

# Security of Hedged Fiat–Shamir Signatures under Fault Attacks\*

Diego F. Aranha<sup>1</sup>, Claudio Orlandi<sup>2</sup>, Akira Takahashi<sup>2</sup>, and Greg Zaverucha<sup>3</sup>

<sup>1</sup>Department of Engineering, DIGIT, Aarhus University  
dfaranha@eng.au.dk

<sup>2</sup>Department of Computer Science, DIGIT, Aarhus University  
orlandi@cs.au.dk, takahashi@cs.au.dk

<sup>3</sup>Microsoft Research  
gregz@microsoft.com

February 23, 2021

## 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

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Our Contributions . . . . .	4
1.2	Related Work . . . . .	5
1.3	Paper Organization . . . . .	7
<b>2</b>	<b>Preliminaries</b>	<b>8</b>
2.1	Fiat–Shamir type Signature Schemes . . . . .	8
2.2	Definitions . . . . .	9
2.3	Relation between UF-KOA Security and UF-CMA Security . . . . .	12
<b>3</b>	<b>Fault Attacks on Deterministic Fiat–Shamir Signatures</b>	<b>13</b>
3.1	Conceptual Overview of the Attacks . . . . .	13
3.2	Applicability to Deterministic Picnic2 . . . . .	15
<b>4</b>	<b>Formal Treatment of Hedged Signatures</b>	<b>16</b>
4.1	Security of Hedged Signature Schemes . . . . .	16
4.2	Security of Hedged FS Type Signature Schemes Against Fault Adversaries . . . . .	17
<b>5</b>	<b>Security Analysis of Generic Hedged Fiat–Shamir Type Signatures Against Fault Attacks</b>	<b>19</b>
5.1	Main Positive Result . . . . .	21
5.2	Faulting Serialization Input/Output and Response Output . . . . .	25
5.3	Faulting Challenge Hash Input . . . . .	25
5.4	Faulting Challenge Hash Output . . . . .	26
5.5	Faulting Response Input . . . . .	27
5.6	Faulting Commitment Output . . . . .	29
5.7	Negative Results . . . . .	29
<b>6</b>	<b>Analysis of XEdDSA</b>	<b>31</b>
<b>7</b>	<b>Analysis of Picnic2</b>	<b>31</b>
7.1	Preliminaries . . . . .	31
7.2	Applying the Results of Section 5 . . . . .	34
7.3	Picnic-Specific Results . . . . .	35
<b>8</b>	<b>Concluding Remarks</b>	<b>35</b>
<b>A</b>	<b>Omitted Proofs</b>	<b>43</b>
A.1	Proof of Lemma 2 . . . . .	43
A.2	Proof of Lemma 3 . . . . .	45
A.3	Proof of Lemma 6 . . . . .	46
<b>B</b>	<b>Schnorr-Like Elliptic Curve Signature Schemes</b>	<b>48</b>
<b>C</b>	<b>Description of Picnic2</b>	<b>49</b>
<b>D</b>	<b>Technicalities in the Definition of c-HVZK</b>	<b>50</b>

# 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<sup>+</sup>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<sup>+</sup>18]. Recent papers have experimentally demonstrated the feasibility of these attacks [RP17, PSS<sup>+</sup>18, SB18]. Moreover, [BP18] and [RJH<sup>+</sup>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<sup>+</sup>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, VEdDSA [Per16], qTESLA [BAA<sup>+</sup>19], and Picnic2 [ZCD<sup>+</sup>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<sup>+</sup>19], both designed using the hedged construction. The XEdDSA scheme is used in the Signal protocol [CCD<sup>+</sup>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<sup>+</sup>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: `flip_biti(x)` to perform a logical negation of the  $i$ -th bit of  $x$ , and `set_biti,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  $f_0, \dots, f_{10}$ . 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

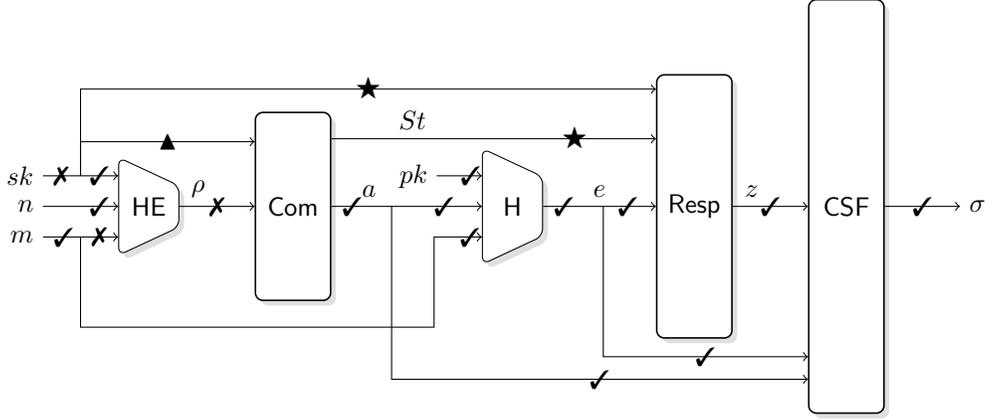


Figure 1: Overview of our results for hedged Fiat–Shamir type signature schemes. ✓ indicates security against 1-bit fault on the corresponding wire value, and ✗ indicates an attack or counterexample. A ★ (resp. ▲) indicates that security only holds for the schemes derived from subset-revealing ID (resp. 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<sup>+</sup>18, PSS<sup>+</sup>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<sup>+</sup>09, BH15, BPS17, HLC<sup>+</sup>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<sup>+</sup>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<sup>+</sup>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<sup>+</sup>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<sup>+</sup>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  $\Sigma$ -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 were 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<sup>+</sup>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<sup>+</sup>06] and the infective countermeasure [YJ00]. IFA has been recently superseded by *statistical ineffective fault attacks (SIFA)* [DEK<sup>+</sup>18, DEG<sup>+</sup>18], that use statistical analysis to enable mounting IFA with low-precision bit-fixing, random or bit-flip faults. Daemen et al. [DDE<sup>+</sup>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<sup>+</sup>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<sup>+</sup>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 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  $x \leftarrow_s X$  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  $\{1, \dots, n\}$  where  $n \in \mathbb{N}$ . When we explicitly mention that an algorithm  $A$  is randomized, we use the notation  $A(x; \rho)$  meaning that it is executed on input  $x$  with random tape  $\rho$ .

### 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<sup>+</sup>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<sup>+</sup>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<sup>+</sup>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<sup>+</sup>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.

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

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

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

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  $\text{SIG} = (\text{Gen}, \text{Sign}, \text{Verify})$ , is defined as follows.

- $\text{Gen}(1^\lambda)$ , where  $\lambda$  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  $\rho$  of length  $\ell_\rho$ -bit is sampled.
- $\text{Sign}(sk, m; \rho)$ , where  $sk$  is a signing key,  $m$  is a message, and  $\rho \in D_\rho$  is a random value, outputs a signature  $\sigma$ .
- $\text{Verify}(pk, m, \sigma)$ , 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  $\text{SIG}$  is *correct* if for every key pair  $(pk, sk)$  output by  $\text{Gen}$ , and for every  $m \in \{0, 1\}^*$  and  $\rho \in D_\rho$ , the corresponding signature  $\sigma = \text{Sign}(sk, m; \rho)$  satisfies

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

We tailor our definition of UF-CMA security to the random oracle model since our analysis will always model the hash function used in  $\text{Sign}$  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  $\text{SIG} = (\text{Gen}, \text{Sign}, \text{Verify})$  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

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

$\text{Gen}(1^\lambda)$	$\text{Sign}(sk, m; \rho)$	$\text{Verify}(pk, m, \sigma)$
$(pk, sk) \leftarrow \text{IGen}(1^\lambda)$	$(a, St) \leftarrow \text{Com}(sk; \rho)$	$(a, e, z) \leftarrow \text{CDF}(\sigma, pk)$
<b>return</b> $(pk, sk)$	$e \leftarrow \text{H}(a, m, pk)$	<b>return</b> $\text{V}(a, e, z, pk) \stackrel{?}{=} 1$
$\text{H}(x)$	$z \leftarrow \text{Resp}(sk, e, St)$	$\wedge \text{H}(a, m, pk) \stackrel{?}{=} e$
$\text{HT}[x] = \perp :$	$\sigma \leftarrow \text{CSF}(a, e, z)$	
$\text{HT}[x] \leftarrow \$_{D_H}$	<b>return</b> $\sigma$	
<b>return</b> $\text{HT}[x]$		

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

is negligible in  $\lambda$ , where  $\text{Exp}_{\text{SIG}}^{\text{UF-CMA}}(\mathcal{A})$  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  $\text{OSign}$  in  $\text{Exp}_{\text{SIG}}^{\text{UF-CMA}}(\mathcal{A})$ .

