on (sk, e, St) to compute a *response* $z \in Z \cup \{\perp\}$, where $\perp \notin Z$ is a special symbol indicating failure. On top of this standard formalization, we further require that Resp returns $\perp$ whenever it receives a malformed challenge $\tilde{e} \notin D_H$, as such a simple sanity check is performed in most practical implementations.
- Verifier executes a verification algorithm V on (a, e, z, pk) as input, to output 1 (i.e., accept) or 0 (i.e., reject).

We call a triple $(a, e, z) \in A \times D_H \times Z \cup \{\perp, \perp, \perp\}$ a *transcript*, and it is said to be *valid* with respect to pk if $\mathsf{V}(a, e, z, pk) = 1$. We say that ID is *correct* if for every pair (pk, sk) output by IGen, for every $\rho \in D_\rho$, and for every transcript (a, e, z) from an honest execution of the protocol between Prover$(sk; \rho)$ and Verifier(pk),

$$\Pr[\mathsf{V}(a, e, z, pk) = 1] = 1.$$

Remark The response algorithm in the above definition does not explicitly take a commitment a as input. We decided to do so since a is generally not required to compute z, such as in the Schnorr identification scheme and, if needed, we assume that St contains a copy of a.

The following definition is adapted from [HL10, Chapter 6]. We explicitly differentiate three flavors of the special HVZK property depending on a level of indistinguishability, following the approach found in [Gol07, Chapter 4]. Note that ϵ_{HVZK} below is equal to 0 for special *perfect* HVZK.

Definition 4 (Special c/s/p-HVZK)**.** Let $\mathsf{ID} = (\mathsf{IGen}, \mathsf{Com}, \mathsf{Resp}, \mathsf{V})$ be a canonical identification protocol. ID is said to be *special computational/statistical/perfect honest-verifier zero knowledge (special c/s/p-HVZK)* if there exists a probabilistic polynomial-time simulator $\mathcal{M}$, which on input pk and e outputs a transcript of the form (a, e, z) that is computationally/statistically/perfectly indistinguishable from a real transcript between an honest prover and verifier on common input pk. Formally, for every pair (pk, sk) output by IGen, and every $e \in D_H$ it holds that

$$\{\mathcal{M}(pk, e)\}_{sk,pk,e} \overset{c/s/p}{\equiv} \{\langle \mathsf{Prover}(sk; \rho), \mathsf{Verifier}(pk, e)\rangle\}_{sk,pk,e}$$

where $\overset{c/s/p}{\equiv}$ means that two families of random variables are computationally/statistically/perfectly indistinguishable, $\{\mathcal{M}(pk, e)\}_{sk,pk,e}$ denotes the output distribution of simulator $\mathcal{M}$ upon input pk and e, and $\{\langle \mathsf{Prover}(sk; \rho), \mathsf{Verifier}(pk, e)\}_{sk,pk,e}$ denotes the transcript distribution of an honest execution between Prover and $\mathsf{Verifier}$, in which we assume that Prover receives uniformly sampled randomness $\rho \in D_\rho$. We also denote by ϵ_{HVZK} the upper bound on the advantage of all probabilistic polynomial-time distinguishing algorithms.

In our security analysis of specific hedged-signature schemes in the presence of faults we will provide a concrete bound on the min-entropy of the associated ID scheme. But here we present a useful lemma stating that the commitment message a of any secure identification scheme must have high min-entropy. The lemma might be folklore but we were unable to find a reference to it, so we include it for completeness.

Lemma 1. *Let* ID *be a canonical identification protocol as in Definition 3, satisfying special-soundness and HVZK (as in Definition 4). Then, the min-entropy* α *of the commitment message* a *(given the public key) is at least* $\alpha = \omega(\log(\lambda))$

Proof. We construct an adversary that, given the public key pk of the ID scheme, recovers the secret key sk if $\alpha < \omega(\log(\lambda))$, thus contradicting the property that IGen generates a hard relation with security parameter λ. The adversary simply runs the HVZK simulator $\mathcal{M}$ on input $\mathcal{M}(pk, e_i)$ for different challenge values $e_1, \ldots, e_t \in D_H$, for a t polynomial in λ. Then, the adversary checks if there are any two equal commitment messages in the generated transcripts $a_i = a_j$ and, if so, outputs the secret key sk using the extractor guaranteed from the special soundness property. We analyze the success probability of this adversary. First of all, let ϵ (negligible in λ) be the HVZK parameter of the system. Then, using hybrid arguments, we can argue that the probability of the attack succeeding when using simulated transcripts is at least $(1 - t\epsilon)$ times the probability that the attack succeeds when using real transcripts (for which the special-soundness property is defined). Then, we note that the attack succeeds (if it was run with real transcripts) only if there is at least a collision between the observed a's, and this happens with probability $O(t^2 2^{-\alpha})$. So, all in all, our adversary can recover the secret key running in time t with probability $O((1 - t\epsilon)t^2 2^{-\alpha})$ which is non-negligible in λ if $\alpha < \omega(\log(\lambda))$. □

Definition 5 (Subset Revealing Identification Protocol)**.** Let $\mathsf{ID} = (\mathsf{IGen}, \mathsf{Com}, \mathsf{Resp}, \mathsf{V})$ be a canonical identification protocol. We say that ID is *subset revealing* if ID satisfies the following.

- St is a set of c states $\{St_1, \ldots, St_c\}$.
- Resp first derives an index set $I \subset [c]$ using only e as input, and outputs St_i for $i \in I$ as z.
- The number of states c and size of the challenge space $|D_H|$ are both polynomial in λ.

Remark. Similar definitions were previously given by Kilian et al. [KMO90] and Chailloux [Cha19], where they make zero-knowledge or identification protocols simply reveal a subset of committed strings. Our definition generalizes their notion so that it can cover some protocols that reveal arbitrary values other than committed strings. Also notice that the Resp function of subset revealing ID schemes does not use sk at all. The above definition includes the $\mathsf{Picnic2}$ identification protocol (discussed in more detail in Section 7), and many classic three-round public-coin zero-knowledge proof protocols, such as the ones for graph isomorphism, Hamilton graphs, and 3-colorable graphs [GMW86]. We also emphasize that $|St|$ and $|D_H|$ need to be restricted for efficiency reasons. For instance, the Schnorr ID scheme and its variants usually have an exponential size challenge space $|D_H| = 2^{2\lambda}$. Without the above restriction, they could also be described as

subset revealing identification protocols by precomputing all possible responses $\rho+sk, \rho+2\cdot sk, \ldots, \rho+2^{\lambda}\cdot sk$ in the Com step, storing them in St, to output one in Resp. However, this is clearly not efficiently computable and we therefore rule out such inefficient constructions by bounding the size of St.

Serialization of Transcripts. For efficiency purposes, most Fiat-Shamir based signature schemes do not include the entire transcript of the identification protocol as part of the signature. Instead, redundant parts are omitted and recomputed during the verification phase. Different signature schemes omit different parts of the transcript: in some cases a is omitted and in others e is omitted. To capture this in our framework without loss of generality we introduce a *serialization* function that turns the transcript of an identification protocol into a signature.

Definition 6 (Canonical Serialization Function)**.** Let $\mathsf{ID} = (\mathsf{IGen}, \mathsf{Com}, \mathsf{Resp}, \mathsf{V})$ be a canonical identification protocol, and let pk be a public key output by IGen. We call a function $\mathsf{CSF} : \{0,1\}^* \to \{0,1\}^*$ a *canonical serialization function* if CSF is efficiently computable and deterministic, and satisfies the following basic properties: 1) it is *valid*, meaning that there exists a corresponding de-serialization function CDF which satisfies the following: for any transcript $(a, e, z) \in A \times D_H \times Z \cup \{\bot, \bot, \bot\}$ such that $\mathsf{V}(a, e, z, pk) = 1$, it holds that $\mathsf{CDF}(\mathsf{CSF}(a, e, z), pk) = (a, e, z)$, and 2) it is *sound with respect to invalid responses*, meaning that it returns $\bot$ upon receiving $z = \bot$ as input.

Definition 7 (Commitment-recoverable Identification [KLS18])**.** Let $\mathsf{ID} = (\mathsf{IGen}, \mathsf{Com}, \mathsf{Resp}, \mathsf{V})$ be a canonical identification protocol, let pk be a public key output by IGen, and let (a, e, z) be a transcript such that $\mathsf{V}(a, e, z, pk) = 1$. We call ID *commitment-recoverable*, if there exists a public function $\mathsf{Recover}(pk, e, z)$ that outputs a.

Definition 8 (Fiat–Shamir Transform)**.** *The Fiat–Shamir transform*, denoted by **FS**, takes a canonical identification protocol ID and canonical serialization function CSF as input, and outputs a signature scheme $\mathbf{FS}[\mathsf{ID}, \mathsf{CSF}] = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ defined in Fig. 4. For convenience, this paper refers to such schemes as *Fiat–Shamir type signature schemes.*

Remarks By construction, it holds that if ID is correct, then $\mathbf{FS}[\mathsf{ID}, \mathsf{CSF}]$ is a correct signature scheme. We assume ID is correct throughout the paper. In Fig. 4, the verification condition may appear redundant. However, the above definition allows us to capture several variations of the Fiat–Shamir transform. For instance, a type of Fiat–Shamir transform found in some papers e.g., Ohta–Okamoto [OO98] and Abdalla et al.[AABN02] can be obtained by letting $\mathsf{CSF}(a, e, z)$ output $\sigma := (a, z)$ and letting $\mathsf{CDF}(\sigma, pk)$ call $e \leftarrow \mathsf{H}(a, m, pk)$ inside to reconstruct the whole transcript. In contrast, if ID is commitment-recoverable [KLS18], one can instantiate its serialization as follows: $\mathsf{CSF}(a, e, z)$ outputs $\sigma := (e, z)$ and $\mathsf{CDF}(\sigma, pk)$ calls $a \leftarrow \mathsf{Recover}(pk, e, z)$ inside to reconstruct the transcript. Finally, we could also consider a trivial (de)serialization where CSF is an identity function and CDF returns the input except pk.

Strictly speaking, these variants of Fiat–Shamir signatures are not equivalently secure without assuming some additional properties of commitment recovering function [BBSS18], but most of the analyses in this paper (especially in Section 5) will only focus on the simulation of signing oracles, where the difference between instances of serialization does not affect the results.

2.3 Relation between UF-KOA Security and UF-CMA Security

The security notion *unforgeability against key-only attacks* (UF-KOA), is the same as UF-CMA, but with the restriction that the adversary is only given the public key, and no Sign oracle. The following result is a mild generalization of [KMP16, Lemma 3.8]: the original lemma only covers perfect HVZK and does not include the serialization function which we use in this work. The proof is very similar to the original one and is provided in Appendix A.1 for completeness. In Section 5, we extend this result, showing that for some signature schemes security against key-only attacks implies security against certain fault attacks.

Lemma 2 (UF-KOA $\to$ UF-CMA)**.** *Let* ID *be a correct canonical identification protocol and* CSF *be a canonical serialization function for* ID*. Suppose* ID *satisfies the following:*

- ID *is special c/s/p-HVZK with efficient distinguishers' advantage being at most* ϵ_{HVZK}*.*

- ID *has* α*-bit min-entropy.*

If $\mathsf{FS} := \mathbf{FS}[\mathsf{ID}, \mathsf{CSF}]$ *is* UF-KOA *secure, then* FS *is* UF-CMA *secure in the random oracle model. Concretely, given a* UF-CMA *adversary* $\mathcal{A}$ *against* FS *running in time* t*, and making at most* Q_s *queries to* OSign*,* Q_h *queries to* H*, one can construct another adversary* $\mathcal{B}$ *against* FS *such that*

$$\mathbf{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathsf{FS}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{UF\text{-}KOA}}_{\mathsf{FS}}(\mathcal{B}) + \frac{(Q_s + Q_h)Q_s}{2^{\alpha}} + Q_s \cdot \epsilon_{HVZK},$$

where $\mathcal{B}$ *makes at most* Q_h *queries to its random oracle, and runs in time* t*.*

3 Fault Attacks on Deterministic Fiat–Shamir Signatures

In this section we survey some recent attacks on deterministic signature schemes like EdDSA [BDL+12], Dilithium [LDK+19] and qTESLA [BAA+19]. For the lattice based signatures (Dilithium and qTESLA) our results later in the paper do not directly apply since they are not following the standard Fiat–Shamir paradigm defined in Section 2, rather they follow the Fiat–Shamir *with aborts* due to Lyubashevsky [Lyu09, Lyu12]. Still, the fundamental attack strategies against them are quite similar to ones against Schnorr-like schemes, and the types of fault attacks surveyed here are captured by our model. We also note that qTESLA now offers randomized variants [qTE19], using the hedged construction studied in this paper.

3.1 Conceptual Overview of the Attacks

In recent years, several papers presented differential fault attacks against deterministic Fiat–Shamir-type schemes. This section briefly reviews the ideas of those previous attacks. Their attacks are summarized in Table 1, together with the specific fault injection technique employed in the attack.

The literature presents many examples of fault injection using different choices of technology and attack vectors [KSV13], ranging from interfering with the program counter through clock glitching, to more expensive and sophisticated optical fault injection targeting individual bits. The intended effects include preventing computations critical for security to be performed, changing or fixing individual bits. In several attacks employing bit-flips, all that is necessary is a single bit-flip introduced in a specific wire value at any position. These *uncontrolled* attacks are easier to mount, as they do not require much precision from the adversary, while *controlled* attacks aim at a specific interval of bits within an intermediate value and typically need post-processing across candidates through exhaustive search to discover what position was changed [ABF+18]. The same classification applies to instruction skipping attacks, which can be coarse-grained and affect any instructions inside a larger computation (e.g. an entire hash function), or involve more synchronization to carefully skip a single instruction (e.g. a function call).

Other than the locations to which a fault is injected, we observe that each attack falls into one of the following categories of attacks, which we describe below. The first two attacks are especially frequent and benefit most from the deterministic nature.

Special Soundness Attack (`SSND`) This type of attack exploits the k*-special soundness* property of the underlying canonical identification protocol. That is, there exists an efficient algorithm that extracts the witness sk corresponding to the statement pk, given k accepting transcripts $(a, e_1, z_1), \ldots, (a, e_k, z_k)$, where $e_i \neq e_j$ if $i \neq j$ [GMO16]. Note in fact that it is easier to extract the secret key for an attacker than for a knowledge extractor in a proof of security, since the attacker can assume that the prover honestly follows the protocol while the special soundness property considers possibly cheating provers. For example, when attacking deterministic Schnorr schemes like EdDSA, which is 2-special sound, the adversary can assume that the response z is of the form $\rho + e \cdot sk$. Then a fault attack can recover the secret key in a deterministic scheme by making two queries with the same message. First, the adversary asks for a non-faulty signature on some message m, to obtain a legitimate signature (e_1, z_1). Second, recall that deterministic signatures use the same ρ when signing the same message m. In that case, if the attacker modifies the hash input or output values with a fault injection technique, the attacker can obtain another signature using a different challenge

Table 1: Overview of the recent fault attacks on deterministic Fiat–Shamir-type signatures. The column Fault Type indicates which fault type from our model (defined in §4) was used in the attack. The column Attack Vector indicates the specific fault injection technique in use. `CBF` refers to Controlled Bit-Flip, a controlled fault introduced in a pre-determined range within a value. `UBF` refers to Uncontrolled Bit-Flip, when a value needs to be faulted in any position. `CIS` and `UIS` refer to Controlled and Uncontrolled Instruction Skipping, respectively, the corresponding analogues for corrupting computation in a more or less controlled manner within the signature computation.

Paper	Fault Type	Attack Type	Attack Vector	Target	Description
[BP16]-1	f_4	`SSND`	`UBF`	EdDSA	Faulty $\tilde{a}$ as hash input.
[BP16]-2	f_7	`LRB`	`CBF`	EdDSA	Faulty $\tilde{\rho} = \rho + \Delta$ with small Δ.
[RP17]	f_6	`SSND`	`UBF`	EdDSA	Faulty hash output $\tilde{e}$.
[ABF+18]-1	–	`SSND`	`UBF`	EdDSA	Faulty EC base point to get a faulty $\tilde{a}$ as hash input.
[ABF+18]-2	f_5	`SSND`	`CBF`	EdDSA	Faulty $\tilde{pk}$ as hash input.
[ABF+18]-3	f_2	`LRB`	`CBF`	EdDSA	Faulty $\tilde{\rho} = \rho + \Delta$ with small Δ.
[ABF+18]-4	f_2	`RR`	`UIS`	EdDSA	Fixed ρ.
[ABF+18]-5	f_4	`SSND`	`UIS`	EdDSA	Faulty commitment computation to get a faulty $\tilde{a}$ as hash input.
[ABF+18]-6	f_6	`SSND`	`CBF`	EdDSA	Faulty hash output $\tilde{e}$.
[ABF+18]-7	f_6	`SSND`	`CIS`	EdDSA	Fixed hash output.
[ABF+18]-8	–	`IR`	`CIS`	EdDSA	Instruction skipping during the response computation.
[PSS+18]	f_5	`SSND`	`UBF`	EdDSA	Faulty hash input $\tilde{a}, \tilde{m}$ or $\tilde{pk}$.
[SB18]	f_4	`SSND`	`UBF/UIS`	EdDSA	Faulty commitment computation to get a faulty $\tilde{a}$ as hash input.
[BP18]-1	f_5	`SSND`	`UBF/UIS`	Dilithium/qTESLA	Faulty hash input or output.
[BP18]-2	f_4	`SSND`	`UBF`	Dilithium/qTESLA	Faulty commitment computation to get a faulty $\tilde{a}$ as hash input.
[BP18]-3	f_4	`SSND`	`UBF`	Dilithium/qTESLA	Faulty sk as commitment input.
[BP18]-4	f_2	`LRB`	`CBF`	Dilithium/qTESLA	Faulty $\tilde{\rho} = \rho + \Delta$ with small Δ.
[RJH+19]	–	`IR`	`CIS`	Dilithium/qTESLA	Instruction skipping during the response computation.

e_2, but with the same per-signature randomness ρ. As a result, the attacker gets two signatures satisfying

$$z_1 = \rho + e_1 \cdot sk \mod q$$
$$z_2 = \rho + e_2 \cdot sk \mod q$$

where the the former was computed correctly and the latter is faulty. Then one can easily extract sk by computing $(z_1 - z_2)/(e_1 - e_2) \mod q$, which is essentially what the knowledge extractor for the underlying identification protocol does. For Schnorr variants that output (a, z) as a signature, including EdDSA (see Appendix B), the attack is slightly more involved since the adversary cannot directly obtain e_1 and e_2, and thus would have to somehow guess these values by simulating the faulty hash computation. However, the essential idea of the attack is same as above. For practical details, see e.g., [RP17].

We also remark that the fault attacker doesn't necessarily require k faulty signatures to break k-special sound signatures. For instance, ZKB++ [CDG+17] is a canonical ID protocol with HVZK and 3-special soundness, and it can be turned into a secure signature scheme; if its randomness generation is deterministic, collecting 2 faulty signatures on the same message would be sufficient to extract the witness, as the fault attacker can assume that the signer is honestly simulating the MPC protocol.

Large Randomness Bias Attack (LRB) This attack slightly modifies the randomness ρ to $\rho' = \rho + \Delta$ using, e.g., `flip_bit` fault. The attack highly relies on the deterministic property because the adversary knows that all signatures on the same message m use the same ρ, and if ρ is slightly perturbed by some sufficiently small Δ, he can find Δ with an exhaustive search. Then the adversary can recover the secret key by querying two deterministic signatures on the same message, which were computed using correlated randomness ρ and $\rho + \Delta$. For example, when attacking the deterministic Schnorr scheme and its variants such as EdDSA, the adversary obtains the following:

$$z_1 = \rho + e_1 \cdot sk \mod q$$
$$z_2 = (\rho + \Delta) + e_2 \cdot sk \mod q$$

where $e_1 \neq e_2$ with overwhelming probability because the second challenge was obtained by hashing the result of faulty commitment $\mathsf{Com}(sk; \rho+\Delta)$. Thus the adversary can extract sk by computing $(z_2 - z_1 - \Delta)/(e_2 - e_1) \mod q$ for all possible Δ and checking whether each secret key candidate corresponds to pk or not.

Randomness Reuse Attack (RR) This attack forces the randomness ρ to be a certain constant by, e.g., skipping the execution of PRF completely. It does not exploit the deterministic nature of the scheme and works for randomized signatures as well.

Invalid Response Attack (IR) This attack causes a fault during the computation of the response z. For instance, Schnorr signatures have $z = e \cdot sk + \rho$, and if the addition of ρ is skipped, the faulty response z' ends up being $e \cdot sk$, from which the recovery of the secret sk is trivial. It does not exploit the deterministic nature of the scheme either and works for randomized signatures as well.