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  $\text{ID} = (\text{IGen}, \text{Com}, \text{Resp}, \text{V})$  and depicted in Fig. 3, is a three-round protocol defined as follows:

- $\text{IGen}(1^\lambda)$ , where  $\lambda$  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  $\text{IGen}$  defines a *hard-relation*, and that  $pk$  defines the parameters of the scheme including: *randomness space*  $D_\rho$ , *commitment space*  $A$ , *challenge space*  $D_H$  and *response space*  $Z$ .
- Prover invokes a committing algorithm  $\text{Com}$  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  $\text{Resp}$  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  $\text{Resp}$  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  $\text{V}$  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  $\text{V}(a, e, z, pk) = 1$ . We say that ID is *correct* if for every pair  $(pk, sk)$  output by  $\text{IGen}$ , for every  $\rho \in D_\rho$ , and for every transcript  $(a, e, z)$  from an honest execution of the protocol between  $\text{Prover}(sk; \rho)$  and  $\text{Verifier}(pk)$ ,

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

**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  $a$ .

We now define *special honest verifier zero knowledge (special HVZK)* property. The following definition is a slightly stronger variant of [HL10, Definition 6.2.2]. The difference is that, the definition is explicitly augmented with the *auxiliary input*  $\text{aux}$ , following the standard definition of full-fledged (i.e., non-honest verifier) ZK (e.g., see [Gol07, Definition 4.3.10]). We chose to do so to make use of the sequential composition result for computational ZK [Gol07, Lemma 4.3.11]. In Appendix D, we describe other variants of HVZK in detail. Moreover, we explicitly differentiate three flavors of the special HVZK property depending on a level of indistinguishability. Note that  $\epsilon_{\text{HVZK}}$  below is equal to 0 for special *perfect* HVZK.

**Definition 4** (Special  $c/s/p$ -HVZK). Let  $ID = (\text{IGen}, \text{Com}, \text{Resp}, \text{V})$  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  $\text{IGen}$ , every  $e \in D_H$ , and every  $\text{aux} \in \{0, 1\}^*$  it holds that

$$\{\mathcal{M}(pk, e, \text{aux})\}_{pk, e, \text{aux}} \stackrel{c/s/p}{\equiv} \{(\text{Prover}(sk; \rho), \text{Verifier}(pk, e, \text{aux}))\}_{pk, e, \text{aux}}$$

where  $\stackrel{c/s/p}{\equiv}$  means that two probability ensembles are computationally/statistically/perfectly indistinguishable,  $\mathcal{M}(pk, e, \text{aux})$  denotes the output distribution of simulator  $\mathcal{M}$  upon input  $pk, e$  and  $\text{aux}$ , and  $\langle \text{Prover}(sk; \rho), \text{Verifier}(pk, e, \text{aux}) \rangle$  denotes the transcript distribution of an honest execution between  $\text{Prover}$  and  $\text{Verifier}$ , in which we assume that  $\text{Prover}$  receives uniformly sampled randomness  $\rho \in D_\rho$  and  $\text{Verifier}$  sends  $e$  as challenge. We also denote by  $\epsilon_{HVZK}$  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  $\alpha$  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  $\text{IGen}$  generates a hard relation with security parameter  $\lambda$ . The adversary simply runs the HVZK simulator  $\mathcal{M}$  on input  $\mathcal{M}(pk, e_i)$  for different challenge values  $e_1, \dots, e_t \in D_H$ , for a  $t$  polynomial in  $\lambda$ . 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  $\epsilon$  (negligible in  $\lambda$ ) 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  $\lambda$  if  $\alpha < \omega(\log(\lambda))$ .  $\square$

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

- $St$  is a set of  $c$  states  $\{St_1, \dots, St_c\}$ .
- $\text{Resp}$  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  $\lambda$ .

**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  $\text{Resp}$  function of subset revealing ID schemes does not use  $sk$  at all. The above definition includes the 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  $\rho + sk, \rho + 2 \cdot sk, \dots, \rho + 2^\lambda \cdot sk$  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  $ID = (\text{IGen}, \text{Com}, \text{Resp}, \text{V})$  be a canonical identification protocol, and let  $pk$  be a public key output by  $\text{IGen}$ . We call a function  $\text{CSF} : \{0, 1\}^* \rightarrow \{0, 1\}^*$  a *canonical serialization function* if  $\text{CSF}$  is efficiently computable and deterministic, and satisfies the following basic properties: 1) it is *valid*, meaning that there exists a corresponding de-serialization function  $\text{CDF}$  which satisfies the following: for any transcript  $(a, e, z) \in A \times D_H \times Z \cup \{\perp, \perp, \perp\}$  such that  $\text{V}(a, e, z, pk) = 1$ , it holds that  $\text{CDF}(\text{CSF}(a, e, z), pk) = (a, e, z)$ , and 2) it is *sound with respect to invalid responses*, meaning that it returns  $\perp$  upon receiving  $z = \perp$  as input.

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

**Definition 8** (Fiat-Shamir Transform). *The Fiat-Shamir transform*, denoted by  $\mathbf{FS}$ , takes a canonical identification protocol  $ID$  and canonical serialization function  $\text{CSF}$  as input, and outputs a signature scheme  $\mathbf{FS}[ID, \text{CSF}] = (\text{Gen}, \text{Sign}, \text{Verify})$  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  $\mathbf{FS}[ID, \text{CSF}]$  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  $\text{CSF}(a, e, z)$  output  $\sigma := (a, z)$  and letting  $\text{CDF}(\sigma, pk)$  call  $e \leftarrow H(a, m, pk)$  inside to reconstruct the whole transcript. In contrast, if  $ID$  is commitment-recoverable [KLS18], one can instantiate its serialization as follows:  $\text{CSF}(a, e, z)$  outputs  $\sigma := (e, z)$  and  $\text{CDF}(\sigma, pk)$  calls  $a \leftarrow \text{Recover}(pk, e, z)$  inside to reconstruct the transcript. Finally, we could also consider a trivial (de)serialization where  $\text{CSF}$  is an identity function and  $\text{CDF}$  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 [BSS18], 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. We remark that the proof essentially relies on a sequential composition of c-HVZK, which is formally proven in [Gol07, 4.3.4] for general (non-honest verifier) ZK. Note also that the proof does not crucially rely on HVZK being *special*, and indeed, similar hybrid arguments can be made assuming only *non-special* c-HVZK (which is weaker than special c-HVZK). 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  $\rightarrow$  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  $\epsilon_{HVZK}$ .
- ID has  $\alpha$ -bit min-entropy.

*If  $FS := \mathbf{FS}[ID, CSF]$  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*

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

*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<sup>+</sup>12], Dilithium [LDK<sup>+</sup>19] and qTESLA [BAA<sup>+</sup>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<sup>+</sup>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  $(a, e_1, z_1), \dots, (a, e_k, z_k)$ , 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

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 $\Delta$ .
[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 $\tilde{p}k$ as hash input.
[ABF+18]-3	$f_2$	LRB	CBF	EdDSA	Faulty $\tilde{\rho} = \rho + \Delta$ with small $\Delta$ .
[ABF+18]-4	$f_2$	RR	UIS	EdDSA	Fixed $\rho$ .
[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 $\tilde{p}k$ .
[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 $\Delta$ .
[RJH+19]	–	IR	CIS	Dilithium/qTESLA	Instruction skipping during the response computation.

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

$$\begin{aligned} z_1 &= \rho + e_1 \cdot sk \pmod{q} \\ z_2 &= \rho + e_2 \cdot sk \pmod{q} \end{aligned}$$

where the former was computed correctly and the latter is faulty. Then one can easily extract  $sk$  by computing  $(z_1 - z_2)/(e_1 - e_2) \pmod{q}$ , 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  $\rho$  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  $\rho$ , and if  $\rho$  is slightly perturbed by some sufficiently small  $\Delta$ , he can find  $\Delta$  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  $\rho$  and  $\rho + \Delta$ . For example, when attacking the deterministic Schnorr scheme and its variants such as EdDSA, the adversary obtains the following:

$$\begin{aligned} z_1 &= \rho + e_1 \cdot sk \pmod q \\ z_2 &= (\rho + \Delta) + e_2 \cdot sk \pmod q \end{aligned}$$

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

**Randomness Reuse Attack (RR)** This attack forces the randomness  $\rho$  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  $\rho$  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  $\tau/M$  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  $\rho$  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  $\rho$  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.

$\text{HSign}(sk, m, n)$	$\text{Exp}_{\text{HSIG, HE}}^{\text{UF-CMNA}}(\mathcal{A})$	$\text{OHSign}(m, n)$	$\text{HE}(sk', (m', n'))$
$\rho \leftarrow \text{HE}(sk, (m, n))$	$M \leftarrow \emptyset; \text{HET} \leftarrow \emptyset$	$\sigma \leftarrow \text{HSign}(sk, m, n)$	If $\text{HET}[sk', m', n'] = \perp$ :
$\sigma \leftarrow \text{Sign}(sk, m; \rho)$	$(sk, pk) \leftarrow \text{Gen}(1^\lambda)$	$M \leftarrow M \cup \{m\}$	$\text{HET}[sk', m', n'] \leftarrow_{\$} D_\rho$
<b>return</b> $\sigma$	$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\text{OHSign, HE}}(pk)$	<b>return</b> $\sigma$	<b>return</b> $\text{HET}[sk', m', n']$
	$v \leftarrow \text{Verify}(m^*, \sigma^*)$		
	<b>return</b> $(v = 1) \wedge m^* \notin M$		

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

### 4.1 Security of Hedged Signature Schemes

We now consider a simple transformation  $\mathbf{R2H}$ , 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  $\mathbf{R2H}$  and the transformations that appeared in previous works.

- On one hand, a hedged signing algorithm  $\text{HSign}$  takes a *nonce*  $n$  along with a message  $m$ , and derives the randomness  $\rho \in D_\rho$  (of length  $\ell_\rho$  bits) with a *hedged extractor*  $\text{HE}$  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  $\text{HE}$  is in practice a cryptographic hash function, that we will model as a random oracle. Therefore  $\mathbf{R2H}$  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  $\text{HE}$  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  $\rho$  (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  $\text{OHSign}$  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.

$\text{Exp}_{\text{PFS}}^{\text{UF-fCMA}}(\mathcal{A})$	$\text{Exp}_{\text{HFS,HE}}^{\text{UF-fCMNA}}(\mathcal{A})$	$\text{OFaultHSig}(m, n, j, \phi)$	$\text{OFaultSig}(m, j, \phi)$
$M \leftarrow \emptyset; \text{HT} \leftarrow \emptyset; \text{HET} \leftarrow \emptyset$		$f_j := \phi; f_k := \text{Id for } k \neq j$	$f_j := \phi; f_k := \text{Id for } k \neq j$
$(sk, pk) \leftarrow \text{Gen}(1^\lambda)$		<b>FaultHSig</b>	<b>FaultSig</b>
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\text{OFaultHSig,H}}(pk)$		$\rho \leftarrow f_2(\text{HE}(f_1(sk), f_0(m, n)))$	$\rho \leftarrow \$_D \rho; \rho \leftarrow f_2(\rho)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\text{OFaultHSig,H,HE}}(pk)$		$(a, St) \leftarrow f_4(\text{Com}(f_3(sk; \rho)))$	$(a, St) \leftarrow f_4(\text{Com}(f_3(sk; \rho)))$
$v \leftarrow \text{Verify}(m^*, \sigma^*)$		$\hat{a}, \hat{m}, \hat{pk} \leftarrow f_5(a, m, pk)$	$\hat{a}, \hat{m}, \hat{pk} \leftarrow f_5(a, m, pk)$
<b>return</b> $(v = 1) \wedge m^* \notin M$		$e \leftarrow f_6(\text{H}(\hat{a}, \hat{m}, \hat{pk}))$	$e \leftarrow f_6(\text{H}(\hat{a}, \hat{m}, \hat{pk}))$
		$z \leftarrow f_8(\text{Resp}(f_7(sk, e, St)))$	$z \leftarrow f_8(\text{Resp}(f_7(sk, e, St)))$
		$\sigma \leftarrow f_{10}(\text{CSF}(f_9(a, e, z)))$	$\sigma \leftarrow f_{10}(\text{CSF}(f_9(a, e, z)))$
		$M \leftarrow M \cup \{\hat{m}\}$	$M \leftarrow M \cup \{\hat{m}\}$
		<b>return</b> $\sigma$	<b>return</b> $\sigma$

Figure 6: UF-fCMNA and UF-fCMA security experiments and faulty signing oracles for both hedged (HFS) and plain (FS) Fiat–Shamir signature schemes.  $\text{Id}$  stands for the identity function. The function  $\text{H}$  and  $\text{HE}$  (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.

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.

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

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

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

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

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

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

## 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:

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

Using  $\text{flip\_bit}_i(x)$  (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<sup>+</sup>14]. We can model this using `set_biti,b`: when this function is applied to  $\rho$ , we can ensure that the first bit of  $\rho$  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<sup>+</sup>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 `OFaultHSign`, which additionally takes a tampering function  $\phi \in \{\text{set\_bit}_{i,b}, \text{flip\_bit}_i, Id\}$  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 ( $\phi$ ) as well as the target input/output position ( $j$ ) within the signing operation to be faulted. For example, when  $j = 6$ ,  $\phi$  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 `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

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

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

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

is negligible in security parameter  $\lambda$ , where  $\text{Exp}_{\text{HFS}, \text{HE}}^{\text{UF-fCMNA}}(\mathcal{A})$  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

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

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

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

is negligible in security parameter  $\lambda$ , where  $\text{Exp}_{\text{FS}}^{\text{UF-fCMA}}(\mathcal{A})$  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  $sk$  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<sup>+</sup>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  $\sigma^*$  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 OFaultHSig 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.,  $\phi \in \{\text{set\_bit}_{i,b}, \text{flip\_bit}_i\}$ ) or not (i.e.,  $\phi := Id$ ), and so a forgery  $(m^*, \sigma^*)$  output by  $\mathcal{A}$  is *not* considered valid even if  $m^*$  was only queried to OFaultHSig 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 OFaultHSig 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 OFaultHSig does not need to verify, but it may need to be well-formed in some way. Typically we show with a hybrid argument that OFaultHSig can be simulated without use of the private key, in a similar way to OHSig. In order for simulated outputs of OFaultHSig to be indistinguishable from real outputs, simulated signatures must be correctly distributed. In [BDF<sup>+</sup>14, CM09], the security proof shows that the faulty signature is statistically close to a value drawn from the uniform distribution, so OFaultHSig 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 OFaultHSig verifies with an appropriately faulted hash function, and our proofs must take care to maintain these properties when simulating OFaultHSig.

**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

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.

Fault type	ID is subset-revealing	ID not subset-revealing	XEdDSA	Picnic2
$f_0$	✗ Lemma 11		✗	✗
$f_1$	✓ Lemma 4		✓ Corollary 1	✓ Corollary 3
$f_2$	✗ Lemma 13		✗	✓ Lemma 19
$f_3$	✗ Lemma 12		✗	✗ §7
$f_4$	✓ Lemma 10	✗ Lemma 15	✗	✓ Corollary 3
$f_5$	✓ Lemma 7		✓ Corollary 1	
$f_6$	✓ Lemma 8			
$f_7$	✓ Lemma 9	✗ Lemma 14		
$f_8, f_9, f_{10}$	✓ Lemma 6			

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 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<sup>+</sup>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  $\{f_1, f_4, \dots, f_{10}\}$ -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  $\mathbf{R2H}[\mathbf{FS}[\text{ID}, \text{CSF}], \text{HE}]$  signature schemes for fault types  $f_0, \dots, f_{10}$ . 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  $\rightarrow$  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  $\text{FS} := \mathbf{FS}[\text{ID}, \text{CSF}]$  is UF-KOA secure,  $\text{HFS} := \mathbf{R2H}[\text{FS}, \text{HE}]$  is  $\{f_1, f_4, \dots, f_{10}\}$ -UF-fCMNA secure in the random oracle model. Concretely, given  $\{f_1, f_4, \dots, f_{10}\}$ -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*

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

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.  $\square$

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-fCMNA 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  $\rightarrow F \cup \{f_1\}$ -UF-fCMNA). *Suppose the fault adversary  $\mathcal{A}$  does not query the same  $(m, n)$  pair to OFaultHSign more than once. If  $\text{FS} := \mathbf{FS}[\text{ID}, \text{CSF}]$  is  $F$ -UF-fCMA secure, then  $\text{HFS} := \mathbf{R2H}[\text{FS}, \text{HE}]$  is  $F' = F \cup \{f_1\}$ -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*