3.2 Applicability to Deterministic Picnic2

The attacks surveyed above are general enough to apply to Fiat–Shamir signatures other than EdDSA, qTESLA and Dilithium. To demonstrate this, we apply them to a deterministic version of Picnic2, a Fiat–Shamir signature scheme with a significantly different design. The Picnic2 specification [Pic19a] recommends implementations be probabilistic, but allows implementations to omit the random value when deriving the per-signature random value from the secret key and message. In this section we use the notation defined in the description of Picnic2, in Appendix C.

`SSND` Unlike its predecessor ZKB++ proof system, the Picnic2 ID scheme is not special-sound, since from any pair of accepting transcripts it is not always possible to extract sk. However, in the context of `SSND` fault attacks, the challenge in the second transcript is always a random value (not a value chosen adversarially as in the definition of special soundness). Therefore, with probability at least τ/M the pair of challenges has an MPC instance j in common, and the corresponding unopened party p_j differs in both challenges. Therefore, in instance j, all parties are opened, and we can recover sk. For the L1 parameter set, with two transcripts we can extract a witness with probability greater than 0.078, and this probably increases quickly with more transcripts.

`LRB` This attack does not directly apply, since ρ is first input to a PRG to derive seeds, then the seeds are used to derive the random tapes for each for each party. If instead the already expanded random tapes are faulted (more analogous to EdDSA, which uses ρ directly), then it is possible to recover a single secret key bit by zeroing one of the random bits at the beginning of the tape. Since the random tape bits are used to share the bits of the secret key as input to the MPC protocol, if we know the unopened party's share is zero at bit i, we can recover a secret key bit i. The probability of faulting a tape that is used in an online execution, and belongs to an unopened party, so that a secret key bit may be recovered is $\tau/(Mn)$. For the L1 parameter set, this requires about 800 signature queries to recover each secret key bit (with probability close to 1).

`IR` If the response is changed to output the secret state of all n parties in MPC instance $j \in \mathcal{C}$ instead of only $n - 1$, this is an invalid response that leaks the secret key. Whether this is possible with a fault attack seems highly dependent on details of the implementation.

4 Formal Treatment of Hedged Signatures

In this section, we give formal definitions for a hedged signature scheme and its security notion, based on Bellare–Tackmann's *nonce-based signatures* [BT16, §5] and Bellare–Poettering–Stebila's *de-randomized signatures* [BPS16, §5.1]. Then we define our new security notion for hedged Fiat–Shamir signature schemes, which guarantees resilience against 1-bit faults on function inputs/outputs.

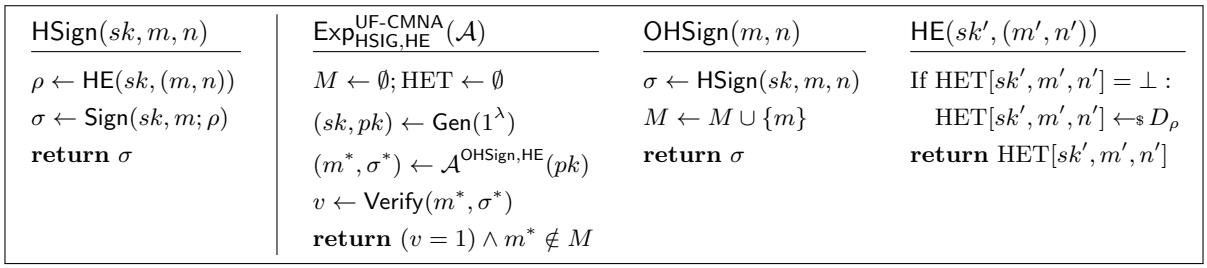

Figure 5: Hedged signature scheme $\mathsf{HSIG} = \mathbf{R2H}[\mathsf{SIG}, \mathsf{HE}] = (\mathsf{Gen}, \mathsf{HSign}, \mathsf{Verify})$ and UF-CMNA experiment. Key generation and verification are unchanged.

4.1 Security of Hedged Signature Schemes

We now consider a simple transformation **R2H**, which converts a randomized signature scheme to a so-called "hedged" one, and its security notion UF-CMNA (unforgeability against chosen message and nonce attacks). See Fig. 5 for the full details. Parts of the transformation appear in the literature independently, but by combining them, we can model the concrete hedged signature schemes of interest. We now describe the differences and similarities between **R2H** and the transformations that appeared in previous works.

- On one hand, a hedged signing algorithm HSign takes a *nonce* n along with a message m, and derives the randomness $\rho \in D_\rho$ (of length ℓ_ρ bits) with a *hedged extractor* HE with $(sk, (m, n))$ as input. We do not specify how the nonces are generated here, but in practice they are the output of a pseudorandom number generator. As we will see soon, low entropy nonces do not really degrade the security of hedged signatures as long as the underlying randomized signature scheme is secure. The hedged construction we presented is essentially based on the approach taken in [BT16]. Note that HE is in practice a cryptographic hash function, that we will model as a random oracle. Therefore **R2H** is also similar to the construction "randomized-encrypt-with-hash" for public-key encryption of Bellare et al. [BBN+09]. Bellare and Tackmann also considered HE in the standard model, but we focus on random oracle version in this paper since the Fiat–Shamir transform already relies on the random oracle model.
- On the other hand, we use the signing key sk as the key for the hedged extractor, whereas Bellare and Tackmann used a separately generated key (which they called the "seed"), that must be stored with sk. We chose to do so in order to model concrete hedged Fiat–Shamir type schemes, such as XEdDSA and Picnic2. In fact, the security of the deterministic construction that hashes sk and m to derive ρ (with no nonce) was formally treated by Bellare–Poettering–Stebila [BPS16], and our security proof in the next section extends their result.
- Moreover, the signing oracle OHSign in our UF-CMNA experiment takes m and n as input adaptively chosen by the adversary $\mathcal{A}$. This can be regarded as the strongest instantiation of the oracle provided in [BT16], where nonces are derived via what they call a nonce generator (NG). Indeed, one of their results for nonce-based signatures (Theorem 5.1) does not impose any restrictions on NG, and it implicitly allows adversaries to fully control how the nonces are chosen in the signing oracle.

Now we formally define a security notion for hedged signature schemes, as a natural extension of the standard UF-CMA security definition. We also give a tweaked version of Theorem 4 in [BPS16], where they only consider the signing oracle that doesn't take adversarially chosen nonces. Note that Lemma 3 applies to *any* secure signature schemes and hence it may be of independent interest. We give a proof in Appendix A.2 for completeness.

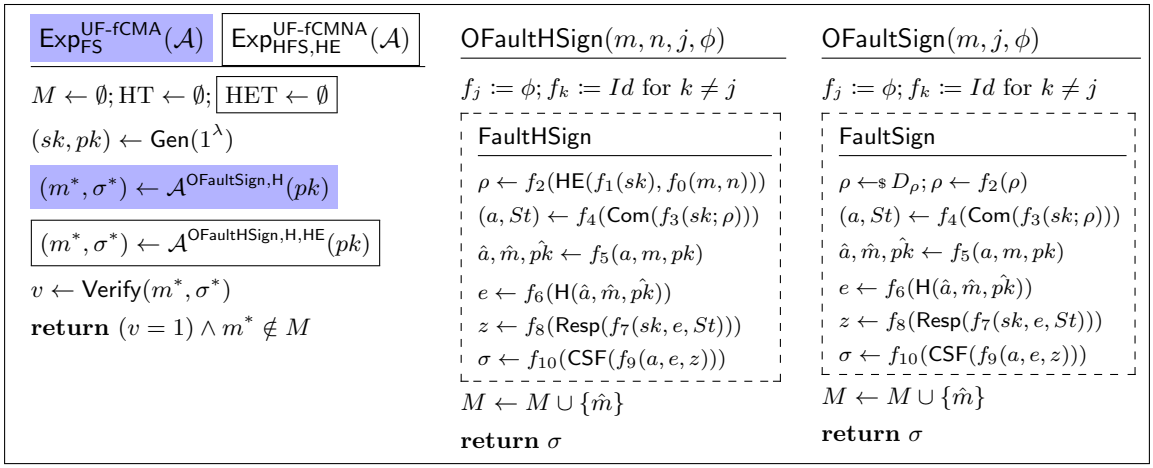

Figure 6: UF-fCMNA and UF-fCMA security experiments and faulty signing oracles for both hedged (HFS) and plain (FS) Fiat–Shamir signature schemes. Id stands for the identity function. The function H and HE (not shown), are the same as in Fig. 4 and Fig. 5, respectively. A dashed box indicates that the instructions inside correspond to the actual faulty signing operation.

Definition 9 (UF-CMNA)**.** A hedged signature scheme $\mathsf{HSIG} = (\mathsf{Gen}, \mathsf{HSign}, \mathsf{Verify})$ is said to be UF-CMNA secure in the random oracle model, if for any probabilistic polynomial time adversary $\mathcal{A}$, its advantage

$$\mathbf{Adv}^{\mathsf{UF\text{-}CMNA}}_{\mathsf{HSIG,HE}}(\mathcal{A}) \coloneqq \Pr\left[\mathsf{Exp}^{\mathsf{UF\text{-}CMNA}}_{\mathsf{HSIG,HE}}(\mathcal{A}) = 1\right]$$

is negligible in security parameter λ, where $\mathsf{Exp}^{\mathsf{UF\text{-}CMNA}}_{\mathsf{HSIG,HE}}(\mathcal{A})$ is described in Fig. 5.

Lemma 3 (UF-CMA $\rightarrow$ UF-CMNA)**.** *Let* $\mathsf{SIG} \coloneqq (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *be a randomized digital signature scheme, and let* $\mathsf{HSIG} \coloneqq \mathbf{R2H}[\mathsf{SIG}, \mathsf{HE}] = (\mathsf{Gen}, \mathsf{HSign}, \mathsf{Verify})$ *be the corresponding hedged signature scheme with* HE *modeled as a random oracle. If* SIG *is* UF-CMA *secure, then* HSIG *is* UF-CMNA *secure. Concretely, given a* UF-CMNA *adversary* $\mathcal{A}$ *against* HSIG *running in time* t*, and making at most* Q_s *queries to* OHSign *and* Q_{he} *queries to* HE*, one can construct an adversary* $\mathcal{B}$ *against* SIG *such that*

$$\mathbf{Adv}^{\mathsf{UF\text{-}CMNA}}_{\mathsf{HSIG,HE}}(\mathcal{A}) \leq 2 \cdot \mathbf{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{B}),$$

where $\mathcal{B}$ *makes at most* Q_s *queries to its signing oracle, and has running time* t *plus* Q_{he} *invocations of* Sign *and* Verify*.*

4.2 Security of Hedged FS Type Signature Schemes Against Fault Adversaries

1-bit Transient Fault on Function Input/Output To model transient fault attackers on data flow, recall that we consider the following 1-bit tampering functions:

$$\texttt{flip\_bit}_i(x) : \text{Does a logical negation of the } i\text{-th bit of } x$$
$$\texttt{set\_bit}_{i,b}(x) : \text{Sets the } i\text{-th bit of } x \text{ to } b$$

Using $\texttt{flip\_bit}_i(x)$ (for instance, with a random position i), we can model a typical bit-flip induced from fault injection to the memory cells, CPU register values, or data buses of the target device. Beyond faults, we also wish to capture the case in which the randomness has a 1-bit bias, which has been shown to be a serious threat for some Fiat–Shamir type signatures [AFG+14]. We can model this using $\texttt{set\_bit}_{i,b}$: when this function is applied to ρ, we can ensure that the first bit of ρ is "stuck" at zero by setting $i = 0$ and $b = 0$ to model 1-bit bias. Moreover, `set_bit` is a typical way to achieve so-called ineffective fault attacks [Cla07, DEK+18]. Our formalization covers many fault attacks found in the surveyed literature in Section 3, as they rely only on low precision faults like random bit flips of the function input or output.

As a notable difference between our fault adversary model and actual attacks, some surveyed papers caused faults on several bits/bytes of function input or output when performing fault attack experiments. This is *not* to take advantage of multiple-bit faults, but rather because reliably causing a fault on a specific target memory cell is difficult in practical experiments. In fact, the attacks we classified as `SSND` and `LRB` can be achieved with uncontrolled 1-bit flip faults, and hence our model at least seems to capture the essence of previous attacks exploiting the deterministic nature of signing.

A natural generalization is to allow `set_bit` to work on multiple bits, for example to model word faults, or word zeroing faults. We can also model stronger attacks that are uncommon in the literature, such as setting words to arbitrary values. However, we focus on 1-bit faults in this paper as a first attempt to perform the formal analyses. We leave the security analysis against multi-bit faults for future work. (Along with other generalizations listed in Section 8.)

Equipping UF-CMNA Adversaries with Faults Now we are ready to define security against fault adversaries using the above tampering functions. In Fig. 6, we give the modified hedged signing oracle $\mathsf{OFaultHSign}$, which additionally takes a tampering function $\phi \in \{\texttt{set\_bit}_{i,b}, \texttt{flip\_bit}_i, Id\}$ and $j \in [0, 10]$ as input, where Id is the identity function. This way, the adversary can specify for each query the tampering function (ϕ) as well as the target input/output position (j) within the signing operation to be faulted. For example, when $j = 6$, ϕ is applied to the output of the hash function H, and when $j = 5$ it is applied to the input to H. The other positions are not faulted. Notice that we also allow the adversary to set $\phi := Id$ in arbitrary signing queries, so $\mathsf{OFaultHSign}$ includes the behavior of the non-faulty oracle OHSign as a special case.
A generalization we considered but decided against, is allowing faults on multiple wire values per sign query. The combinatorial complexity of security analysis in this setting is daunting, and we did not find this to be relevant in practice, based on our survey of practical attacks.

Definition 10 (UF-fCMNA)**.** A hedged Fiat–Shamir signature scheme

$$\mathsf{HFS} := \mathbf{R2H}[\mathbf{FS}[\mathsf{ID}, \mathsf{CSF}], \mathsf{HE}] = (\mathsf{Gen}, \mathsf{HSign}, \mathsf{Verify})$$

is said to be F-UF-fCMNA secure, if for any probabilistic polynomial time adversary $\mathcal{A}$ who makes queries to $\mathsf{OFaultHSign}$ with a fault function $f_j \in F \subseteq \{f_0, \ldots, f_{10}\}$ for each query (called F-adversary), its advantage

$$\mathbf{Adv}^{\mathsf{UF\text{-}fCMNA}}_{\mathsf{HFS},\mathsf{HE}}(\mathcal{A}) := \Pr\left[\mathsf{Exp}^{\mathsf{UF\text{-}fCMNA}}_{\mathsf{HFS},\mathsf{HE}}(\mathcal{A}) = 1\right]$$

is negligible in security parameter λ, where $\mathsf{Exp}^{\mathsf{UF\text{-}fCMNA}}_{\mathsf{HFS},\mathsf{HE}}(\mathcal{A})$ is described in Fig. 6.

In the next section, we also use the following intermediate security notion, which essentially guarantees the security of plain randomized Fiat–Shamir signature scheme against fault adversaries.

Definition 11 (UF-fCMA)**.** A Fiat–Shamir signature scheme

$$\mathsf{FS} := \mathbf{FS}[\mathsf{ID}, \mathsf{CSF}] = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$$

is said to be F-UF-fCMA secure, if for any probabilistic polynomial time adversary $\mathcal{A}$ who makes queries to $\mathsf{OFaultSign}$ with a fault function $f_j \in F \subseteq \{f_2, \ldots, f_{10}\}$ per each query (called F-adversary), its advantage

$$\mathbf{Adv}^{\mathsf{UF\text{-}fCMA}}_{\mathsf{FS}}(\mathcal{A}) := \Pr\left[\mathsf{Exp}^{\mathsf{UF\text{-}fCMA}}_{\mathsf{FS}}(\mathcal{A}) = 1\right]$$

is negligible in security parameter λ, where $\mathsf{Exp}^{\mathsf{UF\text{-}fCMA}}_{\mathsf{FS}}(\mathcal{A})$ is described in Fig. 6.

We now give several remarks regarding our security model.

Trivial Faults on the Root Input Wire Values We remark the existence of two faults on the left most input wires in Fig. 1, which we do not explicitly consider in our model, but its (in)security can be proven trivially. First, faulting message m before it is loaded by the signing oracle can be regarded as a situation where the adversary queries a faulty message $\hat{m}$ to begin with, since the oracle stores $\hat{m}$ in M. Hence we can just treat such a query as one to non-faulty signing oracle (OSign). Second, the adversary could easily

recover the entire secret key after roughly $|sk|$ signing queries by injecting `set_bit` faults to sk before it is loaded by the signing oracle, and the faulty secret key $\hat{sk}$ is globally used throughout the signing operation: for example, if the most significant bit of sk is set to 0 at the very beginning of signing and its output still passed the verification, then the adversary can conclude that sk has 0 in the most significant bit with high probability. In doing so, the adversary iteratively recovers sk bit-by-bit if the fault is transient. The attack above is essentially a well-known impossibility result by Gennaro et al. [GLM+04] and such an attack can be practically achieved with ineffective faults. To overcome this issue, one would require an additional strict assumption on the upper-bound of faulty signing queries [DFMV17], or the signing algorithm needs to have some sophisticated features like self-destruct or key-updating mechanisms, which, however, are not yet widely implemented in real-world systems and are beyond the scope of this paper.

Winning Condition of Fault Adversaries As described in Fig. 6, the UF-fCMNA experiment keeps track of possibly faulty messages $\hat{m}$ instead of queried messages m, and it does not regard σ^* as valid forgery if it verifies with $\hat{m}$ that $\mathcal{A}$ caused in prior queries. This may appear artificial, but we introduced this condition to rule out a trivial forgery "attack": if the experiment only keeps track of queried message m_i in i-th query, and adversaries target f_5 at m_i as hash input, they obtain a valid signature $\hat{\sigma}_i$ on message $\hat{m}_i$, yet $\hat{m}_i$ is not stored in a set of queried messages M. Hence the adversary can trivially win UF-fCMNA game by just submitting $(\hat{\sigma}_i, \hat{m}_i)$, which of course verifies. This is not an actual attack, since what $\mathcal{A}$ does there is essentially asking for a signature on $\hat{m}_i$ from the signing oracle, and hence outputting such a signature as forgery should not be considered as a meaningful threat.

Note that the OFaultHSign oracle in Fig. 6 stores all queried messages in the same set M, whether the adversary $\mathcal{A}$ decides to inject a fault (i.e., $\phi \in \{\texttt{set\_bit}_{i,b}, \texttt{flip\_bit}_i\}$) or not (i.e., $\phi := Id$), and so a forgery (m^*, σ^*) output by $\mathcal{A}$ is *not* considered valid even if m^* was only queried to OFaultHSign to obtain a faulty invalid signature. For some signature algorithms and fault types this is required; for example with Fiat–Shamir type signatures (derived from a commitment recoverable identification [KLS18]), one can query OFaultHSign to get a signature (e, z) with a single bit-flip in z, and create a valid forgery by unflipping the bit.

Validity of Oracle Output The signature output by OFaultHSign does not need to verify, but it may need to be well-formed in some way. Typically we show with a hybrid argument that OFaultHSign can be simulated without use of the private key, in a similar way to OHSign. In order for simulated outputs of OFaultHSign to be indistinguishable from real outputs, simulated signatures must be correctly distributed. In [BDF+14, CM09], the security proof shows that the faulty signature is statistically close to a value drawn from the uniform distribution, so OFaultHSign can output a random value. For the Fiat–Shamir type signature schemes we study this is not the case, for some fault types the real output of OFaultHSign verifies with an appropriately faulted hash function, and our proofs must take care to maintain these properties when simulating OFaultHSign.

UF-KOA against Fault Adversaries Finally, note that we do not explicitly define the notion of UF-KOA security under fault attacks. Since we only consider faults during signing, and there is no signing oracle a in key-only attack, security against fault attacks is immediate in the context of UF-KOA security.

5 Security Analysis of Generic Hedged Fiat–Shamir Type Signatures Against Fault Attacks

In this section we establish the (in)security of the class of hedged Fiat–Shamir signatures schemes **R2H**[**FS**[ID, CSF], HE]. Table 2 summarizes these results (and also those related to XEdDSA and Picnic2, presented in §6 and §7). We give here a short overview of the main intuition behind the results in Table 2: f_0 faults (on the (message, nonce) pair which is input to the hedged-extractor) cannot be tolerated since they allow the adversary to get two signatures with the same randomness. On the other hand f_1 faults (on the secret key input to the hedged-extractor) can be tolerated since they do not significantly change the distribution input to the hedged-extractor. If the adversary faults the output of the hedged extractor (using f_2), we cannot prove security in general (and we can list concrete attacks e.g., against the Schnorr signature schemes), but

Table 2: Summary of results for UF-fCMNA security of the hedged Fiat–Shamir type construction, for all fault types. ✓ indicates a proof of UF-fCMNA security, and ✗ indicates an attack or counterexample.

<table>
<tr><th>Fault type</th><th colspan="2">ID is subset-revealing</th><th>ID not subset-revealing</th><th>XEdDSA</th><th>Picnic2</th></tr>
<tr><td>f_0</td><td colspan="3">✗ Lemma 11</td><td>✗</td><td>✗</td></tr>
<tr><td>f_1</td><td colspan="3">✓ Lemma 4</td><td>✓ Corollary 1</td><td>✓ Corollary 3</td></tr>
<tr><td>f_2</td><td colspan="3">✗ Lemma 13</td><td>✗</td><td>✓ Lemma 19</td></tr>
<tr><td>f_3</td><td colspan="3">✗ Lemma 12</td><td>✗</td><td>✗ §7</td></tr>
<tr><td>f_4</td><td colspan="2">✓ Lemma 10</td><td>✗ Lemma 15</td><td>✗</td><td rowspan="4">✓ Corollary 3</td></tr>
<tr><td>f_5</td><td colspan="3">✓ Lemma 7</td><td rowspan="4">✓ Corollary 1</td></tr>
<tr><td>f_6</td><td colspan="3">✓ Lemma 8</td></tr>
<tr><td>f_7</td><td colspan="2">✓ Lemma 9</td><td>✗ Lemma 14</td></tr>
<tr><td>f_8, f_9, f_{10}</td><td colspan="3">✓ Lemma 6</td></tr>
</table>

we can prove security for the specific case of Picnic2, since the output of the hedged-extractor is not used directly, but is given as input to a PRG – thus the small bias is "absorbed" by PRG security. We remark that, while present, this attack is much less devastating than the large randomness bias LRB attack on deterministic schemes (described in Section 3). With the LRB attack, the adversary only needs two signatures to recover the full key, while the attack we will show on Schnorr signature requires a significant amount of faulty biased signatures as input in practice. This indicates that hedged constructions do, to some extent, mitigate the effect of faults on the synthetic randomness.

The hedged approach does not help when the adversary faults the input to the commitment function (via f_3), since in this case the adversary can attempt to set the bits of the secret key one at the time and check if the output signature is valid or not. Note that in some kinds of ID schemes like Schnorr (known as *input-delayed* protocols [CPS+16]) the secret key is not used in the commitment function. Faulting the input of the commitment function can still lead to insecurity, e.g., in Schnorr the adversary can bias the randomness, which in turns leads to a total break of the signature scheme. Next, the adversary can fault the output of the commitment function (via f_4): this leads to insecurity in general, e.g., in Schnorr this also leads to randomness bias. However, for a large class of ID schemes (which we call *subset-revealing*), including Picnic2, this fault does not lead to insecurity: intuitively either the adversary faults something that will be output as part of the response (which can easily be simulated by learning a non-faulty signature and then applying the fault on the result), or it is not part of the output and therefore irrelevant. Attacking the input or the output of the random oracle used to derive the challenge (f_5 and f_6) does not lead to insecurity, since the distribution of the random oracle does not change due to the fault (note that this would not be the case for deterministic signatures, where this kind of fault would be fatal). Faults against the input of the response function (via f_7) can break non-subset revealing signatures (once again, we can show that this fault can be used to break Schnorr signatures), but do not help the adversary in the case of a subset-revealing signature like Picnic2: similar to the case of f_4 faults, we use the fact that if the response function only outputs subsets of its input, faulting part of the input either has no effect or can be efficiently simulated given a non-faulty signature. Similarly, faults against the output of the response function or the input/output of the serialization function (fault types f_8, f_9, f_{10}) can also be easily simulated from a non-faulty signature.

We expand this high-level intuition into full proofs by carefully measuring the concrete security loss in the reductions which is introduced by the different kind of faults. More precisely, we present a concrete reduction from UF-KOA to $\{f_1, f_4, \ldots, f_{10}\}$-UF-fCMNA security for schemes derived from subset-revealing ID schemes, and to $\{f_1, f_5, f_6, f_8, f_9, f_{10}\}$-UF-fCMNA when ID is non-subset-revealing. Our theorems generalize and adapt results from [BPS16] and [KMP16] without introducing significant additional concrete security loss. Then in Section 5.7, we describe attacks for the remaining fault types (f_0, f_2 and f_3), completely characterizing the security of generic $\mathbf{R2H}[\mathbf{FS}[\mathsf{ID}, \mathsf{CSF}], \mathsf{HE}]$ signature schemes for fault types $f_0, \ldots, f_{10}$. As we will show in Section 7, for specific signature schemes we may be able to prove a stronger result, we show that Picnic2 is secure for fault type f_2, where the generic result does not hold.

5.1 Main Positive Result

Theorem 1 (UF-KOA $\to$ UF-fCMNA)**.** *Let* ID *be a canonical identification protocol and* CSF *be a canonical serialization function for* ID*. Suppose* ID *satisfies the same properties as in Lemma 2 and it is subset revealing, and moreover, let us assume that* $\mathcal{A}$ *does not query the same* (m, n) *pair to* OFaultHSign *more than once. Then if* $\mathsf{FS} := \mathbf{FS}[\mathsf{ID}, \mathsf{CSF}]$ *is* UF-KOA *secure,* $\mathsf{HFS} := \mathbf{R2H}[\mathsf{FS}, \mathsf{HE}]$ *is* $\{f_1, f_4, \ldots, f_{10}\}$-UF-fCMNA *secure in the random oracle model. Concretely, given* $\{f_1, f_4, \ldots, f_{10}\}$*-adversary* $\mathcal{A}$ *against* HFS *running in time* t*, and making at most* Q_s *queries to* OFaultHSign*,* Q_h *queries to* H *and* Q_{he} *queries to* HE*, one can construct another adversary* $\mathcal{B}$ *against* FS *such that*

$$\mathbf{Adv}^{\mathsf{UF\text{-}fCMNA}}_{\mathsf{HFS},\mathsf{HE}}(\mathcal{A}) \le 2 \cdot \left(\mathbf{Adv}^{\mathsf{UF\text{-}KOA}}_{\mathsf{FS}}(\mathcal{B}) + \frac{(Q_s + Q_h)Q_s}{2^{\alpha-1}} + Q_s \cdot \epsilon_{HVZK} \right),$$

where $\mathcal{B}$ *makes at most* Q_h *queries to its hash oracle, and has running time* t *plus* $Q_{he} \cdot |sk|$ *invocations of* Sign *and* Verify *of* FS*. Moreover, if we do not assume the subset-revealing property of* ID *and assume all the other conditions above, then we have that* HFS *is* $\{f_1, f_5, f_6, f_8, f_9, f_{10}\}$-UF-fCMNA *secure.*

Proof. The proof is two-fold. See Lemmas 4 and 5. □

For the rest of this section we will assume that ID satisfies the properties in Lemma 2. As a first step, we give a reduction from UF-fCMA to UF-fCMNA security, and then we later give a reduction from UF-KOA to UF-fCMA. We observe that the UF-CMA-to-UF-CMNA reduction in Lemma 3 is mostly preserved, even in the presence of 1-bit faults on sk as a hedged extractor key. However, our proof shows that such a fault does affect the running time of the adversary because the reduction algorithm needs to go through all secret key candidates queried to random oracle *and* their faulty bit-flipped variants.

Lemma 4 (F-UF-fCMA $\to$ $F \cup \{f_1\}$-UF-fCMNA)**.** *Suppose the fault adversary* $\mathcal{A}$ *does not query the same* (m, n) *pair to* OFaultHSign *more than once. If* $\mathsf{FS} := \mathbf{FS}[\mathsf{ID}, \mathsf{CSF}]$ *is* F-UF-fCMA *secure, then* $\mathsf{HFS} := \mathbf{R2H}[\mathsf{FS}, \mathsf{HE}]$ *is* F'-UF-fCMNA *secure in the random oracle model, where* $F' = F \cup \{f_1\}$*. Concretely, given an* F'*-adversary* $\mathcal{A}$ *against* HFS *running in time* t*, and making at most* Q_s *queries to* OFaultHSign*,* Q_h *queries to* H *and* Q_{he} *queries to* HE*, one can construct* F*-adversary* $\mathcal{B}$ *against* FS *such that*

$$\mathbf{Adv}^{\mathsf{UF\text{-}fCMNA}}_{\mathsf{HFS},\mathsf{HE}}(\mathcal{A}) \le 2 \cdot \mathbf{Adv}^{\mathsf{UF\text{-}fCMA}}_{\mathsf{FS}}(\mathcal{B}),$$

where $\mathcal{B}$ *makes at most* Q_s *queries to its signing oracle* OFaultSign *and* Q_h *queries to its hash oracle, and has running time* $t' \approx t + Q_{he} \cdot |sk|$*.*

Proof. We present the code-based game-playing proof [BR06], and the basic structure of our proof closely resembles the reduction from UUF-CMA to UF-CMA for deterministic signature schemes [BPS16]. Let us consider two *identical-until-bad* games described in Fig. 7: $\mathrm{G}_0(\mathcal{A})$ (without boxed code) and $\mathrm{G}_1(\mathcal{A})$ (with boxed code). We assume wlog that $\mathcal{A}$ does not repeat the same HE query. Notice that $\mathrm{G}_0(\mathcal{A})$ is identical to $\mathsf{Exp}^{\mathsf{UF\text{-}fCMNA}}_{\mathsf{HFS},\mathsf{HE}}(\mathcal{A})$ in Fig. 6 (on condition that $\mathcal{A}$ is not allowed to query the same (m, n) to OFaultHSign), and therefore $\mathbf{Adv}^{\mathsf{UF\text{-}fCMNA}}_{\mathsf{HSIG},\mathsf{HE}}(\mathcal{A}) = \Pr[\mathrm{G}_0(\mathcal{A})]$. Now we obtain the following by simple transformation (where the inequality follows from the fundamental lemma of game-playing [BR06]):

$$\begin{aligned} \mathbf{Adv}^{\mathsf{UF\text{-}fCMNA}}_{\mathsf{HSIG},\mathsf{HE}}(\mathcal{A}) = \Pr[\mathrm{G}_0(\mathcal{A})] &= \Pr[\mathrm{G}_1(\mathcal{A})] + (\Pr[\mathrm{G}_0(\mathcal{A})] - \Pr[\mathrm{G}_1(\mathcal{A})]) \\ &\le \Pr[\mathrm{G}_1(\mathcal{A})] + \Pr[\mathrm{G}_1(\mathcal{A}) \text{ sets } \mathsf{bad}]. \end{aligned}$$

Our goal is to construct the adversary $\mathcal{B}$ breaking UF-fCMA security of a plain randomized Fiat–Shamir scheme FS such that

$$\Pr[\mathrm{G}_1(\mathcal{A})] \le \mathbf{Adv}^{\mathsf{UF\text{-}fCMA}}_{\mathsf{FS}}(\mathcal{B}) \quad \text{and} \quad \Pr[\mathrm{G}_1(\mathcal{A}) \text{ sets } \mathsf{bad}] \le \mathbf{Adv}^{\mathsf{UF\text{-}fCMA}}_{\mathsf{FS}}(\mathcal{B}).$$

We now construct $\mathcal{B}$ as follows. The full description of the algorithm is given in Fig. 8.

Preparation of Public Key Upon receiving pk in UF-fCMA game, $\mathcal{B}$ forwards pk to $\mathcal{A}$.

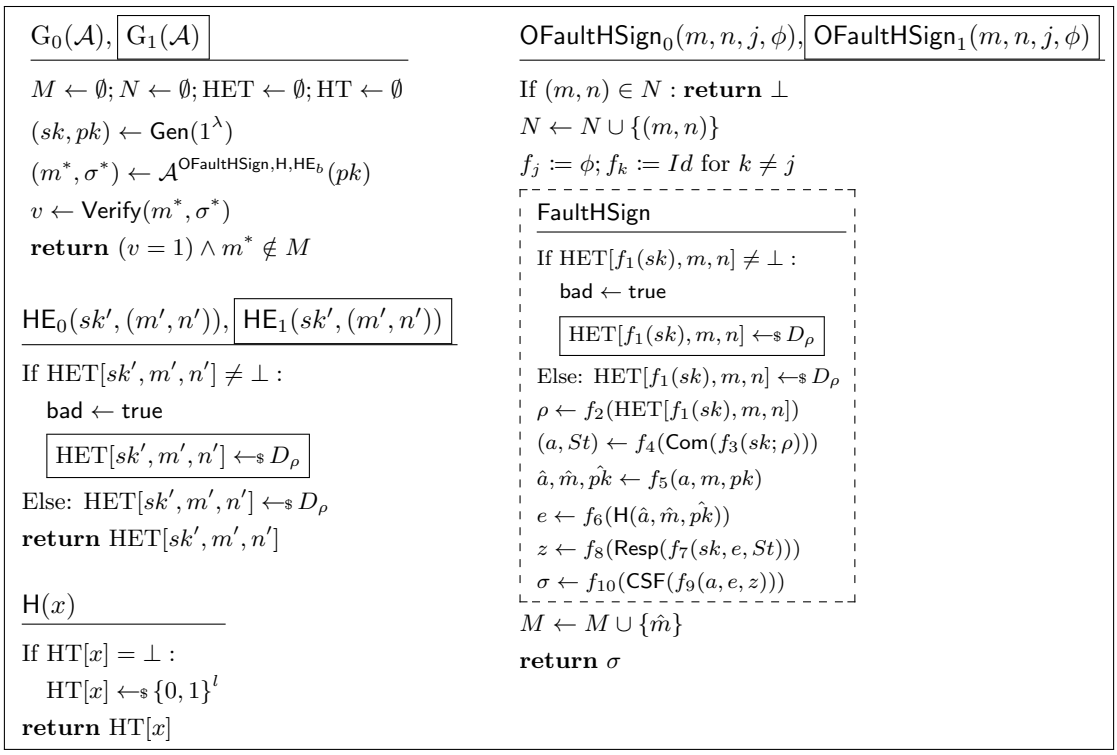

Figure 7: Two identical-until-bad games $\mathrm{G}_0(\mathcal{A})$ and $\mathrm{G}_1(\mathcal{A})$. The game $\mathrm{G}_1(\mathcal{A})$ executes the boxed code, while $\mathrm{G}_0(\mathcal{A})$ does not. The dashed box indicates that the instructions inside correspond to the actual faulty signing operation.

Simulation of Oracle Queries The adversary $\mathcal{B}$ perfectly simulates $\mathcal{A}$'s view in $\mathrm{G}_1(\mathcal{A})$ as follows.

- Sim-$\mathsf{OFaultHSign}_1$ simulates $\mathsf{OFaultHSign}_1$ by forwarding faulty signing queries to $\mathsf{OFaultSign}$, the oracle in UF-fCMA game. Notice that this simulation is perfect since both $\mathsf{OFaultHSign}_1$ and $\mathsf{OFaultSign}$ use freshly generated randomness for every signing query.
- Sim-HE_1 simulates the HE_1 but keeps track of candidate secret keys queried by $\mathcal{A}$ in S.
- Sim-H directly returns the result of hash queries in UF-fCMA game.

Evaluation of $\mathcal{B}$'s Success Probability Suppose $\mathcal{A}$ wins the game $\mathrm{G}_1(\mathcal{A})$, i.e., $\mathcal{A}$ outputs its forgery (m', σ') that passes the verification and $m' \notin M = \{\hat{m}_i : i \in [Q_s]\}$. This means that the reconstructed transcript $(a', e', z') \leftarrow \mathsf{CDF}(\sigma', pk)$ satisfies

$$\mathsf{V}(a', e', z', pk) = 1 \quad \text{and} \quad \mathsf{H}(a', m', pk) = e'.$$

Also note that the sets of (possibly faulty) messages M and random oracle entries are both identical between UF-fCMNA and UF-fCMA experiments. This implies (m', σ') is a valid forgery in UF-fCMA game as well, so that $\mathcal{B}$ can win UF-fCMA game, i.e.,

$$\Pr[\mathrm{G}_1(\mathcal{A})] \leq \mathbf{Adv}^{\mathsf{UF\text{-}fCMA}}_{\mathsf{FS}}(\mathcal{B}).$$

Otherwise, $\mathcal{B}$ picks some message $m^* \notin M$ and tries to sign with all secret key candidates sk^* *and* "faulty bit-flipped candidates" (i.e., $\mathtt{flip\_bit}_i(sk^*)$ for $i \in [|sk|]$). Below we will see why this strategy guarantees that $\mathcal{B}$ generates a valid forgery as long as bad is set in $\mathrm{G}_1(\mathcal{A})$, i.e.,

$$\Pr[\mathrm{G}_1(\mathcal{A}) \text{ sets } \mathsf{bad}] \leq \mathbf{Adv}^{\mathsf{UF\text{-}fCMA}}_{\mathsf{FS}}(\mathcal{B}).$$

$\mathcal{B}^{\mathsf{OFaultSign},\mathsf{H}}(pk)$

$M \leftarrow \emptyset; N \leftarrow \emptyset; \mathrm{HET} \leftarrow \emptyset; S \leftarrow \emptyset$
$(m', \sigma') \leftarrow \mathcal{A}^{\mathsf{Sim\text{-}OFaultHSign}_1,\mathsf{Sim\text{-}H},\mathsf{Sim\text{-}HE}_1}(pk)$
If $\mathsf{Verify}(m', \sigma') = 1 \wedge m' \notin M$:
  return (m', σ')
Pick some $m^* \notin M$
$\rho^* \leftarrow\!\!\$\ D_\rho$
For $sk^* \in S$:
  $\sigma^* \leftarrow \mathsf{Sign}(sk^*, m^*; \rho^*)$
  If $\mathsf{Verify}(m^*, \sigma^*) = 1$:
    return (m^*, σ^*)
  For $i \in [|sk|]$:
    $\sigma^* \leftarrow \mathsf{Sign}(\mathtt{flip\_bit}_i(sk^*), m^*; \rho^*)$
    If $\mathsf{Verify}(m^*, \sigma^*) = 1$:
      return (m^*, σ^*)
return $(\perp, \perp)$

$\mathsf{Sim\text{-}OFaultHSign}_1(m, n, j, \phi)$

If $(m, n) \in N$: **return** $\perp$
$N \leftarrow N \cup \{(m, n)\}$
$\sigma \leftarrow \mathsf{OFaultSign}(m, j, \phi)$
$M \leftarrow M \cup \{\hat{m}\}$
return σ

$\mathsf{Sim\text{-}HE}_1(sk', (m', n'))$

$S \leftarrow S \cup \{sk'\}$
$\rho \leftarrow\!\!\$\ D_\rho$
return ρ

$\mathsf{Sim\text{-}H}(x)$

return $\mathsf{H}(x)$

Figure 8: Description of the reduction $\mathcal{B}$. If the adversary uses the fault f_5 that applies to m, M stores the faulty message $\hat{m}$, which is defined as described in Fig. 7.

Here we show that if bad is set either by $\mathsf{OFaultHSign}_1$ or HE_1, then $\mathcal{A}$ must have queried HE_1 with $(sk^*, (m^*, n^*))$ such that $sk^* = f_1(sk)$ for some $f_1 \in \{\mathtt{set\_bit}_{i,b}, \mathtt{flip\_bit}_i, Id\}$, and for some (m^*, n^*). First, if bad is set by $\mathsf{OFaultHSign}_1$, it must be due to a previous query to HE_1 with $(sk', (m', n'))$ such that $sk' = f_1(sk)$ holds, since we assumed that $\mathcal{A}$ is not allowed to repeat the same (m, n) query to OFaultHSign. Second, we consider the case bad is set by HE_1. Since we assumed that $\mathcal{A}$ does not repeat the same HE query, if a query $(sk', (m', n'))$ to HE_1 sets bad then $\mathrm{HET}[sk', m', n']$ must have been defined by $\mathsf{OFaultHSign}_1$. Here notice that we have $sk' = f_1(sk)$ for some f_1 due to the definition of $\mathsf{OFaultHSign}_1$. Thus if bad is set in $\mathrm{G}_1(\mathcal{A})$ then for some $sk^* \in S$ there exists a faulty function $f_1 \in \{\mathtt{set\_bit}_{i,b}, \mathtt{flip\_bit}_i, Id\}$ such that $sk^* = f_1(sk)$ holds. By checking the validity of signatures generated with each $sk^* \in S$ and its faulty version $\mathtt{flip\_bit}_i(sk^*)$, for each bit position i, $\mathcal{B}$ eventually finds the true secret key sk that leads to a valid forgery.