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

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:  $G_0(\mathcal{A})$  (without boxed code) and  $G_1(\mathcal{A})$  (with boxed code). We assume wlog that  $\mathcal{A}$  does not repeat the same HE query. Notice that  $G_0(\mathcal{A})$  is identical to  $\text{Exp}_{\text{HFS}, \text{HE}}^{\text{UF-fCMNA}}(\mathcal{A})$  in Fig. 6 (on condition that  $\mathcal{A}$  is not allowed to query the same  $(m, n)$  to OFaultHSign), and therefore  $\text{Adv}_{\text{HFS}, \text{HE}}^{\text{UF-fCMNA}}(\mathcal{A}) = \Pr[G_0(\mathcal{A})]$ . Now we obtain the following by simple transformation (where the inequality follows from the fundamental lemma of game-playing [BR06]):

$$\begin{aligned} \text{Adv}_{\text{HFS}, \text{HE}}^{\text{UF-fCMNA}}(\mathcal{A}) &= \Pr[G_0(\mathcal{A})] = \Pr[G_1(\mathcal{A})] + (\Pr[G_0(\mathcal{A})] - \Pr[G_1(\mathcal{A})]) \\ &\leq \Pr[G_1(\mathcal{A})] + \Pr[G_1(\mathcal{A}) \text{ sets bad}]. \end{aligned}$$

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

$$\Pr[G_1(\mathcal{A})] \leq \text{Adv}_{\text{FS}}^{\text{UF-fCMA}}(\mathcal{B}) \quad \text{and} \quad \Pr[G_1(\mathcal{A}) \text{ sets bad}] \leq \text{Adv}_{\text{FS}}^{\text{UF-fCMA}}(\mathcal{B}).$$

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

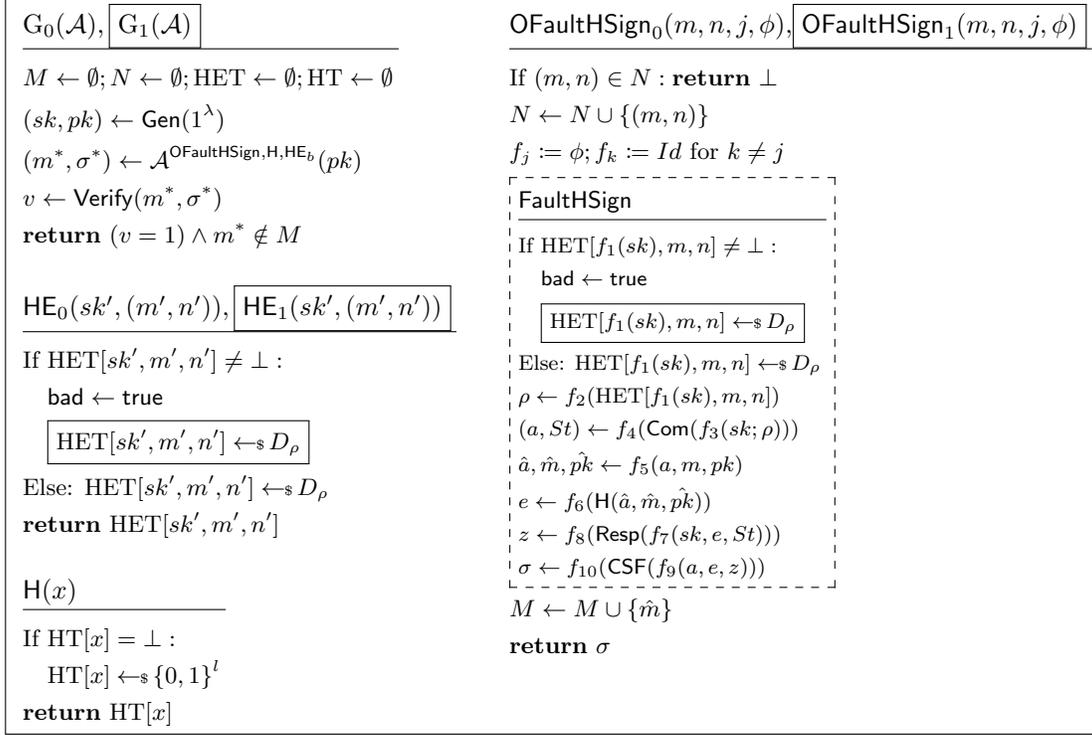


Figure 7: Two identical-until-bad games  $G_0(\mathcal{A})$  and  $G_1(\mathcal{A})$ . The game  $G_1(\mathcal{A})$  executes the boxed code, while  $G_0(\mathcal{A})$  does not. The dashed box indicates that the instructions inside correspond to the actual faulty signing operation.

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

**Simulation of Oracle Queries** The adversary  $\mathcal{B}$  perfectly simulates  $\mathcal{A}$ 's view in  $G_1(\mathcal{A})$  as follows.

- Sim-OFaultHSign<sub>1</sub> simulates OFaultHSign<sub>1</sub> by forwarding faulty signing queries to OFaultSign, the oracle in UF-fCMA game. Notice that this simulation is perfect since both OFaultHSign<sub>1</sub> and OFaultSign use freshly generated randomness for every signing query.
- Sim-HE<sub>1</sub> simulates the HE<sub>1</sub> 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  $G_1(\mathcal{A})$ , i.e.,  $\mathcal{A}$  outputs its forgery  $(m', \sigma')$  that passes the verification and  $m' \notin M = \{\hat{m}_i : i \in [Q_s]\}$ . This means that the reconstructed transcript  $(a', e', z') \leftarrow \text{CDF}(\sigma', pk)$  satisfies

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

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

$$\Pr[G_1(\mathcal{A})] \leq \text{Adv}_{\text{FS}}^{\text{UF-fCMA}}(\mathcal{B}).$$

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.,  $\text{flip\_bit}_i(sk^*)$  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  $G_1(\mathcal{A})$ , i.e.,

$$\Pr[G_1(\mathcal{A}) \text{ sets bad}] \leq \text{Adv}_{\text{FS}}^{\text{UF-fCMA}}(\mathcal{B}).$$

$\mathcal{B}^{\text{OFaultSign,H}}(pk)$	$\text{Sim-OFaultHSign}_1(m, n, j, \phi)$
$M \leftarrow \emptyset; N \leftarrow \emptyset; \text{HET} \leftarrow \emptyset; S \leftarrow \emptyset$	If $(m, n) \in N$ : <b>return</b> $\perp$
$(m', \sigma') \leftarrow \mathcal{A}^{\text{Sim-OFaultHSign}_1, \text{Sim-H}, \text{Sim-HE}_1}(pk)$	$N \leftarrow N \cup \{(m, n)\}$
If $\text{Verify}(m', \sigma') = 1 \wedge m' \notin M$ :	$\sigma \leftarrow \text{OFaultSign}(m, j, \phi)$
<b>return</b> $(m', \sigma')$	$M \leftarrow M \cup \{\hat{m}\}$
Pick some $m^* \notin M$	<b>return</b> $\sigma$
$\rho^* \leftarrow \$_s D_\rho$	<u><math>\text{Sim-HE}_1(sk', (m', n'))</math></u>
For $sk^* \in S$ :	$S \leftarrow S \cup \{sk'\}$
$\sigma^* \leftarrow \text{Sign}(sk^*, m^*; \rho^*)$	$\rho \leftarrow \$_s D_\rho$
If $\text{Verify}(m^*, \sigma^*) = 1$ :	<b>return</b> $\rho$
<b>return</b> $(m^*, \sigma^*)$	
For $i \in [ sk^* ]$ :	<u><math>\text{Sim-H}(x)</math></u>
$\sigma^* \leftarrow \text{Sign}(\text{flip\_bit}_i(sk^*), m^*; \rho^*)$	<b>return</b> $\text{H}(x)$
If $\text{Verify}(m^*, \sigma^*) = 1$ :	
<b>return</b> $(m^*, \sigma^*)$	
<b>return</b> $(\perp, \perp)$	