We finally observe that the total running time of $\mathcal{B}$ is upper bounded by $\mathcal{A}$'s running time t plus $Q_{he} \cdot |sk|$ times the running time of Sign and Verify due to the above double for-loop operations. □

Remarks. Our reduction above crucially relies upon the assumption that adversaries are not allowed to query the same (m, n) pair. Without this condition, OFaultHSign must return a faulty signature derived from the same randomness ρ if the same (m, n) is queried twice, and thus one could not simulate it using OFaultSign as an oracle, since OFaultSign uses the fresh randomness even if queried with the same message m. In fact, by allowing the same (m, n) query the hedged construction HFS degenerates to a deterministic scheme and thus the SSND or LRB type fault attacks would become possible as we saw in Section 3. For the same reason, once we allow the adversaries to mount a fault f_0 on (m, n) right before HE is invoked during the signing query, the security is completely compromised. We will revisit this issue as a negative result in Lemma 11.

Lemma 5 (UF-KOA → UF-fCMA)**.** *Suppose* ID *is subset revealing. If* $\mathsf{FS} := \mathbf{FS}[\mathsf{ID}, \mathsf{CSF}]$ *is* UF-KOA *secure, then* FS *is* $\{f_4, \ldots, f_{10}\}$-UF-fCMA *secure in the random oracle model. Concretely, given* $\{f_4, \ldots, f_{10}\}$*-adversary* $\mathcal{A}$ *against* FS *running in time* t*, and making at most* Q_s *queries to* OFaultSign*,* Q_h *queries to* H*, one can construct another adversary* $\mathcal{B}$ *against* FS *such that*

$$\mathbf{Adv}^{\mathsf{UF\text{-}fCMA}}_{\mathsf{FS}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{UF\text{-}KOA}}_{\mathsf{FS}}(\mathcal{B}) + \frac{(Q_s + Q_h)Q_s}{2^{\alpha-1}} + Q_s \cdot \epsilon_{HVZK},$$

where $\mathcal{B}$ *makes at most* Q_h *queries to its hash oracle, and has running time* t*. If we do not assume the subset-*

revealing property of ID *and assume all the other conditions above, then we have that* FS *is* $\{f_5, f_6, f_8, f_9, f_{10}\}$-UF-fCMA *secure.*

Proof. We obtain the results by putting together Lemmas 6 to 10 for FS derived from subset-revealing ID, and Lemmas 6 to 8 for FS derived from non-subset-revealing ID.

Our proof extends the UF-KOA-to-UF-CMA reduction in [KMP16]. We show that UF-KOA security of a randomized Fiat–Shamir signature scheme FS can be broken by a successful UF-fCMA adversary $\mathcal{A}$ by constructing an adversary $\mathcal{B}$ that uses $\mathcal{A}$ as a subroutine and simulates OFaultSign without using sk. We denote the random oracle and hash table in UF-fCMA experiment (resp. UF-KOA experiment) by H and HT (resp. H′ and HT′).

Preparation of Public Key Upon receiving pk in the UF-KOA game, $\mathcal{B}$ forwards pk to $\mathcal{A}$.

Simulation of Random Oracle Queries Upon receiving a random oracle query $\mathsf{H}(a, m, pk)$ from $\mathcal{A}$, $\mathcal{B}$ forwards the input (a, m, pk) to its own random oracle (H′ from the UF-KOA game) and provides $\mathcal{A}$ with the return value.

Simulation of Faulty Signing Queries Suppose $\mathcal{A}$ chooses to use a fault function f_{j_i} in each faulty signing oracle query $i \in [Q_s]$. Then $\mathcal{B}$ answers i-th query by simulating the signature on m_i (or $\hat{m}_i$ if $\mathcal{A}$ chooses to apply f_5 to the message as hash input) using only pk as described in the lemma for f_{j_i}. Notice that the simulations are independent except they share the random oracle H and the set M storing (possibly faulty) queried messages. The hash input $(\hat{a}_i, \hat{m}_i, \hat{pk})$ in each signature simulation has at least $(\alpha - 1)$ bits of min-entropy (see the simulation in Lemma 7). Because HT has at most $Q_h + Q_s$ existing entries, $\mathcal{B}$ fails to program the random oracle with probability at most $(Q_h + Q_s)/2^{\alpha-1}$ for each query. Moreover, $\mathcal{A}$ distinguishes the simulated signature from the one returned by the real signing oracle OFaultHSign with probability at most ϵ_{HVZK} for each query, since we use the special c/s/p-HVZK simulator $\mathcal{M}$ to derive a signature in every simulation.

Recalling that the number of signing queries is bounded by Q_s, and by a union bound, $\mathcal{A}$ overall distinguishes its simulated view from that in UF-fCMA game with probability at most

$$\frac{(Q_h + Q_s)Q_s}{2^{\alpha-1}} + Q_s \cdot \epsilon_{HVZK}.$$

Forgery Suppose that at the end of the experiment $\mathcal{A}$ outputs its forgery (m^*, σ^*) that verifies and $m^* \notin M = \{\hat{m}_i : i \in [Q_s]\}$. (Recall from Fig. 6 that M stores possibly faulty messages $\hat{m}_i$ here instead of queried messages m_i, and thus $\mathcal{A}$ cannot win the game by simply submitting a signature on some faulty message that has been used for random oracle programming.) This means that the reconstructed transcript $(a^*, e^*, z^*) \leftarrow \mathsf{CDF}(\sigma^*, pk)$ satisfies

$$\mathsf{V}(a^*, e^*, z^*, pk) = 1 \quad \text{and} \quad \mathsf{H}(a^*, m^*, pk) = e^*.$$

Here we can guarantee that the $\mathrm{HT}[a^*, m^*, pk]$ has not been programmed by signing oracle simulation since m^* is fresh, i.e., $m^* \notin M$. Hence we ensure that $e^* = \mathrm{HT}[a^*, m^*, pk]$ has been directly set by $\mathcal{A}$, and $e^* = \mathrm{HT}'[a^*, m^*, pk]$ holds due to the hash query simulation. This implies (m^*, σ^*) is a valid forgery in the UF-KOA game as well. □

5.2 Faulting Serialization Input/Output and Response Output

As a warm-up, we begin with the simplest analysis where faults do not have any meaningful impact on the signing oracle simulation. As we will show below, faulting with f_8, f_9 and f_{10} has no more security loss than the plain UF-KOA-to-UF-CMA reduction [KMP16] does. The proof is provided in Appendix A.3.

Lemma 6 (UF-KOA $\to \{f_8, f_9, f_{10}\}$-UF-fCMA)**.** *If* $\mathsf{FS} \coloneqq \mathbf{FS}[\mathsf{ID}, \mathsf{CSF}]$ *is* UF-KOA *secure, then* FS *is* $\{f_8, f_9, f_{10}\}$-UF-fCMA *secure in the random oracle model. Concretely, given an* $\{f_8, f_9, f_{10}\}$*-adversary* $\mathcal{A}$ *against* FS *running in time* t*, and making at most* Q_s *queries to* OFaultSign*,* Q_h *queries to* H*, one can construct another*

adversary $\mathcal{B}$ *against* FS *such that*

$$\mathbf{Adv}^{\mathsf{UF\text{-}fCMA}}_{\mathsf{FS}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{UF\text{-}KOA}}_{\mathsf{FS}}(\mathcal{B}) + \frac{(Q_s + Q_h)Q_s}{2^{\alpha}} + Q_s \cdot \epsilon_{HVZK},$$

where $\mathcal{B}$ *makes at most* Q_h *queries to its random oracle, and has running time* t*.*

Remark As we briefly remarked after Definition 8, Lemma 6 holds for any instantiation of serialization as long as CSF and CDF are efficiently computable.

5.3 Faulting Challenge Hash Input

Recall that f_5 is the fault type that allows the attacker to fault the input (a, m, pk) to the hash function used to compute the challenge. Here we prove that randomized Fiat–Shamir signature schemes are secure against this type of fault attack, under the same conditions required for the plain UF-KOA-to-UF-CMA reduction [KMP16]. Note that the proof of lemma below introduces a slight additional security loss compared to the plain UF-KOA-to-UF-CMA reduction because `set_bit` faults to the hash input increase the failure probability of random oracle programming.

Lemma 7 ($\mathsf{UF\text{-}KOA} \to \{f_5\}\text{-}\mathsf{UF\text{-}fCMA}$)**.** *If* $\mathsf{FS} := \mathbf{FS}[\mathsf{ID}, \mathsf{CSF}]$ *is* $\mathsf{UF\text{-}KOA}$ *secure, then* FS *is* $\{f_5\}\text{-}\mathsf{UF\text{-}fCMA}$ *secure in the random oracle model. Concretely, given an* $\{f_5\}$*-adversary* $\mathcal{A}$ *against* FS *running in time* t*, and making at most* Q_s *queries to* $\mathsf{OFaultSign}$ *and* Q_h *queries to* H*, one can construct another adversary* $\mathcal{B}$ *against* FS *such that*

$$\mathbf{Adv}^{\mathsf{UF\text{-}fCMA}}_{\mathsf{FS}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{UF\text{-}KOA}}_{\mathsf{FS}}(\mathcal{B}) + \frac{(Q_s + Q_h)Q_s}{2^{\alpha-1}} + Q_s \cdot \epsilon_{HVZK},$$

where $\mathcal{B}$ *makes at most* Q_h *queries to its hash oracle, and has running time* t

Proof. The overall proof structure follows Lemma 5. The preparation of the public key, simulation of hash queries, and forgery phase are the same as before. The proof diverges at signing oracle simulation.

Simulation of Faulty Signing Queries For each $i \in [Q_s]$, $\mathcal{B}$ answers a faulty signing request from $\mathcal{A}$ by simulating the signature on m_i as follows:

1. $\mathcal{B}$ first samples e_i from D_H uniformly at random.
2. $\mathcal{B}$ invokes the special c/s/p-HVZK simulator $\mathcal{M}$ of ID on input pk and e_i and on freshly generated randomness every time, to obtain an accepting transcript (a_i, e_i, z_i).
3. Let $(\hat{a}_i, \hat{m}_i, \hat{pk}) := f_5(a_i, m_i, pk)$. If $\mathrm{HT}[\hat{a}_i, \hat{m}_i, \hat{pk}]$ is already set (via $\mathcal{A}$'s previous hash or signing queries) and $\mathrm{HT}[\hat{a}_i, \hat{m}_i, \hat{pk}] \neq e_i$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ programs the random oracle so that
$$\mathrm{HT}[\hat{a}_i, \hat{m}_i, \hat{pk}] := e_i.$$
4. If random oracle programming is successful, $\mathcal{B}$ gives $\mathcal{A}$ a serialized transcript:
$$\sigma_i := \mathsf{CSF}(a_i, e_i, z_i).$$

Now we evaluate the probability that $\mathcal{A}$ distinguishes the simulated signing oracle above from the real one. There are essentially two cases.

- $\mathcal{B}$ aborts at the third step in the above simulation with probability at most $(Q_h + Q_s)/2^{\alpha-1}$ per query. First, (a_i, m_i, pk) has α bits of min-entropy (since m_i is chosen by the adversary and has zero bits, pk is public and has zero bits, and a_i has α bits due to α-bit min-entropy of ID). Now suppose f_5 is `set_bit`. Then the min-entropy of the faulty input to the hash function $(\hat{a}_i, \hat{m}_i, \hat{pk})$ is at least $(\alpha - 1)$ bits: if f_5 applies to a_i then the min-entropy of $\hat{a}_i$ is reduced to $(\alpha - 1)$ bits, and otherwise $(\hat{a}_i, \hat{m}_i, \hat{pk})$ still has α-bit min-entropy. If f_5 is `flip_bit` or Id (i.e., $\mathcal{A}$ chose *not* to mount any fault), $(\hat{a}_i, \hat{m}_i, \hat{pk})$ still has α bits of min-entropy. Finally, HT has at most $Q_h + Q_s$ existing entries and we thus obtain the upper bound of the overall probability that $\mathcal{B}$ fails to program the random oracle.

- After the fourth step, $\mathcal{A}$ distinguishes σ_i from the one returned by the real signing oracle $\mathsf{OFaultHSign}$ with probability at most ϵ_{HVZK} per query, since we used the special c/s/p-HVZK simulator to derive (a_i, e_i, z_i) and CSF is deterministic.

Recalling that the number of signing queries is bounded by Q_s, and by a union bound, $\mathcal{A}$ overall distinguishes its simulated view from that in $\mathsf{UF\text{-}fCMA}$ game with probability at most

$$\frac{(Q_h + Q_s)Q_s}{2^{\alpha-1}} + Q_s \cdot \epsilon_{HVZK}.$$

□

5.4 Faulting Challenge Hash Output

Recall that f_6 is the fault type that allows the attacker to fault the challenge hash function output, i.e., he can fault the bit string $e = \mathsf{H}(a, m, pk)$. We show that, unlike the fault with f_5, this type of fault does not introduce any additional loss in concrete security as long as the Resp function fails for invalid challenges outside the challenge space D_H.

Lemma 8 ($\mathsf{UF\text{-}KOA} \to \{f_6\}\text{-}\mathsf{UF\text{-}fCMA}$)**.** *If* $\mathsf{FS} \coloneqq \mathbf{FS}[\mathsf{ID}, \mathsf{CSF}]$ *is* $\mathsf{UF\text{-}KOA}$ *secure, then* FS *is* $\{f_6\}$-$\mathsf{UF\text{-}fCMA}$ *secure in the random oracle model. Concretely, given an* $\{f_6\}$*-adversary* $\mathcal{A}$ *against* FS *running in time* t*, making at most* Q_s *queries to* $\mathsf{OFaultSign}$ *and* Q_h *queries to* H*, one can construct another adversary* $\mathcal{B}$ *against* FS *such that*

$$\mathbf{Adv}^{\mathsf{UF\text{-}fCMA}}_{\mathsf{FS}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{UF\text{-}KOA}}_{\mathsf{FS}}(\mathcal{B}) + \frac{(Q_s + Q_h)Q_s}{2^{\alpha}} + Q_s \cdot \epsilon_{HVZK},$$

where $\mathcal{B}$ *makes at most* Q_h *queries to its hash oracle, and has running time* t*.*

Proof. The overall proof structure follows Lemma 5. The preparation of public key, simulation of hash queries and forgery phase are same as before. The proof diverges at signing oracle simulation.

Simulation of Faulty Signing Queries For each $i \in [Q_s]$, $\mathcal{B}$ answers a faulty signing request from $\mathcal{A}$ by simulating the signature on m_i as follows:

1. $\mathcal{B}$ first samples e_i from D_H uniformly at random.
2. Let $\tilde{e}_i \coloneqq f_6(e_i)$ be a faulty challenge. If $\tilde{e}_i \notin D_H$, then $\mathcal{B}$ returns $\perp$. Otherwise $\mathcal{B}$ invokes the special c/s/p-HVZK simulator $\mathcal{M}$ of ID on input pk and $\tilde{e}_i$ and on freshly generated randomness every time, to obtain an accepting transcript $(\tilde{a}_i, \tilde{e}_i, \tilde{z}_i)$.
3. If $\mathrm{HT}[\tilde{a}_i, m_i, pk]$ is already set (via $\mathcal{A}$'s previous hash or signing queries) and $\mathrm{HT}[\tilde{a}_i, m_i, pk] \neq e_i$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ programs the random oracle so that
$$\mathrm{HT}[\tilde{a}_i, m_i, pk] \coloneqq e_i.$$
4. If random oracle programming is successful, $\mathcal{B}$ gives $\mathcal{A}$ a serialized transcript:
$$\tilde{\sigma}_i \coloneqq \mathsf{CSF}(\tilde{a}_i, \tilde{e}_i, \tilde{z}_i).$$

Now we evaluate the probability that $\mathcal{A}$ distinguishes the simulated signing oracle above from the real one. There are essentially two cases.

- $\mathcal{B}$ aborts at the third step in the above simulation with probability at most $(Q_h + Q_s)/2^{\alpha}$ per query. This is because the input $(\tilde{a}_i, m_i, pk)$ to H has α bits of min-entropy (since m_i is chosen by the adversary and has zero bits, pk is public and has zero bits, and $\tilde{a}_i$ has α bits due to α-bit min-entropy of ID), and HT has at most $Q_h + Q_s$ existing entries.
- After the fourth step, $\mathcal{A}$ distinguishes $\tilde{\sigma}_i$ from the one returned by the real signing oracle $\mathsf{OFaultHSign}$ with probability at most ϵ_{HVZK} per query, since we used the special c/s/p-HVZK simulator to derive $(\tilde{a}_i, \tilde{e}_i, \tilde{z}_i)$ and CSF is efficiently computable.

Recalling that the number of signing queries is bounded by Q_s, and by a union bound, $\mathcal{A}$ overall distinguishes its simulated view from that in UF-fCMA game with probability at most

$$\frac{(Q_h + Q_s)Q_s}{2^\alpha} + Q_s \cdot \epsilon_{HVZK}.$$

□

Remarks The above lemma relies on the fact that faulty $\tilde{e}_i$ is necessarily a "well-formed" challenge. For example, the challenge in some subset-revealing schemes has a specific structure (e.g., a list of pairs (c_i, p_i) where the c_i are distinct, as in Picnic2). Computing Resp with a malformed challenge may cause σ to leak private information. This is why we required Definition 3 to have the condition that Resp validates $\tilde{e}_i \in D_h$ and otherwise returns $\perp$. This way, the signing algorithm does not leak information when a malformed challenge is input to the response phase, and eventually outputs $\perp$ as a signature because CSF is sound with respect to invalid response (see Definition 6).

Note that the proof can be generalized to the multi-bit fault setting. More specifically, the random oracle programming becomes unnecessary for output replacement faults (i.e., f_6 applies `set_bit` to every bit of e) because in that case the fault adversary would no longer be able to observe any relation between faulty $\tilde{e}_i$ and the original, unfaulty e.

5.5 Faulting Response Input

Next we prove the security against tampering function f_7, which lets an attacker fault the input (sk, e, St) to the Resp function. We only guarantee security assuming that the signature scheme is based on a subset revealing identification protocol (see Definition 5), and Resp and CSF make sure to rule out invalid challenge and response, respectively. As we will see in the next section, Picnic2 satisfies these additional properties.

Lemma 9 (UF-KOA $\to$ $\{f_7\}$-UF-fCMA)**.** *Suppose* ID *is subset revealing. If* $\mathsf{FS} := \mathbf{FS}[\mathsf{ID}, \mathsf{CSF}]$ *is* UF-KOA *secure, then* FS *is* $\{f_7\}$-UF-fCMA *secure in the random oracle model. Concretely, given an* $\{f_7\}$*-adversary* $\mathcal{A}$ *against* FS *running in time* t*, and making at most* Q_s *queries to* OFaultSign*,* Q_h *queries to* H*, one can construct another adversary* $\mathcal{B}$ *against* FS *such that*

$$\mathbf{Adv}^{\mathsf{UF\text{-}fCMA}}_{\mathsf{FS}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{UF\text{-}KOA}}_{\mathsf{FS}}(\mathcal{B}) + \frac{(Q_s + Q_h)Q_s}{2^\alpha} + Q_s \cdot \epsilon_{HVZK},$$

where $\mathcal{B}$ *makes at most* Q_h *queries to its hash oracle, and has running time* t

Proof. The overall proof structure follows Lemma 5. The preparation of public key, simulation of hash queries and forgery phase are same as before. The proof diverges at signing oracle simulation.

Simulation of Faulty Signing Queries If f_7 is targeted at e, the situation is essentially the same as tampering with f_6 and thus the simulation strategy is identical to Lemma 8, except that CSF this time receives $(\tilde{a}_i, e_i, \tilde{z}_i)$ (i.e., simulated transcript with unfaulted challenge value) instead of $(\tilde{a}_i, \tilde{e}_i, \tilde{z}_i)$. Moreover, f_7 targeting sk has no effect on the signing operation since we assume the subset revealing property of ID. Below we consider the case when f_7 is targeted at $St_\iota \in St$. For each $i \in [Q_s]$, $\mathcal{B}$ answers a faulty signing request from $\mathcal{A}$ by simulating the signature on m_i as follows:

1. $\mathcal{B}$ first samples e_i from D_H uniformly at random.
2. $\mathcal{B}$ invokes the special c/s/p-HVZK simulator $\mathcal{M}$ of ID on input pk and e_i and on freshly generated randomness every time, to obtain an accepting transcript (a_i, e_i, z_i).
3. If $\mathrm{HT}[a_i, m_i, pk]$ is already set (via $\mathcal{A}$'s previous hash or signing queries) and $\mathrm{HT}[a_i, m_i, pk] \neq e_i$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ programs the random oracle so that
$$\mathrm{HT}[a_i, m_i, pk] := e_i.$$
4. If random oracle programming is successful, $\mathcal{B}$ derives from e_i an index set I as Resp of the real signing oracle would do.