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  $\text{OFaultHSign}_1$  or  $\text{HE}_1$ , then  $\mathcal{A}$  must have queried  $\text{HE}_1$  with  $(sk^*, (m^*, n^*))$  such that  $sk^* = f_1(sk)$  for some  $f_1 \in \{\text{set\_bit}_{i,b}, \text{flip\_bit}_i, \text{Id}\}$ , and for some  $(m^*, n^*)$ . First, if `bad` is set by  $\text{OFaultHSign}_1$ , it must be due to a previous query to  $\text{HE}_1$  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  $\text{OFaultHSign}$ . Second, we consider the case `bad` is set by  $\text{HE}_1$ . Since we assumed that  $\mathcal{A}$  does not repeat the same HE query, if a query  $(sk', (m', n'))$  to  $\text{HE}_1$  sets `bad` then  $\text{HET}[sk', m', n']$  must have been defined by  $\text{OFaultHSign}_1$ . Here notice that we have  $sk' = f_1(sk)$  for some  $f_1$  due to the definition of  $\text{OFaultHSign}_1$ . Thus if `bad` is set in  $G_1(\mathcal{A})$  then for some  $sk^* \in S$  there exists a faulty function  $f_1 \in \{\text{set\_bit}_{i,b}, \text{flip\_bit}_i, \text{Id}\}$  such that  $sk^* = f_1(sk)$  holds. By checking the validity of signatures generated with each  $sk^* \in S$  and its faulty version  $\text{flip\_bit}_i(sk^*)$ , 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  $\text{Sign}$  and  $\text{Verify}$  due to the above double for-loop operations.  $\square$

**Remarks.** Our reduction above crucially relies upon the assumption that adversaries are not allowed to query the same  $(m, n)$  pair. Without this condition,  $\text{OFaultHSign}$  must return a faulty signature derived from the same randomness  $\rho$  if the same  $(m, n)$  is queried twice, and thus one could not simulate it using  $\text{OFaultSign}$  as an oracle, since  $\text{OFaultSign}$  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  $\rightarrow$  UF-fCMA). *Suppose ID is subset revealing. If  $\text{FS} := \mathbf{FS}[\text{ID}, \text{CSF}]$  is UF-KOA secure, then  $\text{FS}$  is  $\{f_4, \dots, f_{10}\}$ -UF-fCMA secure in the random oracle model. Concretely, given  $\{f_4, \dots, f_{10}\}$ -adversary  $\mathcal{A}$  against FS running in time  $t$ , and making at most  $Q_s$  queries to  $\text{OFaultSign}$ ,  $Q_h$  queries to H, one can construct another adversary  $\mathcal{B}$  against FS such that*

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

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 from Appendix A.1. 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  $H(a, m, pk)$  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 OFaultHSig with probability at most  $\epsilon_{HVZK}$  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  $(m^*, \sigma^*)$  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  $(a^*, e^*, z^*) \leftarrow \text{CDF}(\sigma^*, pk)$  satisfies

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

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

**Hybrid argument if ID is special c-HVZK** As in the plain UF-KOA-to-UF-CMA reduction (Lemma 2), in case special HVZK of ID is only computational the formal security reduction amounts to proving a sequential composition of c-HVZK, which requires  $Q_s + 1$  hybrids. To avoid redundancy we only enumerate the differences with the detailed hybrid argument presented in Appendix A.1, depending on the internal value targeted by a fault attacker for each query  $i$ :

- When  $a_i$  is targeted (i.e.,  $f_4$  or  $f_5$  is used), the probability upper-bound that programming the RO fails is reduced to  $(Q_h + Q_s)/2^{\alpha-1}$  for each query  $i$ .

- When  $e_i$  is targeted (i.e.,  $f_6$  or  $f_7$  is used), the  $\mathcal{D}$  first samples  $e_i \leftarrow_s D_H$  and uses faulty  $\tilde{e}_i$  to answer the  $i$ th query. Then reduction to *special c*-HVZK is made with respect to  $\tilde{e}_i$ .
- When  $St_\iota \in St$  of subset-revealing ID is targeted (i.e.,  $f_4$  or  $f_7$  is used), if  $\iota \in I$  then the  $\mathcal{D}$  replaces  $St_\iota$  inside  $z_i$  with faulty  $\widetilde{St}_\iota$ , where  $I$  is an index set derived from  $e_i$ ; otherwise it uses  $z_i$  as it is.

□

## 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  $\rightarrow$   $\{f_8, f_9, f_{10}\}$ -UF-fCMA). *If  $\text{FS} := \mathbf{FS}[\text{ID}, \text{CSF}]$  is UF-KOA secure, then  $\text{FS}$  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  $\text{FS}$  running in time  $t$ , and making at most  $Q_s$  queries to  $\text{OFaultSign}$ ,  $Q_h$  queries to  $\text{H}$ , one can construct another adversary  $\mathcal{B}$  against  $\text{FS}$  such that*

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

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

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

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  $\text{HT}[\hat{a}_i, \hat{m}_i, \hat{pk}]$  is already set (via  $\mathcal{A}$ 's previous hash or signing queries) and  $\text{HT}[\hat{a}_i, \hat{m}_i, \hat{pk}] \neq e_i$ ,  $\mathcal{B}$  aborts. Otherwise,  $\mathcal{B}$  programs the random oracle so that

$$\text{HT}[\hat{a}_i, \hat{m}_i, \hat{pk}] := e_i.$$

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

$$\sigma_i := \text{CSF}(a_i, e_i, z_i).$$

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  $\alpha$  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  $\alpha$  bits due to  $\alpha$ -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  $\alpha$ -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  $\alpha$  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  $\sigma_i$  from the one returned by the real signing oracle `OFaultHSign` with probability at most  $\epsilon_{\text{HVZK}}$  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 UF-fCMA game with probability at most

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

□

## 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  $e = \text{H}(a, m, pk)$ . 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** (UF-KOA  $\rightarrow$   $\{f_6\}$ -UF-fCMA). *If  $\text{FS} := \text{FS}[\text{ID}, \text{CSF}]$  is UF-KOA secure, then FS is  $\{f_6\}$ -UF-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 `OFaultSign` and  $Q_h$  queries to H, one can construct another adversary  $\mathcal{B}$  against FS such that*

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

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  $\tilde{e}_i := f_6(e_i)$  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  $\text{HT}[\tilde{a}_i, m_i, pk]$  is already set (via  $\mathcal{A}$ 's previous hash or signing queries) and  $\text{HT}[\tilde{a}_i, m_i, pk] \neq e_i$ ,  $\mathcal{B}$  aborts. Otherwise,  $\mathcal{B}$  programs the random oracle so that

$$\text{HT}[\tilde{a}_i, m_i, pk] := e_i.$$

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

$$\tilde{\sigma}_i := \text{CSF}(\tilde{a}_i, \tilde{e}_i, \tilde{z}_i).$$

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  $\text{H}$  has  $\alpha$  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  $\alpha$  bits due to  $\alpha$ -bit min-entropy of  $\text{ID}$ ), and  $\text{HT}$  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  $\text{OFaultHSign}$  with probability at most  $\epsilon_{\text{HVZK}}$  per query, since we used the special c/s/p-HVZK simulator to derive  $(\tilde{a}_i, \tilde{e}_i, \tilde{z}_i)$  and  $\text{CSF}$  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  $\text{UF-fCMA}$  game with probability at most

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

□

**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  $\text{Picnic2}$ ). Computing  $\text{Resp}$  with a malformed challenge may cause  $\sigma$  to leak private information. This is why we required [Definition 3](#) to have the condition that  $\text{Resp}$  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  $\text{CSF}$  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  $\text{set\_bit}$  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  $\text{Resp}$  function. We only guarantee security assuming that the signature scheme is based on a subset revealing identification protocol (see [Definition 5](#)), and  $\text{Resp}$  and  $\text{CSF}$  make sure to rule out invalid challenge and response, respectively. As we will see in the next section,  $\text{Picnic2}$  satisfies these additional properties.

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

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

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

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

4. If random oracle programming is successful,  $\mathcal{B}$  derives from  $e_i$  an index set  $I$  as  $\text{Resp}$  of the real signing oracle would do.
  - If  $\iota \in I$ , then  $\mathcal{B}$  searches the corresponding entry  $St_\iota$  in  $z_i$  and replaces it with  $\tilde{St}_\iota := f_7(St_\iota)$ . Let  $\tilde{z}_i$  be a faulty response modified as above.
  - If  $\iota \notin I$ , then  $\mathcal{B}$  does not modify a response, i.e.  $\tilde{z}_i = z_i$ .