- If $\iota \in I$, then $\mathcal{B}$ searches the corresponding entry St_ι in z_i and replaces it with $\widetilde{St}_\iota := f_7(St_\iota)$. Let $\tilde{z}_i$ be a faulty response modified as above.
- If $\iota \in I$, then $\mathcal{B}$ does not modify a response, i.e. $\tilde{z}_i = z_i$.

Then $\mathcal{B}$ gives $\mathcal{A}$ a serialized transcript

$$\tilde{\sigma}_i := \mathsf{CSF}(a_i, e_i, \tilde{z}_i).$$

Now we evaluate the probability that $\mathcal{A}$ distinguishes the simulated signing oracle above from the real one. There are essentially two cases.

- $\mathcal{B}$ aborts at the third step in the above simulation with probability at most $(Q_h + Q_s)/2^\alpha$ per query. This is because the input (a_i, m_i, pk) to H has α bits of min-entropy (since m_i is chosen by the adversary and has zero bits, pk is public and has zero bits, and a_i has α bits due to α-bit min-entropy of ID), and HT has at most $Q_h + Q_s$ existing entries.
- After the fourth step, $\mathcal{A}$ distinguishes $\tilde{\sigma}_i$ from the one returned by the real signing oracle $\mathsf{OFaultHSign}$ with probability at most ϵ_{HVZK} per each query, since we used the special c/s/p-HVZK simulator to derive (a_i, e_i, z_i), CSF is efficiently computable, and $\mathcal{B}$ modified z_i in the same way that the real faulty signing operation would.

Recalling that the number of signing queries is bounded by Q_s, and by a union bound, $\mathcal{A}$ overall distinguishes its simulated view from that in $\mathsf{UF\text{-}fCMA}$ game with probability at most

$$\frac{(Q_h + Q_s)Q_s}{2^\alpha} + Q_s \cdot \epsilon_{HVZK}.$$

□

Remark Intuitively, subset revealing ID schemes are secure against faults on St because the adversary only obtains what they could have computed by changing non-faulty signatures by themselves. On the other hand, the Schnorr signature scheme is not secure against tampering with f_7 and we describe concrete fault attacks in Lemma 14.

As we remarked after Definition 5, one can consider a highly inefficient version of Schnorr signature that enumerates all possible responses in St and opens one of them. In doing so, the Resp function avoids any algebraic operations involving sk and ρ, and we can mitigate the risk of faulty response input attacks described above. This countermeasure is of course impractical since the challenge space is too large, but it illustrates a concrete case where subset revealing ID schemes are more robust against fault attacks, in our model.

5.6 Faulting Commitment Output

Recall that a fault of type f_4 allows the attacker to fault the output of $\mathsf{Com}(sk; \rho)$, the commitment function in the first step of the ID scheme. Here we prove that randomized Fiat–Shamir signature schemes are secure against this type of fault attack, under the same conditions as ones in Lemma 9.

Lemma 10 ($\mathsf{UF\text{-}KOA} \to \{f_4\}\text{-}\mathsf{UF\text{-}fCMA}$)**.** *Suppose* ID *is subset revealing. If* $\mathsf{FS} := \mathbf{FS}[\mathsf{ID}, \mathsf{CSF}]$ *is* $\mathsf{UF\text{-}KOA}$ *secure, then* FS *is* $\{f_4\}$-$\mathsf{UF\text{-}fCMA}$ *secure in the random oracle model. Concretely, given an* $\{f_4\}$*-adversary* $\mathcal{A}$ *against* FS *running in time* t*, and making at most* Q_s *queries to* $\mathsf{OFaultSign}$ *and* Q_h *queries to* H*, one can construct another adversary* $\mathcal{B}$ *against* FS *such that*

$$\mathbf{Adv}^{\mathsf{UF\text{-}fCMA}}_{\mathsf{FS}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{UF\text{-}KOA}}_{\mathsf{FS}}(\mathcal{B}) + \frac{(Q_s + Q_h)Q_s}{2^{\alpha-1}} + Q_s \cdot \epsilon_{HVZK},$$

where $\mathcal{B}$ *makes at most* Q_h *queries to its hash oracle, and has running time* t

Proof. If f_4 is targeted at a_i per query $i \in [Q_s]$, the proof is essentially a special case of Lemma 7, where f_5 is applied to a_i, and CSF receives $(\hat{a}_i, e_i, z_i)$ instead of (a_i, e_i, z_i); if f_4 is targeted at St, the proof directly follows Lemma 9. Putting both cases together, we obtain the statement. □

5.7 Negative Results

Here we show that fault attacks of type f_0, f_2 and f_3 are not mitigated by the hedged construction for an ID scheme with the same properties as in Theorem 1.

Lemma 11. *There exist canonical* ID *schemes such that* $\mathbf{R2H}[\mathbf{FS}[\mathsf{ID}, \mathsf{CSF}], \mathsf{HE}]$ *is* UF-CMNA*-secure, but not* $\{f_0\}$-UF-fCMNA *secure.*

Proof. We consider the Schnorr scheme that returns (e, z) as a signature (see Algorithm 1), for which $\mathbf{FS}[\mathsf{ID}, \mathsf{CSF}]$ is known to be UF-CMA secure and therefore $\mathbf{R2H}[\mathbf{FS}[\mathsf{ID}, \mathsf{CSF}], \mathsf{HE}]$ is UF-CMNA secure due to Lemma 3. Our $\{f_0\}$-adversary's strategy is as follows. The adversary first calls OFaultHSign with some (m, n) without fault (i.e., $\phi = Id$) to obtain a legitimate signature (e, z). Next, the adversary calls OFaultHSign with $\phi = \texttt{flip\_bit}_i$, $j = 0$ and (m', n), where m' is identical to m except at the i-th bit. This way, it can fault m' back to m before the invocation of HE and hence the signature is derived from the same ρ as in the previous query, while the challenge and response are different since $e' = \mathsf{H}(a, m', pk)$ and $z = \rho + e' \cdot sk$ mod q. Hence we can recover sk with the SSND attack in Section 3 and break the scheme. □

Lemma 12. *There exist canonical* ID *schemes such that* $\mathbf{R2H}[\mathbf{FS}[\mathsf{ID}, \mathsf{CSF}], \mathsf{HE}]$ *is* UF-CMNA*-secure, but not* $\{f_3\}$-UF-fCMNA *secure.*

Proof. We describe a simple attack that works for the Picnic ID scheme. Recall that f_3 is applied to input of $\mathsf{Com}(sk; \rho)$. When querying OFaultHSign, the attacker uses `set_bit` to set the i-th bit of sk, denoted sk_i to 0, then observes whether the signature output is valid. If so, then the true value of sk_i is 0, and if not, then sk_i is one. By repeating this for each of the secret key bits, the entire key may be recovered. Some ID schemes may include internal checks and abort if some computations are detected to be incorrect relative to the public key, in this case the attacker checks whether OFaultHSign aborts. □

Note that Lemma 12 only applies to ID schemes where sk is used by the Com function. For the Schnorr scheme and other so-called *input delayed protocols* [CPS+16], sk is only used by the Resp function. In this way subset-revealing ID schemes and input delayed ID schemes have the opposite behavior, since subset-revealing schemes do not use sk in the Resp function, but they must use it in the Com function.

The sensitivity of ephemeral randomness ρ in Schnorr-like schemes is well known, and once the attacker obtains sufficiently many biased signatures, the secret key can be recovered by solving the so-called *hidden number problem (HNP)* [BV96]. Previous works have shown that even a single-bit bias helps to recover sk by making use Bleichenbacher's solution to HNP [Ble00, AFG+14]. However, the currently known algorithms for the HNP do not give an asymptotically efficient attack, they only reduce the concrete security of the scheme sufficiently to allow a practical attack on some parameter sets. For instance, with the current state-of-the-art algorithm based on Bleichenbacher's attack found in the literature [TTA18, Theorem 2], one can practically break 1-bit biased signatures instantiated over 192-bit prime order groups, using $2^{29.6}$ signatures as input, and with $2^{29.6}$ space and $2^{59.2}$ time, which is tractable for computationally well-equipped adversaries as of today.

To attack Schnorr-like schemes with f_3, the adversary would instead target the randomness ρ to cause a single-bit bias in it, and this situation is essentially same as faulting with f_2. Such an attack would be also powerful enough to recover the entire signing key, which we describe below.

Lemma 13. *Relative to an oracle for the hidden number problem, there exist a non-subset revealing canonical* ID *scheme such that* $\mathbf{R2H}[\mathbf{FS}[\mathsf{ID}, \mathsf{CSF}], \mathsf{HE}]$ *is* UF-CMNA*-secure, but neither* $\{f_2\}$-UF-fCMNA *nor* $\{f_3\}$-UF-fCMNA *secure.*

Proof. We describe an attack that works for the Schnorr signature scheme. Recall that both f_2 and f_3 can tamper with ρ in Schnorr, as its St contains the randomness ρ. If f_2 or f_3 is `set_bit` and always targets at the most significant bit of ρ to fix its value, the attacker can introduce 1-bit bias in ρ.

Relative to an oracle for the HNP, the Schnorr scheme with unbiased ρ remains secure, however, the scheme with biased ρ is broken. We must assume here that the HNP oracle does not help an attacker break the Schnorr scheme with unbiased nonces (otherwise the Theorem is trivial). It is easy to see that the HNP with uniformly random nonces does not give a unique solution – the adversary is given a system of Q_s equations with $Q_s + 1$ unknowns, so a direct application of the HNP oracle does not help. However, there may be other ways to use the HNP oracle, so we must make the assumption. □

We remark that, despite the existence of the above attack, it is much less devastating than the large randomness bias (LRB) attack on deterministic schemes described in Section 3. With the LRB attack, the adversary only needs two signatures to recover the full key, while the above attack relying on a HNP solver requires a significant amount of faulty biased signatures as input in practice. This indicates that hedged constructions do, to some extent, mitigate the effect of faults on ρ. For fault types f_7 and f_4, we have shown that $\mathbf{R2H}[\mathbf{FS}[\mathsf{ID}, \mathsf{CSF}], \mathsf{HE}]$ is secure assuming ID is subset-revealing. The following two lemmas give counterexamples when ID is not subset revealing, showing that canonical ID schemes are not generically secure for faults f_7 and f_4.

Lemma 14. *There exist non-subset-revealing canonical* ID *schemes such that* $\mathbf{R2H}[\mathbf{FS}[\mathsf{ID}, \mathsf{CSF}], \mathsf{HE}]$ *is* UF-CMNA*-secure, but not* $\{f_7\}$-UF-fCMNA *secure.*

Proof. We describe two attacks that work for the Schnorr signature scheme.

- If f_7 is `set_bit` and targeted at sk, the adversary can use the strategy of Lemma 12 to learn each bit of sk by checking whether the faulty signatures pass verification.
- If f_7 is `flip_bit` and targeted at the most significant bit of $St = \rho$, the adversary obtains (e, z') such that $z' = e \cdot sk + f_7(\rho)$, and he can recover the "faulty" commitment $a' = [f_7(\rho)]G$. Recall that the non-faulty commitment $a = [\rho]G$ satisfies $\mathsf{H}(a, m, pk) = e$, so the adversary can learn 1-bit of ρ by checking whether $\mathsf{H}(a' + [2^{\ell_\rho - 1}]G, m, pk) = e$ or $\mathsf{H}(a' - [2^{\ell_\rho - 1}]G, m, pk) = e$ holds, where ℓ_ρ is the bit length of ρ. Since we now have the most significant bit of ρ, we use the same argument as in Lemma 13 to show the scheme is vulnerable to fault attacks.

□

Lemma 15. *There exist non-subset-revealing canonical* ID *schemes such that* $\mathbf{R2H}[\mathbf{FS}[\mathsf{ID}, \mathsf{CSF}], \mathsf{HE}]$ *is* UF-CMNA*-secure, but not* $\{f_4\}$-UF-fCMNA *secure.*

Proof. Recall that f_4 is applied to (a, St), the output of Com. In the Schnorr signature scheme, St contains the per-signature ephemeral value ρ, which is the output of the hedged extractor. Therefore, the same attack as described in Lemma 14 for f_7-faults can be mounted with an f_4-fault. □

6 Analysis of XEdDSA

In this section we apply the results of Section 5 to the XEdDSA signature scheme. The scheme is presented in Appendix B. The associated ID scheme is the Schnorr ID scheme (denoted ID-Schnorr). Then we define $\mathsf{Schnorr} := \mathbf{FS}[\mathsf{ID\text{-}Schnorr}, \mathsf{CSF}]$ and $\mathsf{XEdDSA} := \mathbf{R2H}[\mathsf{Schnorr}, \mathsf{HE}]$, where CSF returns (a, z). We start by establishing some well-known properties of ID-Schnorr. As noted in Section 2 ID-Schnorr is not subset-revealing.

Lemma 16. ID-Schnorr *is perfect HVZK (therefore* $\epsilon_{HVZK} = 0$*) and has* 2λ *bits of min-entropy.*

Proof. Proof of HVZK is given in [KMP16, Lemma 5.2]. The commitment a is a random group element, in a group of order at least 2λ bits (to provide λ-bit security against key recovery attacks using generic algorithms for the discrete logarithm problem [Sho97, Nec94]). □

UF-KOA Security Let $\mathbf{Adv}^{\text{UF-KOA}}_{\mathsf{Schnorr}}(\mathcal{A})$ be the (concrete) UF-KOA security of Schnorr against an adversary $\mathcal{A}$ running in time t. As non-hedged XEdDSA is identical to Schnorr in the UF-KOA setting, the concrete analysis for Schnorr of [KMP16, Lemmas 3.5-3.7] and [PS00, Lemma 8] are applicable. We do not repeat those results here (as they are lengthy and don't add much to the present paper), but instead state our results in terms of $\mathbf{Adv}^{\text{UF-KOA}}_{\mathsf{Schnorr}}(\mathcal{A})$. We can now apply the results of Section 5.

Corollary 1. *XEdDSA is* $\{f_1, f_5, f_6, f_8, f_9, f_{10}\}$-UF-fCMNA *secure.*

Proof. We've shown above that ID-Schnorr is perfect HVZK (so $\epsilon_{HVZK} = 0$) and has $\alpha = 2\lambda$ bits of min-entropy. Then we can apply Theorem 1, to obtain

$$\mathbf{Adv}^{\text{UF-fCMNA}}_{\mathsf{XEdDSA}}(\mathcal{A}) \leq 2\left(\mathbf{Adv}^{\text{UF-KOA}}_{\mathsf{Schnorr}}(\mathcal{B}) + \frac{(Q_s + Q_h)Q_s}{2^{2\lambda-1}}\right)$$

□

Remaining fault types. We now consider the faults of type f_0, f_2, f_3, f_4, and f_7 where we can't prove security. For each of these, we have given an attack elsewhere in the paper, for Schnorr signatures, but that also applies to XEdDSA. For type f_0 see Lemma 11, for types f_2 and f_3 see Lemma 13, for type f_4 see Lemma 15 and for type f_7 see Lemma 14.

7 Analysis of Picnic2

In this section we analyze the Picnic2 variant of the Picnic signature scheme using our formal model for fault attacks. Since Picnic is constructed from a subset-revealing ID scheme, more of the results from Section 5 apply, reducing our effort in this section. We use ID-Picnic2 to denote the ID scheme, and Picnic2 := **FS**[ID-Picnic2, CSF] and HS-Picnic2 := **R2H**[Picnic2, HE] to denote the randomized and hedged signature schemes. The signature scheme is described in some detail in Appendix C, and complete details are in the Picnic design document [Pic19b] and specification [Pic19a].

7.1 Preliminaries

In this section we state (and prove) some general properties of Picnic2 that will then be used when proving resistance to fault attacks.

ID-Picnic2 is a subset-revealing ID scheme. Note that its St consists of

$$\{h_j, h'_j, \mathsf{seed}^*_j, \{\hat{z}_{j,\alpha}\}, \mathsf{state}_{j,i}, \mathsf{com}_{j,i}, \mathsf{msgs}_{j,i}\}_{j\in[M],i\in[n]}$$

and Resp simply reveals a subset of it depending on a challenge $\mathcal{C}$ and $\mathcal{P}$.

The Picnic2 specification is an instance of R2H. The specification recommends a hedging construction that is an instance of the **R2H** construction from Section 4. In this case, the salt and random seeds are derived deterministically from $sk\|m\|pk\|n$ where n is a 2λ-bit random value (acting as the nonce in the notation of Section 4). The function HE is instantiated with the SHA-3 based derivation function SHAKE. The security analysis in [Pic19b] applies to the randomized version of the signature scheme, so we must use Lemma 3 to establish UF-CMNA security of the hedged variant.

Lemma 17. *For security parameter λ, ID-Picnic2 has $\alpha \geq 2\lambda + 256$ bits of min-entropy.*

Proof. The first message a is a list of commitments (of length $2M$), output by a hash function modeled as a random oracle (having 2λ-bit outputs). In a fully randomized version of ID-Picnic2, the inputs to the commitments contain randomly chosen seeds (each of length λ bits).

When using the optimizations and salt described in the specification, the seeds are generated with a PRG, whose input includes the signer's secret key (λ bits), an initial λ-bit seed, chosen per-signature, and a random 256-bit salt. Since a secure PRG must be a (nearly) injective function this preserves the entropy in the outputs. Similarly, when computing commitments with a random oracle, the entropy in the 2λ-bit commitments preserves the entropy of the λ-bit seeds. □

The next corollary shows that Picnic2 is secure against key-only attacks, and it follows from the unforgeability security proof of Picnic2 from [Pic19b].

Corollary 2. *The signature scheme Picnic2 is UF-KOA secure, when the hash functions H_0, H_1, H_2 and G are modeled as random oracles with 2λ-bit outputs, and key generation function Gen is (t, ϵ_{OW})-one-way.*

Proof. Theorem 6.2 of the Picnic design document [Pic19b] gives a concrete analysis of the UF-CMA security of the randomized scheme. This immediately implies UF-KOA security, with the same bound. Concretely, for all PPT adversaries $\mathcal{A}$, we have

$$\mathbf{Adv}^{\mathsf{UF\text{-}KOA}}_{\mathsf{Picnic2}}(\mathcal{A}) \leq O(q_s \cdot \tau \cdot \epsilon_{PRG}) + O\left(\frac{(q_0 + q_1 + q_2 + Mnq_s)^2}{2^\lambda}\right) + 2\epsilon_{OW} + 2q_G \cdot 2^{-\lambda}, \tag{1}$$

and $\mathcal{A}$'s runtime is $\approx t$. Here q_s is the number of signing queries, and q_0, q_1, q_2 and q_G are the number queries to H_0, H_1, H_2 and G respectively. Note here that we've replaced the quantity $\epsilon(M, n, \tau)$ from the Picnic design document with $2^{-\lambda}$, since all parameter sets choose (M, n, τ) so that $\epsilon(M, n, \tau) \leq 2^{-\lambda}$.

Security of the PRG is not required in a key-only attack, as the PRG is only used during signing, and some of the loss of tightness comes from simulating the signing oracle, in Experiments 3–6 (of [Pic19b, Theorem 6.2]). When we remove the adversary's signing oracle from the proof, and replace O-notation with concrete analysis, the bound simplifies to

$$\mathbf{Adv}^{\mathsf{UF\text{-}KOA}}_{\mathsf{Picnic2}}(\mathcal{A}) \leq \frac{3{Q_h}^2}{2^{2\lambda}} + 2\epsilon_{OW} + \frac{Q_h}{2^{\lambda}}$$

where $Q_h = q_0 + q_1 + q_2 + q_G$.

□

Honest-Verifier Zero-Knowledge Here we prove that ID-Picnic2 is special computational honest-verifier zero-knowledge (c-HVZK). The Picnic design document proves security of HS-Picnic2 directly, and does not first prove HVZK of ID-Picnic2. In [KKW18], it is shown that a variant of ID-Picnic2 with randomized commitments is HVZK, but in the specified version no additional randomness is included (in order to make the opening information smaller). Because of this optimization, the proof is somewhat different.