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

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

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  $\alpha$  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  $\alpha$  bits due to  $\alpha$ -bit min-entropy of ID), and  $\text{HT}$  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  $\text{OFaultHSign}$  with probability at most  $\epsilon_{HVZK}$  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 UF-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  $\text{Resp}$  function avoids any algebraic operations involving  $sk$  and  $\rho$ , 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  $\text{Com}(sk; \rho)$ , 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** (UF-KOA  $\rightarrow$   $\{f_4\}$ -UF-fCMA). *Suppose ID is subset revealing. If  $\text{FS} := \mathbf{FS}[\text{ID}, \text{CSF}]$  is UF-KOA secure, then  $\text{FS}$  is  $\{f_4\}$ -UF-fCMA secure in the random oracle model. Concretely, given an  $\{f_4\}$ -adversary  $\mathcal{A}$  against  $\text{FS}$  running in time  $t$ , and making at most  $Q_s$  queries to  $\text{OFaultSign}$  and  $Q_h$  queries to  $\text{H}$ , one can construct another adversary  $\mathcal{B}$  against  $\text{FS}$  such that*

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

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  $\text{CSF}$  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.  $\square$

## 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  $\mathbf{R2H}[\mathbf{FS}[\text{ID}, \text{CSF}], \text{HE}]$  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  $\mathbf{FS}[\text{ID}, \text{CSF}]$  is known to be UF-CMA secure and therefore  $\mathbf{R2H}[\mathbf{FS}[\text{ID}, \text{CSF}], \text{HE}]$  is UF-CMNA secure due to [Lemma 3](#). Our  $\{f_0\}$ -adversary’s strategy is as follows. The adversary first calls  $\text{OFaultHSign}$  with some  $(m, n)$  without fault (i.e.,  $\phi = \text{Id}$ ) to obtain a legitimate signature  $(e, z)$ . Next, the adversary calls  $\text{OFaultHSign}$  with  $\phi = \text{flip\_bit}_i$ ,  $j = 0$  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  $\text{HE}$  and hence the signature is derived from the same  $\rho$  as in the previous query, while the challenge and response are different since  $e' = \text{H}(a, m', pk)$  and  $z = \rho + e' \cdot sk \bmod q$ . Hence we can recover  $sk$  with the SSND attack in [Section 3](#) and break the scheme.  $\square$

**Lemma 12.** *There exist canonical ID schemes such that  $\mathbf{R2H}[\mathbf{FS}[\text{ID}, \text{CSF}], \text{HE}]$  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  $\text{Com}(sk; \rho)$ . When querying  $\text{OFaultHSign}$ , the attacker uses  $\text{set\_bit}$  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  $\text{OFaultHSign}$  aborts.  $\square$

Note that [Lemma 12](#) only applies to ID schemes where  $sk$  is used by the  $\text{Com}$  function. For the Schnorr scheme and other so-called *input delayed protocols* [[CPS+16](#)],  $sk$  is only used by the  $\text{Resp}$  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  $\text{Resp}$  function, but they must use it in the  $\text{Com}$  function.

The sensitivity of ephemeral randomness  $\rho$  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  $\rho$  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  $\mathbf{R2H}[\mathbf{FS}[\text{ID}, \text{CSF}], \text{HE}]$  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  $\rho$  in Schnorr, as its  $St$  contains the randomness  $\rho$ . If  $f_2$  or  $f_3$  is `set_bit` and always targets at the most significant bit of  $\rho$  to fix its value, the attacker can introduce 1-bit bias in  $\rho$ .

Relative to an oracle for the HNP, the Schnorr scheme with unbiased  $\rho$  remains secure, however, the scheme with biased  $\rho$  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.  $\square$

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  $\rho$ . For fault types  $f_7$  and  $f_4$ , we have shown that  $\mathbf{R2H}[\mathbf{FS}[\text{ID}, \text{CSF}], \text{HE}]$  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  $\mathbf{R2H}[\mathbf{FS}[\text{ID}, \text{CSF}], \text{HE}]$  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  $H(a, m, pk) = e$ , so the adversary can learn 1-bit of  $\rho$  by checking whether  $H(a' + [2^{\ell_\rho - 1}]G, m, pk) = e$  or  $H(a' - [2^{\ell_\rho - 1}]G, m, pk) = e$  holds, where  $\ell_\rho$  is the bit length of  $\rho$ . Since we now have the most significant bit of  $\rho$ , we use the same argument as in Lemma 13 to show the scheme is vulnerable to fault attacks.  $\square$

**Lemma 15.** *There exist non-subset-revealing canonical ID schemes such that  $\mathbf{R2H}[\mathbf{FS}[\text{ID}, \text{CSF}], \text{HE}]$  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  $\rho$ , 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.  $\square$

## 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  $\text{Schnorr} := \mathbf{FS}[\text{ID-Schnorr}, \text{CSF}]$  and  $\text{XEdDSA} := \mathbf{R2H}[\text{Schnorr}, \text{HE}]$ , 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  $\epsilon_{\text{HVZK}} = 0$ ) and has  $2\lambda$  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  $2\lambda$  bits (to provide  $\lambda$ -bit security against key recovery attacks using generic algorithms for the discrete logarithm problem [[Sho97](#), [Nec94](#)]).  $\square$

**UF-KOA Security** Let  $\text{Adv}_{\text{Schnorr}}^{\text{UF-KOA}}(\mathcal{A})$  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  $\text{Adv}_{\text{Schnorr}}^{\text{UF-KOA}}(\mathcal{A})$ . 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  $\epsilon_{\text{HVZK}} = 0$ ) and has  $\alpha = 2\lambda$  bits of min-entropy. Then we can apply [Theorem 1](#), to obtain

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

$\square$

**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  $\text{Picnic2} := \mathbf{FS}[\text{ID-Picnic2}, \text{CSF}]$  and  $\text{HS-Picnic2} := \mathbf{R2H}[\text{Picnic2}, \text{HE}]$  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](#)].

We also consider a randomized variant of Picnic2. Let  $\text{KKW}_\tau$  denote the KKW protocol ([Figure 2](#) of [[KKW18](#)]), changed to include a  $\lambda$ -bit random value as input to the commitment  $h'_j$  (on line 4 of [Fig. 11](#)), and  $\text{ID-Picnic2}_\tau$  denote ID-Picnic2 instantiated with  $\text{KKW}_\tau$ . We then denote the non-hedged signature scheme by  $\text{Picnic2}_\tau := \mathbf{FS}[\text{ID-Picnic2}_\tau, \text{CSF}]$  and the hedged one with  $\text{HS-Picnic2}_\tau := \mathbf{R2H}[\text{Picnic2}_\tau, \text{HE}]$ .  $\text{Picnic2}_\tau$  has a simpler HVZK proof, but the drawback is that  $\tau$  additional random values must be included in the proof, making signatures larger.

### 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. Except for the HVZK property these apply equally to Picnic2 and  $\text{Picnic2}_\tau$ .

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

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

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  $2\lambda$ -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  $\lambda$ , 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  $2\lambda$ -bit outputs). In a fully randomized version of ID-Picnic2, the inputs to the commitments contain randomly chosen seeds (each of length  $\lambda$  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 ( $\lambda$  bits), an initial  $\lambda$ -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  $2\lambda$ -bit commitments preserves the entropy of the  $\lambda$ -bit seeds.  $\square$

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  $2\lambda$ -bit outputs, and key generation function Gen is  $(t, \epsilon_{OW})$ -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

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

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  $(M, n, \tau)$  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

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

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

**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, Theorem 2.2], it is shown that ID-Picnic2<sub>r</sub> is HVZK and in fact, it also satisfies Definition 4, but in the specified version no additional randomness is included (in order to make the opening information smaller). Because of this optimization, the proof for ID-Picnic2 is somewhat different. Also, as discussed in Appendix D, the definition of c-HVZK is slightly different than Definition 4, instead we use special multi-cHVZK (as defined in [GHHM20, Definition 2], see discussion in Appendix D), which is sufficient for the results of the paper.

**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, key generation is a  $(t, \epsilon_{OWF})$ -secure one-way function, and the PRG is  $(t, \epsilon_{PRG})$ -secure. Simulated transcripts are computationally indistinguishable from real transcripts, and all polynomial-time distinguishing algorithms succeed with probability at most

$$\epsilon_{HVZK} \leq Q_s \left( (n+2)\tau \cdot \epsilon_{PRG} + \frac{M}{2^{2\lambda}} + \epsilon_{OWF} \right).$$

where  $q_0$  and  $q_2$  are the number of queries to  $H_0$  and  $H_2$ ,  $Q_s$  is the number of transcripts that are simulated,  $\lambda$  is the security parameter, and  $(M, n, \tau)$  are parameters of the scheme.

*Proof.* Here we bound  $\epsilon_{HVZK}$  for one transcript, and note that when  $Q_s$  transcripts are generated, changes for each game are applied to all of the transcripts and we have

$$\epsilon_{HVZK} \text{ for } Q_s \text{ transcripts} \leq Q_s (\epsilon_{HVZK} \text{ for 1 transcript}).$$

Let  $\Pi$  denote the MPC protocol, and  $\text{Sim}_\Pi$  denote a simulator for  $\Pi$ . Recall that  $\Pi$  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  $\{\text{seed}_{j,i}\}_{i=1}^n$  for all  $j \in \mathcal{C}$ , rather than using the PRG. We have  $G_2 - G_1 = n\tau \cdot \epsilon_{PRG}$  (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  $\text{seed}_{j,p_j}$  with the PRG. Recall that there are a total of  $\tau$  unopened parties. Since the seeds are random values committed to as  $\text{com}_{j,p_j} = H_0(\text{seed}_{j,p_j})$ , this goes undetected except with probability  $\frac{q_0\tau}{2^\lambda}$ , where  $q_0$  is the number of queries to  $H_0$ . Therefore  $G_2 - G_3 \leq \tau \cdot \epsilon_{PRG} + \frac{q_0\tau}{2^\lambda}$ .

**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  $(\{\hat{z}_{j,\alpha}\}, \text{msgs}_{j,1}, \dots, \text{msgs}_{j,n})$  and since  $j \notin \mathcal{C}$ ,  $\text{seed}^*$  is revealed and  $\text{msgs}_{j,i}$  may be computed. The other value has the form  $\{\hat{z}_{j,\alpha}\} = sk \oplus \lambda_{sk}$  where  $\lambda_{sk}$  is computed from the first bits of each party's random tape (also all revealed).

In Game 4 we additionally change the prover to compute the values  $\lambda_{sk}$  and  $\text{msgs}_{j,i}$  from  $\text{seed}^*$  for all  $j$ . Then when the adversary makes an  $H_2$ -query of the form  $H_2(z', \text{msgs}'_{j,1}, \dots, \text{msgs}'_{j,N})$ , the prover checks whether  $\text{msgs}_{j,i} = \text{msgs}'_{j,i}$  (for all  $i, j$ ) and if so computes  $sk' = z' \oplus \lambda_{sk}$  and aborts if  $pk = \text{OWF}(sk')$ . Denote this probability of aborting by  $\Pr[\text{Abort}]$ . Therefore  $G_4 - G_3 = \Pr_4[\text{Abort}] + M/2^{2\kappa}$ , where the second term captures the probability that the random choice of  $h'_j$  was already output by  $H_2$ .

**Game 5** We change the prover, for each  $j \in \mathcal{C}$ , to compute  $\text{com}_{j,p_j}$  as a commitment to a random value. Since  $\text{seed}_{j,p_j}$  is no longer used (as of Game 4), computing  $\text{com}_{j,p_j}$  in this way is consistent with the rest of the transcript, and so  $G_4 = G_5$ .

**Game 6** We change the prover, for  $j \in \mathcal{C}$ , to use  $\text{Sim}_\Pi$  to generate the views of the parties (excluding  $p_j$ ), in an execution of  $\Pi$  when evaluating  $C$  with output 1. This results in values  $\{\text{state}_{j,i}\}_{i \neq p_j}$ , masked input-wire values  $\{\hat{z}_{j,\alpha}\}$ , and  $\text{msg}_{p_j}$ . From the respective views,  $\{\text{msg}_{j,i}\}_{i \neq p_j}$  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  $\Pi$  are distributed exactly as real transcripts, under the semi-honest security of  $\Pi$ , therefore Games 5 and 6 produce identically distributed outputs. The security of  $\Pi$  (proven in Section 6.1 of [Pic19b]), holds assuming the PRG used to generate the random tapes is secure, therefore  $G_6 - G_5 = \tau \cdot \epsilon_{PRG}$ .

In Game 6 the witness is no longer used, and the simulated transcripts are computationally indistinguishable from real transcripts, provided event **Abort** has not occurred. We now argue that  $\Pr[\text{Abort}] \leq \epsilon_{OWF}$ . In Game 6, since the witness is no longer used, and the simulation works for any choice of  $pk$ , we can construct an inversion algorithm for the OWF, that on input  $pk$ , outputs  $sk$  with probability  $\Pr[\text{Abort}]$ , since the **Abort** event happens when  $pk = \text{OWF}(sk')$ .

□

## 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  $f_4, \dots, f_{10}$ . Here we apply those results to the HS-Picnic2 signature scheme. Recall that  $\text{HS-Picnic2}_r$  denotes  $\mathbf{R2H}[\mathbf{FS}[\text{ID-Picnic2}_r, \text{CSF}], \text{HE}]$ . We can directly apply the results of Section 5 to  $\text{HS-Picnic2}_r$ . We do this first, then explain in the following section that the same results also hold for  $\text{HS-Picnic2}$ .

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

*Proof.* Recall that by Corollary 2,  $\text{Picnic2}_r$  is UF-KOA secure with

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

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

We can apply Theorem 1, to obtain

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

where  $\epsilon_{HVZK}$ , given in Lemma 18 and the remark following it, is

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

□

**Discussion** Our model places the restriction on  $\text{OFaultHSign}$  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  $\text{HS-Picnic2}$ ,  $(\mathcal{C}, \mathcal{P})$ , 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

In the previous section, the general results of Section 5 show that Picnic2 is secure for fault types  $f_1$  and  $f_4, \dots, f_{10}$ , with the caveat that certain commitments in the KKW protocol are randomized. The specified version of Picnic2 does not use this extra randomization, and only satisfies a slightly weaker definition of c-HVZK than Definition 4, called special multi-cHVZK (as described in Appendix D). However, special multi-cHVZK is essentially equivalent in the context of secure signatures, and the generic results of Section 5 hold for it as well. In the bound of Eq. (2) we must replace  $Q_s \cdot \epsilon_{HVZK}$  with the bound on  $\epsilon_{HVZK}$  given in Lemma 18 (the latter already includes the term  $Q_s$ ).

For the remaining fault types, we’ve shown that security against fault types  $f_2$  and  $f_3$  does 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  $\rho$ , the output of the hedged extractor. Intuitively, HS-Picnic2 is  $\{f_2\}$ -UF-fCMNA secure since  $\rho$  is not used directly,  $\rho$  is the list of  $\text{seed}_j^*$  values, which are used as input to a PRG when deriving the  $\text{seed}_{i,j}^*$  values. Applying a 1-bit fault to a  $\text{seed}_j^*$  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.  $\text{Adv}_{\text{HS-Picnic2}}^{\text{UF-fCMNA}}(\mathcal{A})$  is the same as given in Corollary 3, except that  $\alpha$  is reduced by 1.*

**Fault type  $f_3$**  Recall that  $f_3$  faults are applied to  $\text{Com}(f_3(sk; \rho))$ . An  $f_3$  fault that changes  $\rho$  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  $\lambda$ -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  $\text{tape} = \text{PRG}(\text{seed}, t)$ . 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,  $\text{tape}' = \text{PRG}(\text{seed}, t')$  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<sup>+</sup>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,  $C = H(\text{seed})$  where  $\text{seed}$  is a 128-bit value, with two 64-bit halves,  $\text{seed} = \text{seed}_{\text{low}} \parallel \text{seed}_{\text{high}}$ . 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  $C' = H(0 \parallel \text{seed}_{\text{high}})$ , he can find  $\text{seed}_{\text{high}}$  by brute force with  $2^{64}$  hash computations, then find  $\text{seed}_{\text{low}}$  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  $\mathbf{R2H}[\mathbf{FS}[\text{ID}, \text{CSF}], \text{HE}]$  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  $\text{OFaultHSign}$ . 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  $\text{OFaultHSign}$  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  $\text{OFaultHSign}$ . 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  $\text{OFaultHSign}(m, n)$  with no faults, then calling  $\text{OFaultHSign}(m', n, 1, \text{flip\_bit})$  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  $\rho$  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<sup>+</sup>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.** We thank the authors of [GHHM20] for communicating the multiple transcripts issue to us (explained in Appendix D), after which we revised Definition 4 and Lemma 18. 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<sup>+</sup>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<sup>+</sup>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<sup>+</sup>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<sup>+</sup>14] Diego F. Aranha, Pierre-Alain Fouque, Benoît Gérard, Jean-Gabriel Kammerer, Mehdi Tibouchi, and Jean-Christophe Zavalowicz. 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<sup>+</sup>19] Nina Bindel, Sedat Akleyek, 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<sup>+</sup>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.