Lemma 18. ID-Picnic2 *is a special c-HVZK proof, under the following assumptions: the hash functions H_0, H_1 and H_2 are modeled as random oracles, and the PRG is (t, ϵ_{PRG})-secure. Simulated transcripts are computationally indistinguishable from real transcripts, and all polynomial-time distinguishing algorithms succeed with probability at most*

$$\epsilon_{HVZK} \leq (n+2)\tau \cdot \epsilon_{PRG} + \frac{q_0\tau + q_2 M}{2^{\lambda}} .$$

where q_0 and q_2 are the number of queries to H_0 and H_2, λ is the security parameter, and (M, n, τ) are parameters of the scheme.

Proof. Proof that the protocol is HVZK is similar to the one given in [KKW18, Theorem 2.2], but accounts for the change to remove additional randomness from commitments.

Let Π denote the MPC protocol, and Sim_{Π} denote a simulator for Π. Recall that Π has semi-honest security (for corruption of any $n-1$ parties).

We now describe a simulator that creates transcripts for the interactive version of the protocol in Figures 11 and 12 that are computationally indistinguishable from real transcripts, using a hybrid argument. Game 0 is the real protocol, and each subsequent game will change the protocol until it no longer uses the witness. We write G_i to denote the probability that an adversary is a successful distinguisher in Game i.

Game 1 We change the prover so that it first chooses the challenge $(\mathcal{C}, \mathcal{P})$ at random instead of the verifier doing so. The challenge has the exact distribution as the challenge chosen by the verifier (who is honest by assumption), so games 0 and 1 produce identically distributed transcripts. In the case of *special* HVZK, the challenge is provided as input, and we use it directly (the simulation below works for all challenge values).

Game 2 We change the prover to choose random $\{\mathsf{seed}_{j,i}\}_{i=1}^{n}$ for all $j \in \mathcal{C}$, rather than using the PRG. We have $\mathsf{G}_2 - \mathsf{G}_1 = n\tau \cdot \epsilon_{PRG}$ (recall that $|\mathcal{C}| = \tau$).

Game 3 In each $j \in \mathcal{C}$, we change the prover to choose random bits to be used in the MPC protocol by party p_j (the unopened party), instead of deriving them from seed_{j,p_j} with the PRG. Recall that there are a total of τ unopened parties. Since the seeds are random values committed to as $\mathsf{com}_{j,p_j} = H_0(\mathsf{seed}_{j,p_j})$, this goes undetected except with probability $\frac{q_0\tau}{2^{\lambda}}$, where q_0 is the number of queries to H_0. Therefore $\mathsf{G}_2 - \mathsf{G}_3 \leq \tau \cdot \epsilon_{PRG} + \frac{q_0\tau}{2^{\lambda}}$.

Game 4 We change the prover, for each $j \in [M] \setminus \mathcal{C}$, to choose uniform h'_j (i.e., without querying H_2). The input to H_2 when computing h'_j is $I := (\{\hat{z}_{j,\alpha}\}, \mathsf{msgs}_{j,1}, \ldots, \mathsf{msgs}_{j,n})$ Since $\{\hat{z}_{j,\alpha}\}$ is computed from the first λ bits of each party's random tape, and p_j's tape is chosen uniformly at random (as of Game 3), and not used elsewhere, I has at least λ bits of min-entropy. Therefore $\mathsf{G}_4 - \mathsf{G}_3$ is the probability that I is queried to H_2 at another point in the game, and $\mathsf{G}_4 - \mathsf{G}_3 \leq q_2 \cdot M \cdot 2^{-\lambda}$.

Game 5 We change the prover, for each $j \in \mathcal{C}$, to compute com_{j,p_j} as a commitment to a random value. Since seed_{j,p_j} is no longer used (as of Game 4), computing com_{j,p_j} in this way is consistent with the rest of the transcript, and so $\mathsf{G}_4 = \mathsf{G}_5$.

Game 6 We change the prover, for $j \in \mathcal{C}$, to use Sim_Π to generate the views of the parties (excluding p_j), in an execution of Π when evaluating C with output 1. This results in values $\{\mathsf{state}_{j,i}\}_{i \neq p_j}$, masked input-wire values $\{\hat{z}_{j,\alpha}\}$, and msgs_{p_j}. From the respective views, $\{\mathsf{msgs}_{j,i}\}_{i \neq p_j}$ can be computed, and h_j and h'_j can be computed as well. Since $j \in \mathcal{C}$, only $n-1$ parties are opened, and the simulated transcripts for Π are distributed exactly as real transcripts, under the semi-honest security of Π, therefore Games 5 and 6 produce identically distributed outputs. The security of Π (proven in Section 6.1 of [Pic19b]), holds assuming the PRG used to generate the random tapes is secure, therefore $\mathsf{G}_6 - \mathsf{G}_5 = \tau \cdot \epsilon_{PRG}$.

In Game 6 the witness is no longer used, and the simulated transcripts are computationally indistinguishable from real transcripts, concluding proof of the HVZK property. □

7.2 Applying the Results of Section 5

In Section 5 we proved that a class of hedged FS signature schemes are UF-fCMNA secure for faults of type f_1, and $f_4, \ldots, f_{10}$. Here we apply those results to the HS-Picnic2 signature scheme. Recall that HS-Picnic2 denotes $\mathbf{R2H}[\mathbf{FS}[\mathsf{ID\text{-}Picnic2}, \mathsf{CSF}], \mathsf{HE}]$.

Corollary 3. HS-Picnic2 *is* $\{f_1, f_4, \ldots, f_{10}\}$-UF-fCMNA *secure.*

Proof. Recall that by Corollary 2, Picnic2 is UF-KOA secure with

$$\mathbf{Adv}^{\mathsf{UF\text{-}KOA}}_{\mathsf{Picnic2}}(\mathcal{A}) \leq \frac{3{Q_h}^2}{2^{2\lambda}} + 2\epsilon_{OW} + \frac{Q_h}{2^\lambda}$$

and the min-entropy α is $2\lambda + 256$ as shown in Lemma 17.

We can apply Theorem 1, to obtain

$$\begin{aligned}\mathbf{Adv}^{\mathsf{UF\text{-}fCMNA}}_{\mathsf{HS\text{-}Picnic2}}(\mathcal{A}) &\leq 2 \cdot \mathbf{Adv}^{\mathsf{UF\text{-}KOA}}_{\mathsf{Picnic2}}(\mathcal{B}) + \frac{(Q_s + Q_h)Q_s}{2^{\alpha-2}} + 2Q_s \cdot \epsilon_{HVZK} \\ &= \frac{6{Q_h}^2}{2^{2\lambda}} + 4\epsilon_{OW} + \frac{2Q_h}{2^\lambda} + \frac{(Q_s + Q_h)Q_s}{2^{2\lambda+254}} + 2Q_s \cdot \epsilon_{HVZK},\end{aligned}$$

where ϵ_{HVZK}, given in Lemma 18, is

$$\epsilon_{HVZK} \leq (n+2)\tau \cdot \epsilon_{PRG} + \frac{q_0\tau + q_2 M}{2^\lambda}.$$

□

Discussion Our model places the restriction on OFaultHSign that the challenge supplied by the adversary must be in the challenge space defined by the signature scheme. This is natural from a theoretical modeling perspective but an f_6-fault attack may make the challenge for HS-Picnic2, $(\mathcal{C}, \mathcal{P})$, be any value of the correct length. The integers in $\mathcal{C}$ and $\mathcal{P}$ may be out of range, in this case we can reasonably expect most implementations to fail, since the values are used to index into data structures maintained by the signer. The other case is that $\mathcal{C}$ may not be a set, for example if a byte of the output is randomly corrupted, with some probability two values in $\mathcal{C}$ will match. If so, there is an efficient attack. Suppose c_1 and c_2 from $\mathcal{C}$ are the same, and the corresponding values in $\mathcal{P}$ are different. Then the signer will reveal the state of all parties in the MPC simulation, which leaks the signing key.

Our model and analysis surfaces a required implementation countermeasure for protecting against fault attacks, which corresponds to the sanity check of challenge format we mentioned in Definition 3. Implementations must ensure that each MPC instance is opened only once. A natural way to achieve this is to iterate over the MPC instances, rather than the challenge values.

7.3 Picnic-Specific Results

We've shown that security against fault types f_2 and f_3 do not hold in general. In this section we prove that HS-Picnic2 is $\{f_2\}$-UF-fCMNA secure, and discuss possible mitigations to fault attacks of type f_3.

Fault type f_2 Recall that f_2 is a fault on ρ, the output of the hedged extractor. Intuitively, HS-Picnic2 is $\{f_2\}$-UF-fCMNA secure since ρ is not used directly, ρ is the list of seed^*_j values, which are used as input to a PRG when deriving the $\mathsf{seed}_{i,j}$ values. Applying a 1-bit fault to a seed^*_j value reduces the min-entropy by at most one bit, so only a small change to the security proof and analysis is required. Concretely we have:

Lemma 19. HS-Picnic2 *is* $\{f_2\}$-UF-fCMNA *secure.* $\mathbf{Adv}^{\text{UF-fCMNA}}_{\text{HS-Picnic2}}(\mathcal{A})$ *is the same as given in Corollary 3, except that* α *is reduced by 1.*

Fault type f_3 Recall that f_3 faults are applied to $\mathsf{Com}(f_3(sk;\rho))$. An f_3 fault that changes ρ is equivalent to an f_2 fault, so without loss of generality we consider only f_3-faults to sk. The generic attack described above also applies to HS-Picnic2: the attacker sets the i-th bit of sk, denoted sk_i to 0, then observes whether the signature output is valid. If so, then the true value of sk_i is 0, and if not, then sk_i is one. By repeating this for each of the secret key bits, the entire key may be recovered.

A generic mitigation to this attack (in software implementations) is for the signer to always ensure the signature being output is valid. For example, verifying the signature, then if it fails, re-loading the private key and recomputing the signature. To make this constant time the implementation must always compute two signatures and one verification. In the one-fault model, two signatures is always sufficient.

With Picnic this mitigation can be nearly free of cost, because a fault to sk can be detected without verifying a whole signature. After completing the online MPC simulation (Step 1.3) for the first MPC instance ($j = 1$), the signer can check whether the simulation is correct by comparing the output to part of pk (by comparing λ-bit strings). If simulation has failed, the signer can re-load and re-share the private key, then repeat the simulation, this time with the correct sk. Repeating the MPC simulation for a single MPC instance is less than $1/M$ of the total signing cost, and the range of M values in [Pic19a] is 343–803.

8 Concluding Remarks

This paper explored the effects of bit-tampering fault attacks on various internal values in hedged Fiat–Shamir signing operations, within the provable security methodology. Our security model is general enough to capture a large class of signatures, but also fine-grained enough to cover existing attacks surveyed in Section 3. In this section we describe some fault attacks that are not covered by our model, to illustrate the limitations of our analysis. Each of these issues makes an interesting direction for future work.

Use of the random oracle model Since the adversary does not have a description of the hash function, he cannot specify a fault in the hash function implementation, only the inputs and outputs can be faulted. For an example of how this limitation may be exploited, consider the PRG in Picnic2, modeled as a random oracle, used to expand a short seed into a random tape of length t bytes. The call is something like $tape = \text{PRG}(seed, t)$. Since t is a constant, the attacker can fault the high bit to zero, replacing t with $t' = t - 2^{\lfloor \log_2 t \rfloor}$. In the faulted execution, $tape' = \text{PRG}(seed, t')$ and the PRG outputs $2^{\lfloor \log_2 t \rfloor}$ fewer random bytes than expected, and the signature is computed with uninitialized memory (zero or low-entropy depending on the system and implementation) and the signature leaks the signing key with high probability (when the tape belongs to the unopened party).

Removing this limitation appears to be difficult. One could add the length parameter to the random oracle and model this specific attack, but other hash function internal computations are still not modeled. If

we allowed faults on the hash function internals, after any fault it becomes unclear what high-level properties (such as collision resistance) the faulted hash function still provides, preventing many security reductions.

Faulting global parameters Most cryptographic algorithms depend on global parameters, for example, the parameters of an elliptic curve in EdDSA [ABF+18], or the round constants of the LowMC block cipher in Picnic2. Injecting a fault to these parameters causes the signing algorithm to compute a different function of sk and m, and the resulting output may leak information about sk. Our model does not include these parameters, so implementers must evaluate whether faults to these parameters lead to attacks for each specific algorithm.

Multiple bit and word faults Picnic computes commitments to random seeds, for example, $C = H(\mathsf{seed})$ where seed is a 128-bit value, with two 64-bit halves, $\mathsf{seed} = \mathsf{seed}_{\text{low}} \| \mathsf{seed}_{\text{high}}$. There may be other public inputs to H, which we can ignore for the moment. If the fault attacker can zero a 64-bit machine word, then get $C' = H(0 || \mathsf{seed}_{\text{high}})$, he can find $\mathsf{seed}_{\text{high}}$ by brute force with 2^{64} hash computations, then find $\mathsf{seed}_{\text{low}}$ with 2^{64} work as well. This generalizes to fewer than 64 bits – if n-bit faults are allowed, then seed recovery is a factor 2^n cheaper. This also generalizes to other functions with private input and public output.

Faults within the Com and Resp functions Our model only allows faults to the inputs and outputs of functions that describe the $\mathbf{R2H}[\mathbf{FS}[\mathsf{ID}, \mathsf{CSF}], \mathsf{HE}]$ construction. However, in typical ID schemes significant computation happens inside the Com and Resp functions, and a real-world attacker may be able to fault these computations. For Picnic2, one example is injecting faults in the random tape, this is within the Com function, so it's outside our model. By faulting the random tape of an unopened party, it is possible to learn something about that party's share. Because the sharing is bitwise, given one bit of the unopened party's share means we get one bit of the secret key.

Multiple faults per signature query In our model, the attacker must choose a single fault type f_i per query to OFaultHSign. In practice, it may be possible for an attacker to introduce two faults in a signature computation, of different types. It is likely that allowing combinations of fault types leads to attacks, or that the security analysis we give here does not hold.

Instruction skip attacks With this type of fault attack, the attacker causes specific instructions in the signing code to be skipped. For example, the attacker can cause RNG calls to be skipped, causing signatures to use degenerate random values. Or the attacker may skip the assignment of a variable, causing execution to proceed with a zero value. This could be modeled with a generalization of our model where faults may set variables to arbitrary values, or zero them.

Persisting faults Referring to the definition of OFaultHSign in Figure 6, suppose fault f_1 is applied to m. In our model, the fault specified by f_1 to m is only effective in the HE function, then later uses of m, say in H, are unfaulted. An alternative model could have the fault introduced by f_1 persist throughout OFaultHSign. Which model is more realistic depends on the implementation, and how faults are injected. The persistent fault option may be more realistic if the fault is injected to memory, and all functions read the value from memory. On the other hand, if faults are injected to values once they are in registers, or if the implementation copies values to the different functions, then faults would not be persistent. We briefly mention an example to show that this aspect of the model can lead to different conclusions. The attacker mounts the SSND attack, by first calling $\mathsf{OFaultHSign}(m, n)$ with no faults, then calling $\mathsf{OFaultHSign}(m', n, 1, \texttt{flip\_bit})$ where $m = f_1(m')$, i.e., the second call makes m' equal to m when input to HE. The inputs to HE are the same, and so ρ is the same. If the faults are not propagated, e will differ in the two signatures, allowing the SSND attack to succeed (as described in Lemma 11), while if faults are propagated, both signatures are the same and the attack fails.

Fiat–Shamir with Aborts In this paper, we tailored our definitions to schemes based on the standard Fiat–Shamir transform. However, a few recent fault attack papers [BP18, RJH+19] targeted deterministic

lattice-based signatures following the Fiat–Shamir *with aborts* paradigm [Lyu09, Lyu12], which has an additional rejection sampling phase after the response, forcing the response output z to follow some probability distribution independent of the secret key (typically the discrete Gaussian distribution centered at 0, or the uniform distribution in a certain interval). We leave for future work the extension of our formal analysis to signatures of Fiat–Shamir with aborts family, so as to clarify to what extent the hedged version of such schemes could mitigate the risk of bit-tampering fault attacks.

Acknowledgments. This research was supported by: the Concordium Blockchain Research Center, Aarhus University, Denmark; the Carlsberg Foundation under the Semper Ardens Research Project CF18-112 (BCM); the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 803096 (SPEC); the Danish Independent Research Council under Grant-ID DFF-6108-00169 (FoCC). We thank anonymous reviewers for their valuable comments and suggestions.

References

[AABN02] Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 418–433. Springer, Heidelberg, April / May 2002.

[AASA+19] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, et al. *Status report on the first round of the NIST post-quantum cryptography standardization process*. 2019.

[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, *CT-RSA 2018*, volume 10808 of *LNCS*, pages 339–353. Springer, Heidelberg, April 2018.

[ACM+17] Per Austrin, Kai-Min Chung, Mohammad Mahmoody, Rafael Pass, and Karn Seth. On the impossibility of cryptography with tamperable randomness. *Algorithmica*, 79(4):1052–1101, 2017.

[AFG+14] Diego F. Aranha, Pierre-Alain Fouque, Benoît Gérard, Jean-Gabriel Kammerer, Mehdi Tibouchi, and Jean-Christophe Zapalowicz. GLV/GLS decomposition, power analysis, and attacks on ECDSA signatures with single-bit nonce bias. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 262–281. Springer, Heidelberg, December 2014.

[BAA+19] Nina Bindel, Sedat Akleylek, Erdem Alkim, Paulo S. L. M. Barreto, Johannes Buchmann, Edward Eaton, Gus Gutoski, Juliane Kramer, Patrick Longa, Harun Polat, Jefferson E. Ricardini, and Gustavo Zanon. qTESLA. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[Bae14] Maarten Baert. Ed25519 leaks private key if public key is incorrect #170. https://github.com/jedisct1/libsodium/issues/170, 2014.

[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, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 232–249. Springer, Heidelberg, December 2009.

[BBSS18] Matilda Backendal, Mihir Bellare, Jessica Sorrell, and Jiahao Sun. The Fiat-Shamir Zoo: Relating the Security of Different Signature Variants. In Nils Gruschka, editor, *NordSec 2018*, Lecture Notes in Computer Science, pages 154–170. Springer, 2018.

[BCM11] Mihir Bellare, David Cash, and Rachel Miller. Cryptography secure against related-key attacks and tampering. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 486–503. Springer, Heidelberg, December 2011.

[BCN+06] Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall, and Claire Whelan. The sorcerer's apprentice guide to fault attacks. *Proceedings of the IEEE*, 94(2):370–382, 2006.

[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, *CHES 2014*, volume 8731 of *LNCS*, pages 206–222. Springer, Heidelberg, September 2014.

[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, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 37–51. Springer, Heidelberg, May 1997.

[BDL+12] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. *Journal of Cryptographic Engineering*, 2(2):77–89, September 2012.

[BH15] Mihir Bellare and Viet Tung Hoang. Resisting randomness subversion: Fast deterministic and hedged public-key encryption in the standard model. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 627–656. Springer, Heidelberg, April 2015.

[BK03] Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 491–506. Springer, Heidelberg, May 2003.

[Ble00] Daniel Bleichenbacher. On the generation of one-time keys in DL signature schemes. Presentation at IEEE P1363 working group meeting, 2000.

[BP16] Alessandro Barenghi and Gerardo Pelosi. A note on fault attacks against deterministic signature schemes. In Kazuto Ogawa and Katsunari Yoshioka, editors, *IWSEC 16*, volume 9836 of *LNCS*, pages 182–192. Springer, Heidelberg, September 2016.

[BP18] Leon Groot Bruinderink and Peter Pessl. Differential fault attacks on deterministic lattice signatures. *IACR TCHES*, 2018(3):21–43, 2018. https://tches.iacr.org/index.php/TCHES/article/view/7267.

[BPS16] Mihir Bellare, Bertram Poettering, and Douglas Stebila. From identification to signatures, tightly: A framework and generic transforms. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 435–464. Springer, Heidelberg, December 2016.

[BPS17] Alexandra Boldyreva, Christopher Patton, and Thomas Shrimpton. Hedging public-key encryption in the real world. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 462–494. Springer, Heidelberg, August 2017.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006.

[BR18] Michael Brengel and Christian Rossow. Identifying key leakage of bitcoin users. In *RAID*, volume 11050 of *Lecture Notes in Computer Science*, pages 623–643. Springer, 2018.

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 513–525. Springer, Heidelberg, August 1997.

[BT16] Mihir Bellare and Björn Tackmann. Nonce-based cryptography: Retaining security when randomness fails. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 729–757. Springer, Heidelberg, May 2016.

[BV96] Dan Boneh and Ramarathnam Venkatesan. Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 129–142. Springer, Heidelberg, August 1996.

[CCD+17] Katriel Cohn-Gordon, Cas J. F. Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A formal security analysis of the signal messaging protocol. In *EuroS&P*, pages 451–466. IEEE, 2017.

[CDG+17] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1825–1842. ACM Press, October / November 2017.

[Cha19] André Chailloux. Quantum security of the Fiat-Shamir transform of commit and open protocols. Cryptology ePrint Archive, Report 2019/699, 2019. https://eprint.iacr.org/2019/699.

[Cla07] Christophe Clavier. Secret external encodings do not prevent transient fault analysis. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES 2007*, volume 4727 of *LNCS*, pages 181–194. Springer, Heidelberg, September 2007.

[CM09] Jean-Sébastien Coron and Avradip Mandal. PSS is secure against random fault attacks. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 653–666. Springer, Heidelberg, December 2009.

[CPS+16] Michele Ciampi, Giuseppe Persiano, Alessandra Scafuro, Luisa Siniscalchi, and Ivan Visconti. Improved OR-composition of sigma-protocols. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 112–141. Springer, Heidelberg, January 2016.

[DAN+18] Lauren De Meyer, Victor Arribas, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. M&M: Masks and macs against physical attacks. *IACR TCHES*, 2019(1):25–50, 2018. https://tches.iacr.org/index.php/TCHES/article/view/7333.

[DDE+19] Joan Daemen, Christoph Dobraunig, Maria Eichlseder, Hannes Gross, Florian Mendel, and Robert Primas. Protecting against statistical ineffective fault attacks. Cryptology ePrint Archive, Report 2019/536, 2019. https://eprint.iacr.org/2019/536.

[DEG+18] Christoph Dobraunig, Maria Eichlseder, Hannes Groß, Stefan Mangard, Florian Mendel, and Robert Primas. Statistical ineffective fault attacks on masked AES with fault countermeasures. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 315–342. Springer, Heidelberg, December 2018.

[DEK+18] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Stefan Mangard, Florian Mendel, and Robert Primas. SIFA: Exploiting ineffective fault inductions on symmetric cryptography. *IACR TCHES*, 2018(3):547–572, 2018. https://tches.iacr.org/index.php/TCHES/article/view/7286.

[DFMV17] Ivan Damgård, Sebastian Faust, Pratyay Mukherjee, and Daniele Venturi. Bounded tamper resilience: How to go beyond the algebraic barrier. *Journal of Cryptology*, 30(1):152–190, January 2017.

[DPW18] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. *J. ACM*, 65(4):20:1–20:32, 2018.

[fai10] fail0verflow. Console hacking 2010 – PS3 epic fail. 27th Chaos Communications Congress, 2010.

[FG20] Marc Fischlin and Felix Günther. Modeling memory faults in signature and authenticated encryption schemes. In Stanislaw Jarecki, editor, *CT-RSA 2020*, volume 12006 of *LNCS*, pages 56–84. Springer, 2020.

[FNSV18] Antonio Faonio, Jesper Buus Nielsen, Mark Simkin, and Daniele Venturi. Continuously non-malleable codes with split-state refresh. In Bart Preneel and Frederik Vercauteren, editors, *ACNS 18*, volume 10892 of *LNCS*, pages 121–139. Springer, Heidelberg, July 2018.

[FPV11] Sebastian Faust, Krzysztof Pietrzak, and Daniele Venturi. Tamper-proof circuits: How to trade leakage for tamper-resilience. In Luca Aceto, Monika Henzinger, and Jiri Sgall, editors, *ICALP 2011, Part I*, volume 6755 of *LNCS*, pages 391–402. Springer, Heidelberg, July 2011.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

[FV16] Antonio Faonio and Daniele Venturi. Efficient public-key cryptography with bounded leakage and tamper resilience. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 877–907. Springer, Heidelberg, December 2016.

[FX16] Eiichiro Fujisaki and Keita Xagawa. Public-key cryptosystems resilient to continuous tampering and leakage of arbitrary functions. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 908–938. Springer, Heidelberg, December 2016.

[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*, volume 2951 of *LNCS*, pages 258–277. Springer, Heidelberg, February 2004.

[GMO16] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. ZKBoo: Faster zero-knowledge for boolean circuits. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016*, pages 1069–1083. USENIX Association, August 2016.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *27th FOCS*, pages 174–187. IEEE Computer Society Press, October 1986.

[Gol07] Oded Goldreich. *Foundations of Cryptography*, volume 1. Cambridge University Press, 2007.

[HL10] Carmit Hazay and Yehuda Lindell. *Efficient Secure Two-Party Protocols - Techniques and Constructions*. Information Security and Cryptography. Springer, 2010.

[HLC+18] Zhengan Huang, Junzuo Lai, Wenbin Chen, Man Ho Au, Zhen Peng, and Jin Li. Hedged nonce-based public-key encryption: Adaptive security under randomness failures. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 253–279. Springer, Heidelberg, March 2018.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 21–30. ACM Press, June 2007.

[IPSW06] Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private circuits II: Keeping secrets in tamperable circuits. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 308–327. Springer, Heidelberg, May / June 2006.

[JL17] S. Josefsson and I. Liusvaara. Edwards-curve digital signature algorithm (EdDSA). RFC 8032, 2017. https://tools.ietf.org/html/rfc8032.

[JT12] Marc Joye and Michael Tunstall. *Fault analysis in cryptography*, volume 147 of *Information Security and Cryptography*. Springer, 2012.

[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 *ISCA*, pages 361–372. IEEE Computer Society, 2014.

[KKW18] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 525–537. ACM Press, October 2018.

[KLS18] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 552–586. Springer, Heidelberg, April / May 2018.

[KM15] Neal Koblitz and Alfred J. Menezes. The random oracle model: a twenty-year retrospective. *Designs, Codes and Cryptography*, 77(2-3):587–610, 2015.

[KMO90] Joe Kilian, Silvio Micali, and Rafail Ostrovsky. Minimum resource zero-knowledge proofs (extended abstract). In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 545–546. Springer, Heidelberg, August 1990.

[KMP16] Eike Kiltz, Daniel Masny, and Jiaxin Pan. Optimal security proofs for signatures from identification schemes. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 33–61. Springer, Heidelberg, August 2016.

[KSV13] Dusko Karaklajic, Jörn-Marc Schmidt, and Ingrid Verbauwhede. Hardware designer's guide to fault attacks. *IEEE Trans. VLSI Syst.*, 21(12):2295–2306, 2013.

[LDK+19] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[LL12] Feng-Hao Liu and Anna Lysyanskaya. Tamper and leakage resilience in the split-state model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 517–532. Springer, Heidelberg, August 2012.

[Lyu09] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, Heidelberg, December 2009.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755. Springer, Heidelberg, April 2012.

[MHER14] Nicolas Moro, Karine Heydemann, Emmanuelle Encrenaz, and Bruno Robisson. Formal verification of a software countermeasure against instruction skip attacks. *Journal of Cryptographic Engineering*, 4(3):145–156, September 2014.

[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*, volume 1556 of *LNCS*, pages 72–80. Springer, Heidelberg, August 1999.

[MSM+16] Hiraku Morita, Jacob C. N. Schuldt, Takahiro Matsuda, Goichiro Hanaoka, and Tetsu Iwata. On the Security of the Schnorr Signature Scheme and DSA Against Related-Key Attacks. In Soonhak Kwon and Aaram Yun, editors, *ICISC 2015*, Lecture Notes in Computer Science, pages 20–35. Springer, 2016.

[Nec94] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 1994.

[OO98] Kazuo Ohta and Tatsuaki Okamoto. On concrete security treatment of signatures derived from identification. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 354–369. Springer, Heidelberg, August 1998.

[Per16] Trevor Perrin. *The XEdDSA and VXEdDSA Signature Schemes*. Signal, 2016. Revision 1, https://signal.org/docs/specifications/xeddsa/.

[Pic19a] The Picnic Design Team. *The Picnic Signature Algorithm Specification*, July 2019. Version 2.1, Available at https://microsoft.github.io/Picnic/.

[Pic19b] The Picnic Design Team. *The Picnic Signature Scheme Design Document*, March 2019. Version 2.0, Available at https://microsoft.github.io/Picnic/.

[Por13] Thomas Pornin. Deterministic usage of the digital signature algorithm (DSA) and elliptic curve digital signature algorithm (ECDSA). RFC 6979, 2013. https://tools.ietf.org/html/rfc6979.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000.

[PSS14] Kenneth G. Paterson, Jacob C. N. Schuldt, and Dale L. Sibborn. Related randomness attacks for public key encryption. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 465–482. Springer, Heidelberg, March 2014.

[PSS+18] Damian Poddebniak, Juraj Somorovsky, Sebastian Schinzel, Manfred Lochter, and Paul Rosler. Attacking Deterministic Signature Schemes using Fault Attacks. In *Euro S&P 2018*, pages 338–352. IEEE, 2018.

[qTE19] The qTESLA Team. *Submission to NIST's post-quantum project (2nd round): lattice-based digital signature scheme qTESLA*, August 2019. Version 2.7, Available at https://qtesla.org/.

[RG14] Pablo Rauzy and Sylvain Guilley. A formal proof of countermeasures against fault injection attacks on CRT-RSA. *Journal of Cryptographic Engineering*, 4(3):173–185, September 2014.

[RJH+19] Prasanna Ravi, Mahabir Prasad Jhanwar, James Howe, Anupam Chattopadhyay, and Shivam Bhasin. Exploiting Determinism in Lattice-based Signatures: Practical Fault Attacks on Pqm4 Implementations of NIST Candidates. In *Asia CCS 2019*, Asia CCS '19, pages 427–440. ACM, 2019.

[RP17] Y. Romailler and S. Pelissier. Practical Fault Attack against the Ed25519 and EdDSA Signature Schemes. In *FDTC 2017*, pages 17–24, September 2017.

[RY10] Thomas Ristenpart and Scott Yilek. When good randomness goes bad: Virtual machine reset vulnerabilities and hedging deployed cryptography. In *NDSS 2010*. The Internet Society, February / March 2010.

[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*, volume 10831 of *LNCS*, pages 306–321. Springer, Heidelberg, May 2018.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991.

[Sch16] Benedikt Schmidt. [curves] EdDSA specification. https://moderncrypto.org/mail-archive/curves/2016/000768.html, 2016.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.

[TTA18] Akira Takahashi, Mehdi Tibouchi, and Masayuki Abe. New Bleichenbacher records: Fault attacks on qDSA signatures. *IACR TCHES*, 2018(3):331–371, 2018. https://tches.iacr.org/index.php/TCHES/article/view/7278.

[YJ00] Sung-Ming Yen and Marc Joye. Checking before output may not be enough against fault-based cryptanalysis. *IEEE Trans. Computers*, 49(9):967–970, 2000.

[ZCD+19] Greg Zaverucha, Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, Jonathan Katz, Xiao Wang, and Vladmir Kolesnikov. Picnic. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

A Omitted Proofs

A.1 Proof of Lemma 2

Lemma 2 (UF-KOA $\to$ UF-CMA)**.** *Let* ID *be a correct canonical identification protocol and* CSF *be a canonical serialization function for* ID*. Suppose* ID *satisfies the following:*

- ID *is special c/s/p-HVZK with efficient distinguishers' advantage being at most* ϵ_{HVZK}*.*
- ID *has* α*-bit min-entropy.*

If $\mathsf{FS} := \mathbf{FS}[\mathsf{ID}, \mathsf{CSF}]$ *is* UF-KOA *secure, then* FS *is* UF-CMA *secure in the random oracle model. Concretely, given a* UF-CMA *adversary* $\mathcal{A}$ *against* FS *running in time* t*, and making at most* Q_s *queries to* OSign*,* Q_h *queries to* H*, one can construct another adversary* $\mathcal{B}$ *against* FS *such that*

$$\mathbf{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathsf{FS}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{UF\text{-}KOA}}_{\mathsf{FS}}(\mathcal{B}) + \frac{(Q_s + Q_h)Q_s}{2^{\alpha}} + Q_s \cdot \epsilon_{HVZK},$$

where $\mathcal{B}$ *makes at most* Q_h *queries to its random oracle, and runs in time* t*.*

Proof. The proof is essentially same as UF-KOA-to-UF-CMA reduction in [KMP16], but we also consider the case ID is only non-perfect HVZK. We show that UF-KOA security of a randomized Fiat–Shamir signature scheme FS can be broken by letting $\mathcal{B}$ simulate OSign without using sk. We denote the random oracle and hash table in UF-CMA (resp. UF-KOA) by H and HT (resp. H′ and HT′).

Preparation of Public Key Upon receiving pk in UF-KOA game, $\mathcal{B}$ forwards pk to $\mathcal{A}$.

Simulation of Random Oracle Queries Upon receiving a random oracle query $\mathsf{H}(a, m, pk)$ from $\mathcal{A}$, $\mathcal{B}$ forwards the input (a, m, pk) to its own random oracle and provides $\mathcal{A}$ with the return value of $\mathsf{H}'(a, m, pk)$.

Simulation of Faulty Signing Queries For each $i \in [Q_s]$, $\mathcal{B}$ answers a faulty signing request from $\mathcal{A}$ by simulating the signature on m_i as follows:

1. $\mathcal{B}$ first samples e_i from D_H uniformly at random.
2. $\mathcal{B}$ invokes the special c/s/p-HVZK simulator $\mathcal{M}$ of ID on input pk and e_i and on freshly generated randomness every time, to obtain an accepting transcript (a_i, e_i, z_i).

3. If $\mathrm{HT}[a_i, m_i, pk]$ is already set (via $\mathcal{A}$'s previous hash or signing queries) and $\mathrm{HT}[a_i, m_i, pk] \neq e_i$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ programs the random oracle so that

$$\mathrm{HT}[a_i, m_i, pk] \coloneqq e_i.$$

Notice that programming makes H's output inconsistent with that of the external random oracle H', but as we will see later it doesn't make a difference since a forgery must use a previously unqueried m.

4. If random oracle programming is successful, $\mathcal{B}$ gives $\mathcal{A}$ a serialized transcript

$$\sigma_i \coloneqq \mathsf{CSF}(a_i, e_i, z_i).$$

Now we evaluate the probability that $\mathcal{A}$ distinguishes the simulated signing oracle above from the real one. There are essentially two cases.

- $\mathcal{B}$ aborts at the third step in the above simulation with probability at most $(Q_h + Q_s)/2^\alpha$ per each query. This is because the input (a_i, m_i, pk) to H has α bits of min-entropy (since m_i is chosen by the adversary and has zero bits, pk is public and has zero bits, and a_i has α bits due to α-bit min-entropy of ID), and HT has at most $Q_h + Q_s$ existing entries.
- $\mathcal{A}$ after the forth step distinguishes σ_i from the one returned by the real signing oracle OSign with probability at most ϵ_{HVZK} per each query, since we used the special c/s/p-HVZK simulator to derive (a_i, e_i, z_i) and CSF is efficiently computable.

Recalling that the number of signing queries is bounded by Q_s, and by a union bound, $\mathcal{A}$ overall distinguishes its simulated view from that in $\mathsf{UF\text{-}CMA}$ game with probability at most

$$\frac{(Q_h + Q_s)Q_s}{2^\alpha} + Q_s \cdot \epsilon_{HVZK}.$$

Forgery Suppose that at the end of the experiment $\mathcal{A}$ outputs its forgery (m^*, σ^*) that passes the verification and $m^* \notin M = \{m_i : i \in [Q_s]\}$. This means that the reconstructed transcript $(a^*, e^*, z^*) \leftarrow \mathsf{CDF}(\sigma^*, pk)$ satisfies

$$\mathsf{V}(a^*, e^*, z^*, pk) = 1 \quad \text{and} \quad \mathsf{H}(a^*, m^*, pk) = e^*.$$

Here we can guarantee that the $\mathrm{HT}[a^*, m^*, pk]$ has not been programmed by signing oracle simulation since m^* is fresh, i.e., $m^* \notin M$. Hence we ensure that $e^* = \mathrm{HT}[a^*, m^*, pk]$ has been directly set by $\mathcal{A}$, and $e^* = \mathrm{HT}'[a^*, m^*, pk]$ holds due to the hash query simulation. This implies (m^*, σ^*) is a valid forgery in $\mathsf{UF\text{-}KOA}$ game as well. □

A.2 Proof of Lemma 3

Lemma 3 ($\mathsf{UF\text{-}CMA} \to \mathsf{UF\text{-}CMNA}$)**.** *Let* $\mathsf{SIG} \coloneqq (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *be a randomized digital signature scheme, and let* $\mathsf{HSIG} \coloneqq \mathbf{R2H}[\mathsf{SIG}, \mathsf{HE}] = (\mathsf{Gen}, \mathsf{HSign}, \mathsf{Verify})$ *be the corresponding hedged signature scheme with* HE *modeled as a random oracle. If* SIG *is* $\mathsf{UF\text{-}CMA}$ *secure, then* HSIG *is* $\mathsf{UF\text{-}CMNA}$ *secure. Concretely, given a* $\mathsf{UF\text{-}CMNA}$ *adversary* $\mathcal{A}$ *against* HSIG *running in time* t*, and making at most* Q_s *queries to* OHSign *and* Q_{he} *queries to* HE*, one can construct an adversary* $\mathcal{B}$ *against* SIG *such that*

$$\mathbf{Adv}^{\mathsf{UF\text{-}CMNA}}_{\mathsf{HSIG},\mathsf{HE}}(\mathcal{A}) \leq 2 \cdot \mathbf{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{B}),$$

where $\mathcal{B}$ *makes at most* Q_s *queries to its signing oracle, and has running time* t *plus* Q_{he} *invocations of* Sign *and* Verify*.*

Proof. We rely on the code-based game-playing proof [BR06], and the basic structure of our proof is essentially same as the reduction from $\mathsf{UUF\text{-}CMA}$ to $\mathsf{UF\text{-}CMA}$ for deterministic signature schemes [BPS16]. Let us consider two *identical-until-bad* games described in Fig. 9: $\mathrm{G}_0(\mathcal{A})$ (without boxed code) and $\mathrm{G}_1(\mathcal{A})$ (with boxed code). We assume wlog that $\mathcal{A}$ does not repeat the same HE query. Notice that $\mathrm{G}_0(\mathcal{A})$ is identical to

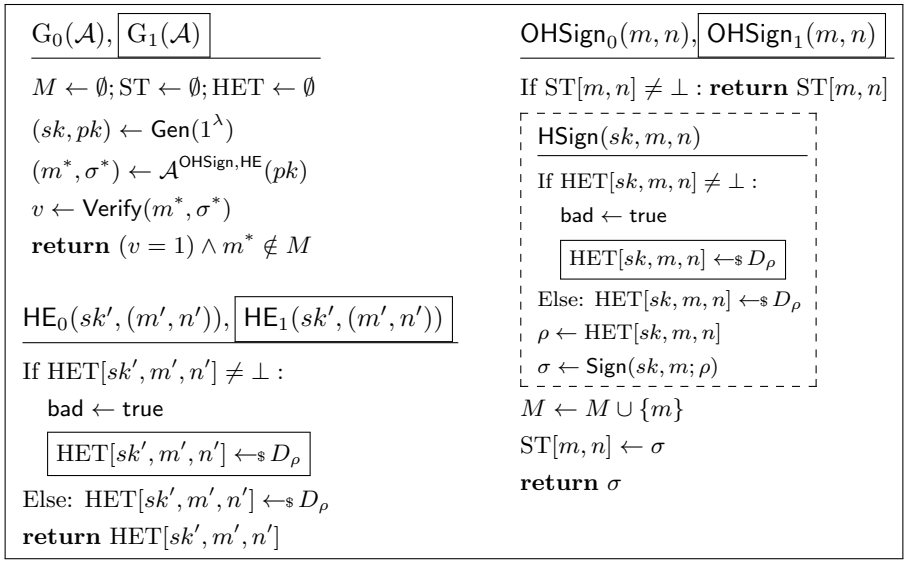

Figure 9: Two identical-until-bad games $\mathrm{G}_0(\mathcal{A})$ and $\mathrm{G}_1(\mathcal{A})$. The game $\mathrm{G}_1(\mathcal{A})$ executes the boxed code, while $\mathrm{G}_0(\mathcal{A})$ does not. The dashed box indicates that the instructions inside correspond to the actual hedged signing operation.

$\mathsf{Exp}^{\mathsf{UF\text{-}CMNA}}_{\mathsf{HSIG},\mathsf{HE}}(\mathcal{A})$ in Fig. 6 and therefore $\mathbf{Adv}^{\mathsf{UF\text{-}CMNA}}_{\mathsf{HSIG},\mathsf{HE}}(\mathcal{A}) = \Pr[\mathrm{G}_0(\mathcal{A})]$. Now we obtain the following by simple transformation:

$$\begin{aligned}\mathbf{Adv}^{\mathsf{UF\text{-}CMNA}}_{\mathsf{HSIG},\mathsf{HE}}(\mathcal{A}) = \Pr[\mathrm{G}_0(\mathcal{A})] &= \Pr[\mathrm{G}_1(\mathcal{A})] + (\Pr[\mathrm{G}_0(\mathcal{A})] - \Pr[\mathrm{G}_1(\mathcal{A})]) \\ &\leq \Pr[\mathrm{G}_1(\mathcal{A})] + \Pr[\mathrm{G}_1(\mathcal{A}) \text{ sets } \mathsf{bad}].\end{aligned}$$

Our goal is to construct the adversary $\mathcal{B}$ breaking UF-CMA security of a plain randomized signature scheme SIG such that

$$\Pr[\mathrm{G}_1(\mathcal{A})] \leq \mathbf{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{B}) \quad \text{and} \quad \Pr[\mathrm{G}_1(\mathcal{A}) \text{ sets } \mathsf{bad}] \leq \mathbf{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{B}).$$

The description of $\mathcal{B}$ is given in Fig. 10. The adversary $\mathcal{B}$ perfectly simulates $\mathcal{A}$'s view in $\mathrm{G}_1(\mathcal{A})$ as follows.

- $\mathsf{Sim\text{-}OHSign}_1$ simulates OHSign_1 by forwarding signing queries to OSign, the oracle in UF-CMA game. Notice that this simulation is perfect since both OHSign_1 and OSign use freshly generated randomness for every signing query, except when the same (m, n) pair is queried.
- $\mathsf{Sim\text{-}HE}_1$ simulates the HE_1 but keeps track of candidate secret keys queried by $\mathcal{A}$ in S.

If the adversary $\mathcal{A}$ wins the game $\mathrm{G}_1(\mathcal{A})$, this implies that $\mathcal{B}$ receives a valid forgery (m', σ') from $\mathcal{A}$ so that $\mathcal{B}$ can win UF-CMA game, i.e., $\Pr[\mathrm{G}_1(\mathcal{A})] \leq \mathbf{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{B})$. Otherwise, $\mathcal{B}$ picks some message $m^* \notin M$ and tries to sign with all secret key candidates sk^*. Below we will see why this strategy guarantees $\mathcal{B}$ to generate a valid forgery as long as bad is set in $\mathrm{G}_1(\mathcal{A})$, i.e., $\Pr[\mathrm{G}_1(\mathcal{A}) \text{ sets } \mathsf{bad}] \leq \mathbf{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{B})$.

Here we show that if bad is set either by OHSign_1 or HE_1, then $\mathcal{A}$ must have queried HE_1 with $(sk^*, (m^*, n^*))$ for some (m^*, n^*) such that $sk^* = sk$. First, if bad is set by OHSign_1, it must be due to a previous query to HE_1 with $(sk', (m', n'))$ such that $sk' = sk$ holds, since OHSign, without invoking HSign, returns previously generated signature σ stored in ST whenever the same (m, n) pair is queried. Second, we consider the case bad is set by HE_1. Since we assumed that $\mathcal{A}$ does not repeat the same HE query, if a query $(sk', (m', n'))$ to HE_1 sets bad then $\mathrm{HET}[sk', m', n']$ must have been defined by OHSign_1. Here notice that we have $sk' = sk$ due to the definition of OHSign_1. Thus if bad is set in $\mathrm{G}_1(\mathcal{A})$ then for some $sk^* \in S$, $sk^* = sk$ holds. By checking the validity of signatures generated with each $sk^* \in S$, $\mathcal{B}$ eventually finds the true secret key sk that leads to a valid forgery.

We finally observe that the total running time of $\mathcal{B}$ is upper bounded by $\mathcal{A}$'s running time t plus Q_{he} times the running time of Sign and Verify due to the above for-loop operations.

□

$\mathcal{B}^{\mathsf{OSign}}(pk)$

$M \leftarrow \emptyset; \mathrm{ST} \leftarrow \emptyset; \mathrm{HET} \leftarrow \emptyset; S \leftarrow \emptyset$
$(m', \sigma') \leftarrow \mathcal{A}^{\mathsf{Sim\text{-}OHSign}_1, \mathsf{Sim\text{-}HE}_1}(pk)$
If $\mathsf{Verify}(m', \sigma') = 1 \wedge m' \notin M$:
  return (m', σ')
Pick some $m^* \notin M$
$\rho^* \leftarrow\!\!\$\ D_\rho$
For $sk^* \in S$:
  $\sigma^* \leftarrow \mathsf{Sign}(sk^*, m^*; \rho^*)$
  If $\mathsf{Verify}(m^*, \sigma^*) = 1$:
    return (m^*, σ^*)
return $(\perp, \perp)$

$\mathsf{Sim\text{-}OHSign}_1(m, n)$

If $\mathrm{ST}[m, n] \neq \perp$: **return** $\mathrm{ST}[m, n]$
$\sigma \leftarrow \mathsf{OSign}(m)$
$M \leftarrow M \cup \{m\}$
$\mathrm{ST}[m, n] \leftarrow \sigma$
return σ

$\mathsf{Sim\text{-}HE}_1(sk', (m', n'))$

$S \leftarrow S \cup \{sk'\}$
$\rho \leftarrow\!\!\$\ D_\rho$
return ρ

Figure 10: Description of $\mathcal{B}$

A.3 Proof of Lemma 6

Lemma 6 ($\mathsf{UF\text{-}KOA} \to \{f_8, f_9, f_{10}\}\text{-}\mathsf{UF\text{-}fCMA}$)**.** *If* $\mathsf{FS} := \mathbf{FS}[\mathsf{ID}, \mathsf{CSF}]$ *is* $\mathsf{UF\text{-}KOA}$ *secure, then* FS *is* $\{f_8, f_9, f_{10}\}$-$\mathsf{UF\text{-}fCMA}$ *secure in the random oracle model. Concretely, given an* $\{f_8, f_9, f_{10}\}$*-adversary* $\mathcal{A}$ *against* FS *running in time* t*, and making at most* Q_s *queries to* $\mathsf{OFaultSign}$*,* Q_h *queries to* H*, one can construct another adversary* $\mathcal{B}$ *against* FS *such that*

$$\mathbf{Adv}^{\mathsf{UF\text{-}fCMA}}_{\mathsf{FS}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{UF\text{-}KOA}}_{\mathsf{FS}}(\mathcal{B}) + \frac{(Q_s + Q_h)Q_s}{2^\alpha} + Q_s \cdot \epsilon_{HVZK},$$

where $\mathcal{B}$ *makes at most* Q_h *queries to its random oracle, and has running time* t*.*

Proof. We denote the random oracle and hash table in $\mathsf{UF\text{-}fCMA}$ experiment (resp. $\mathsf{UF\text{-}KOA}$ experiment) by H and HT (resp. H' and HT′).

Preparation of Public Key Upon receiving pk in the $\mathsf{UF\text{-}KOA}$ game, $\mathcal{B}$ forwards pk to $\mathcal{A}$.

Simulation of Random Oracle Queries Upon receiving a random oracle query $\mathsf{H}(a, m, pk)$ from $\mathcal{A}$, $\mathcal{B}$ forwards the input (a, m, pk) to its own random oracle (H' from the $\mathsf{UF\text{-}KOA}$ game) and provides $\mathcal{A}$ with the return value.

Simulation of Faulty Signing Queries For each $i \in [Q_s]$, $\mathcal{B}$ answers a faulty signing request from $\mathcal{A}$ by simulating the signature on m_i as follows:

1. $\mathcal{B}$ first samples e_i from D_H uniformly at random.
2. $\mathcal{B}$ invokes the special c/s/p-HVZK simulator $\mathcal{M}$ of ID on input pk and e_i and on freshly generated randomness every time, to obtain an accepting transcript (a_i, e_i, z_i).
3. If $\mathrm{HT}[a_i, m_i, pk]$ is already set (via $\mathcal{A}$'s previous hash or signing queries) and $\mathrm{HT}[a_i, m_i, pk] \neq e_i$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ programs the random oracle so that

   $$\mathrm{HT}[a_i, m_i, pk] := e_i.$$

   Notice that programming makes H's output inconsistent with that of the external random oracle H', but as we will see later it doesn't make a difference since a forgery must use a previously unqueried m.

4. If random oracle programming is successful, $\mathcal{B}$ gives $\mathcal{A}$ an incorrectly serialized transcript depending on a fault position j_i and tampering function ϕ_i:

$$\begin{aligned}
\tilde{\sigma}_i &\coloneqq \mathsf{CSF}(a_i, e_i, f_8(z_i)) && \text{if } j_i = 8 \\
\tilde{\sigma}_i &\coloneqq \mathsf{CSF}(f_9(a_i, e_i, z_i)) && \text{if } j_i = 9 \\
\tilde{\sigma}_i &\coloneqq f_{10}(\mathsf{CSF}(a_i, e_i, z_i)) && \text{if } j_i = 10 \\
\tilde{\sigma}_i &\coloneqq \mathsf{CSF}(a_i, e_i, z_i) && \text{if } \phi_j = Id.
\end{aligned}$$

Now we evaluate the probability that $\mathcal{A}$ distinguishes the simulated signing oracle above from the real one. There are essentially two cases.

- $\mathcal{B}$ aborts at the third step in the above simulation with probability at most $(Q_h + Q_s)/2^\alpha$ per query. This is because the input (a_i, m_i, pk) to H has α bits of min-entropy (since m_i is chosen by the adversary and has zero bits, pk is public and has zero bits, and a_i has α bits due to α-bit min-entropy of ID), and HT has at most $Q_h + Q_s$ existing entries.
- After the fourth step, $\mathcal{A}$ distinguishes $\tilde{\sigma}_i$ from the one returned by the real signing oracle $\mathsf{OFaultHSign}$ with probability at most ϵ_{HVZK} per query, since we used the special c/s/p-HVZK simulator to derive (a_i, e_i, z_i) and f_8, f_9, f_{10} and CSF are efficiently computable.

Recalling that the number of signing queries is bounded by Q_s, and by a union bound, $\mathcal{A}$ overall distinguishes its simulated view from that in $\mathsf{UF\text{-}fCMA}$ game with probability at most

$$\frac{(Q_h + Q_s)Q_s}{2^\alpha} + Q_s \cdot \epsilon_{HVZK}.$$

Forgery Suppose that at the end of the experiment $\mathcal{A}$ outputs its forgery (m^*, σ^*) that passes the verification and $m^* \notin M = \{\hat{m}_i : i \in [Q_s]\}$. This means that the reconstructed transcript $(a^*, e^*, z^*) \leftarrow \mathsf{CDF}(\sigma^*, pk)$ satisfies

$$\mathsf{V}(a^*, e^*, z^*, pk) = 1 \quad \text{and} \quad \mathsf{H}(a^*, m^*, pk) = e^*.$$

Here we can guarantee that the $\mathrm{HT}[a^*, m^*, pk]$ has not been programmed by the signing oracle simulation since m^* is fresh, i.e., $m^* \notin M$. Hence we ensure that $e^* = \mathrm{HT}[a^*, m^*, pk]$ has been directly set by $\mathcal{A}$, and $e^* = \mathrm{HT}'[a^*, m^*, pk]$ holds due to the hash query simulation. This implies (m^*, σ^*) is a valid forgery in $\mathsf{UF\text{-}KOA}$ game as well. □

B Schnorr-Like Elliptic Curve Signature Schemes

Here we present the Schnorr signature and two popular variants, all instantiated with elliptic curve groups. The base point of order q on the elliptic curve is denoted G. Functions H, H' and H'' are cryptographic hash functions, H and H' have 2λ-bit digests, and H'' has 4λ-bit digests.

In EdDSA, a single initial secret value k is used to derive the secret key sk, and a key K used to derive the per-signature random value ρ. In XEdDSA (Algorithm 2), a single secret key is used for both computing ρ and in the signing equation (matching our formalization of hedging in Fig. 5). The specification [Per16] contains a conversion step (called `calculate_key_pair`) at the beginning of signing, that converts a bitstring to sk and recomputes pk from sk. Since this step is both expensive and the same for every signature, we expect all implementations would cache the result and omit this step (matching our presentation here).

Algorithm 1 EC-Schnorr signing

Input: $sk \in \mathbb{Z}/q\mathbb{Z}$: signing key
$m \in \{0,1\}^*$: message to be signed

Output: a signature σ

1: $\rho \leftarrow\!\!\$\ D_\rho$
2: $(a, St) \leftarrow ([\rho]G, \rho)$ // $(a, St) \leftarrow \mathsf{Com}(sk; \rho)$
3: $e \leftarrow \mathsf{H}(a, m)$
4: $z \leftarrow \rho + e \cdot sk \mod q$ // $z \leftarrow \mathsf{Resp}(sk, e, St)$
5: $\sigma \leftarrow (e, z)$ // $\sigma \leftarrow \mathsf{CSF}(a, e, z)$
6: **return** σ

Algorithm 2 XEdDSA signing

Input: $sk \in \mathbb{Z}_q^*$: signing key
$pk = [sk]G$: public key
$m \in \{0,1\}^*$: message to be signed
$n \in \{0,1\}^{512}$: random value

Output: a signature σ

1: $\rho \leftarrow \mathsf{H}'(sk, m, n)$
2: $(a, St) \leftarrow ([\rho]G, \rho)$ // $(a, St) \leftarrow \mathsf{Com}(sk; \rho)$
3: $e \leftarrow \mathsf{H}(a, m, pk)$
4: $z \leftarrow \rho + e \cdot sk \mod q$ // $z \leftarrow \mathsf{Resp}(sk, e, St)$
5: $\sigma \leftarrow (a, z)$ // $\sigma \leftarrow \mathsf{CSF}(a, e, z)$
6: **return** σ

Algorithm 3 EdDSA signing

Input: $k \in \{0,1\}^{2\lambda}$: the secret key
$m \in \{0,1\}^*$: message to be signed

Output: a signature σ

1: $(sk, K) \leftarrow \mathsf{H}''(k)$
2: $pk \leftarrow [sk]G$
3: $\rho \leftarrow \mathsf{H}'(K, m)$
4: $(a, St) \leftarrow ([\rho]G, \rho)$ // $(a, St) \leftarrow \mathsf{Com}(sk; \rho)$
5: $e \leftarrow \mathsf{H}(a, m, pk)$
6: $z \leftarrow \rho + e \cdot sk \mod q$ // $z \leftarrow \mathsf{Resp}(sk, e, St)$
7: $\sigma \leftarrow (a, z)$ // $\sigma \leftarrow \mathsf{CSF}(a, e, z)$
8: **return** σ

C Description of Picnic2

Figure 11 shows Picnic2 signing and Figure 12 shows verification. The Picnic2 ID scheme is obtained by undoing the Fiat–Shamir transform. There are a few differences with respect to the Picnic specification [Pic19a], made here for simplicity.

In particular the specification includes a per-signature, 256-bit salt value, chosen by the signer, and included in all hash computations. The specification also derives the per-instance seeds $\{\mathsf{seed}_j^*\}_{j=1}^M$, from a single root seed, using a tree construction. Similarly, the per-party, per-instance seeds $\{\mathsf{seed}_{i,j}\}$ are derived from seed_j^* using the same tree construction. Since most of the seeds are revealed, the signer can reduce the signature size by revealing intermediate nodes in the tree. Another optimization omitted here is that the commitments to the views h'_j are formed using a Merkle tree, and the root of the tree is input to G. For the values h'_j not recomputed by the verifier, the signer includes the path from h'_j to the root. Again this reduces the signature size significantly.

The specification also recommends a hedging construction that is an instance of the **R2H** construction from Section 4. In this case, the salt and random seeds are derived deterministically from $sk\|m\|pk\|n$ where n is a 2λ-bit random value (acting as the nonce in the notation of Section 4).

$\mathsf{Gen}(1^\lambda)$ In the presentation below, the key pair is $(pk, sk) = (C, w)$, a circuit C and an input w such that $C(w) = 1$. Concretely, the circuit is $E_w(p) \stackrel{?}{=} c$, where E is the LowMC block cipher with λ-bit key and block size, p is a random λ-bit plaintext and c is a ciphertext. If the input w is a block cipher key that maps p to c, the circuit outputs 1.

Picnic2 $\mathsf{Sign}(m;\rho)$

Values (M, n, τ) are parameters of the protocol.

Parse ρ as $(\mathsf{seed}_1^*, \ldots, \mathsf{seed}_M^*)$, where each value is λ bits long.

$\mathsf{Com}(w;\rho)$ For each $j \in [M]$:

1. Use seed_j^* to generate values $\mathsf{seed}_{j,1}$, ..., $\mathsf{seed}_{j,n}$ with a PRG. Also compute $\mathsf{aux}_j \in \{0,1\}^{|C|}$ as described in the text. For $i = 1, \ldots, n-1$, let $\mathsf{state}_{j,i} := \mathsf{seed}_{j,i}$; let $\mathsf{state}_{j,n} := \mathsf{seed}_{j,n} \| \mathsf{aux}_j$.
2. For $i \in [n]$, compute $\mathsf{com}_{j,i} := H_0(\mathsf{state}_{j,i})$.
3. The signer runs the online phase of the n-party protocol Π (as described in the text) using $\{\mathsf{state}_{j,i}\}_i$, beginning by computing the masked inputs $\{\hat{z}_{j,\alpha}\}$ (based on w and the $\{\lambda_{j,\alpha}\}$ defined by the preprocessing). Let $\mathsf{msgs}_{j,i}$ denote the messages broadcast by S_i in this protocol execution.
4. Let $h_j := H_1(\mathsf{com}_{j,1}, \ldots, \mathsf{com}_{j,n})$ and let $h'_j := H_2(\{\hat{z}_{j,\alpha}\}, \mathsf{msgs}_{j,1}, \ldots, \mathsf{msgs}_{j,n})$.

Let $a := (h_1, h'_1, \ldots, h_M, h'_M)$.
Let $St := \{\mathsf{seed}_j^*\}, \{\mathsf{com}_{j,i}\}, \{\mathsf{state}_{j,i}\}, \{h_j\}, \{h'_j\}$ for $j \in [M], i \in [n]$.

Challenge Compute $(\mathcal{C}, \mathcal{P}) := G(a, m)$, where $\mathcal{C} \subset [M]$ is a set of size τ, and $\mathcal{P}$ is a list $\{p_j\}_{j \in \mathcal{C}}$ with $p_j \in [n]$. Let $e := (\mathcal{C}, \mathcal{P})$.

$\mathsf{Resp}(w, e, St)$ Initialize the output list z. For each $j \in [M] \setminus \mathcal{C}$, add seed_j^*, h'_j to z. Also, for each $j \in \mathcal{C}$, add $\{\mathsf{state}_{j,i}\}_{i \neq p_j}$, com_{j,p_j}, $\{\hat{z}_{j,\alpha}\}$, and msgs_{j,p_j} to z.

Serialize Output (e, z).

Figure 11: The signing algorithm in the Picnic2 signature scheme.

Picnic2 Verification $\mathsf{Verify}(\sigma, m)$

Parse the signature $\sigma = (e, z)$ as $(\mathcal{C}, \mathcal{P}, \{\mathsf{seed}_j^*, h'_j\}_{j \notin \mathcal{C}}, \{\{\mathsf{state}_{j,i}\}_{i \neq p_j}, \mathsf{com}_{j,p_j}, \{\hat{z}_{j,\alpha}\}, \mathsf{msgs}_{j,p_j}\}_{j \in \mathcal{C}})$. Verify the signature as follows:

1. For every $j \in \mathcal{C}$ and $i \neq p_j$, set $\mathsf{com}_{j,i} := H_0(\mathsf{state}_{j,i})$; then compute the value $h_j := H_1(\mathsf{com}_{j,1}, \ldots, \mathsf{com}_{j,n})$.
2. For $j \notin \mathcal{C}$, use seed_j^* to compute h_j as the signer would.
3. For each $j \in \mathcal{C}$, run an execution of Π among the parties $\{S_i\}_{i \neq p_j}$ using $\{\mathsf{state}_{j,i}\}_{i \neq p_j}$, $\{\hat{z}_\alpha\}$, and msgs_{j,p_j}; this yields $\{\mathsf{msgs}_i\}_{i \neq p_j}$ and an output bit b. Check that $b \stackrel{?}{=} 1$. Then compute $h'_j := H_2(\{\hat{z}_{j,\alpha}\}\ \mathsf{msgs}_{j,1}, \ldots, \mathsf{msgs}_{j,n})$.
4. Output 1 (valid) if $(\mathcal{C}, \mathcal{P}) \stackrel{?}{=} G(m, h_1, h'_1, \ldots, h_M, h'_M)$, otherwise output 0 (invalid).

Figure 12: The verification algorithm in the Picnic2 signature scheme.