- [BSS18] 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<sup>+</sup>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<sup>+</sup>14] Gilles Barthe, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Mehdi Tibouchi, and Jean-Christophe Zepalawicz. 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<sup>+</sup>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 Brenzel 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.
- [BS17] Dan Boneh and Victor Shoup. A graduate course in applied cryptography, 2017. Available online <https://crypto.stanford.edu/~dabo/cryptobook/>.
- [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<sup>+</sup>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<sup>+</sup>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<sup>+</sup>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<sup>+</sup>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<sup>+</sup>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<sup>+</sup>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<sup>+</sup>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] Eiichi 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.
- [GHHM20] Alex B. Grilo, Kathrin Hövelmanns, Andreas Hülsing, and Christian Majenz. Tight adaptive reprogramming in the QROM. Cryptology ePrint Archive, Report 2020/1361, 2020. <https://eprint.iacr.org/2020/1361>.
- [GLM<sup>+</sup>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<sup>+</sup>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.
- [Kat10] Jonathan Katz. *Digital signatures*. Springer Science & Business Media, 2010.
- [KDK<sup>+</sup>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<sup>+</sup>19] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrede 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’Raihi, 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<sup>+</sup>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 VEdDSA 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<sup>+</sup>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<sup>+</sup>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. Romailier 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<sup>+</sup>19] Greg Zaverucha, Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, Jonathan Katz, Xiao Wang, and Vladimir 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  $\rightarrow$  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  $\epsilon_{HVZK}$ .
- ID has  $\alpha$ -bit min-entropy.

*If  $\text{FS} := \text{FS}[\text{ID}, \text{CSF}]$  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*

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

*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  $H(a, m, pk)$  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  $H'(a, m, pk)$ .

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

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

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  $\mathcal{A}$  a serialized transcript

$$\sigma_i := \text{CSF}(a_i, e_i, z_i).$$

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  $\alpha$  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  $\alpha$  bits due to  $\alpha$ -bit min-entropy of ID), and  $\text{HT}$  has at most  $Q_h + Q_s$  existing entries.
- $\mathcal{A}$  after the fourth step distinguishes  $\sigma_i$  from the one returned by the real signing oracle  $\text{OSign}$  with probability at most  $\epsilon_{HVZK}$  per each query, since we used the special c/s/p-HVZK simulator to derive  $(a_i, e_i, z_i)$  and  $\text{CSF}$  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-CMA game with probability at most

$$\frac{(Q_h + Q_s)Q_s}{2^\alpha} + Q_s \cdot \epsilon_{HVZK}. \quad (3)$$

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

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

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

**Detailed hybrid argument if ID is special c-HVZK** In more detail, the probability bound (3) can be derived via  $Q_s + 1$  hybrids in case special HVZK of ID is only computational. Below we closely follow the sequential composition of ZK proven in [Gol07, 4.3.4]. For the  $j$ -th hybrid  $G_j(\mathcal{A})$  for  $j \in [0, Q_s]$ , we construct another adversary  $\mathcal{D}$  who answers first  $j$  signing queries using an honest execution of ID, and then replies the remaining  $Q_s - j$  queries by invoking a (special) c-HVZK simulator. Concretely, for the query  $i \in [1, j]$  with  $j \neq 0$ , the  $\mathcal{D}$  samples  $e_i \leftarrow D_H$ , runs  $(a_i, e_i, z_i) \leftarrow (\text{Prover}(sk), \text{Verifier}(pk, e_i, \text{aux}^{(i-1)}))$ , and programs the RO such that  $\text{HT}[a_i, m_i, pk] := e_i$  (if  $\text{HT}[a_i, m_i, pk]$  is already set then  $\mathcal{D}$  aborts), where  $\text{aux}^{(0)}$  is an empty string and  $\text{aux}^{(i-1)}$  consists of a sequence of previous transcripts:  $(a_1, e_1, z_1), \dots, (a_{i-1}, e_{i-1}, z_{i-1})$ . For the query  $i \in [j+1, Q_s]$  with  $j \neq Q_s$ , the  $\mathcal{D}$  instead answers by invoking  $(a_i, e_i, z_i) \leftarrow \mathcal{M}(pk, e_i, \text{aux}^{(i-1)})$ . Let  $\text{Pr}[G_j(\mathcal{A})]$  be a probability that  $\mathcal{A}$  succeeds in submitting a valid forgery in  $G_j(\mathcal{A})$ . Clearly, by construction

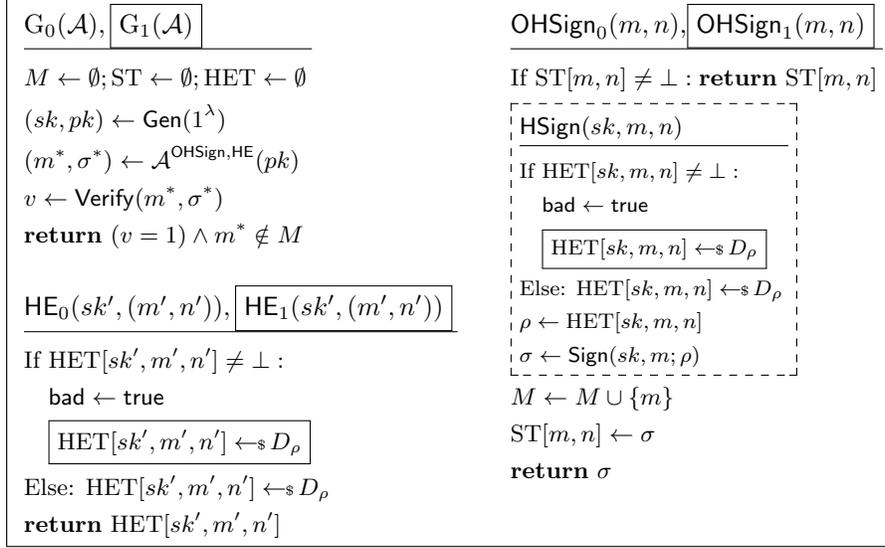


Figure 9: Two identical-until-bad games  $G_0(\mathcal{A})$  and  $G_1(\mathcal{A})$ . The game  $G_1(\mathcal{A})$  executes the boxed code, while  $G_0(\mathcal{A})$  does not. The dashed box indicates that the instructions inside correspond to the actual hedged signing operation.

we have  $|\text{Adv}_{\text{FS}}^{\text{UF-CMA}}(\mathcal{A}) - \Pr[G_{Q_s}(\mathcal{A})]| \leq (Q_h + Q_s)Q_s/2^\alpha$  and  $\Pr[G_0(\mathcal{A})] = \text{Adv}_{\text{FS}}^{\text{UF-KOA}}(\mathcal{D})$ . (Note that  $\mathcal{D}$  in  $G_0(\mathcal{A})$  is now identical to the resulting reduction  $\mathcal{B}$  we described above, by letting  $\mathcal{D}$  output the forgery submitted by  $\mathcal{A}$ .)

Now suppose for some  $j \in [1, Q_s]$ , the success probabilities of  $\mathcal{A}$  in  $j$ th and  $j - 1$ th hybrids are non-negligibly far apart, i.e., there exists some  $\text{aux}^{(j-1)}$  such that  $|\Pr[G_j(\mathcal{A})] - \Pr[G_{j-1}(\mathcal{A})]|$  is non-negligible. (Notice that  $\text{aux}^{(j-1)}$  in  $G_j(\mathcal{A})$  and  $G_{j-1}(\mathcal{A})$  are identical. ) Then in  $G_j(\mathcal{A})$  the  $j$ th query is answered with  $(a_j, e_j, z_j) \leftarrow (\text{Prover}(sk), \text{Verifier}(pk, e_j, \text{aux}^{(j-1)}))$  for randomly chosen  $e_j \in D_H$ , whereas in  $G_{j-1}(\mathcal{A})$ , the  $\mathcal{D}$  instead uses  $(a_j, e_j, z_j) \leftarrow \mathcal{M}(pk, e_j, \text{aux}^{(j-1)})$ . If two hybrids are non-negligibly far apart for some  $e_j \in D_H$  and for some  $\text{aux}^{(j-1)}$ , then it means that the  $\mathcal{D}$ , upon receiving  $(pk, e_j, \text{aux}^{(j-1)}, (a_j, e_j, z_j))$ , serves as a distinguisher of special  $c$ -HVZK game by letting  $\mathcal{D}$  return 1 iff  $\mathcal{A}$  submits a valid forgery. Putting together, we obtain  $|\Pr[G_{Q_s}(\mathcal{A})] - \Pr[G_0(\mathcal{A})]| \leq Q_s \cdot \epsilon_{\text{HVZK}}$ .  $\square$

## A.2 Proof of Lemma 3

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

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

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

*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 UUF-CMA to UF-CMA for deterministic signature schemes [BPS16]. Let us consider two *identical-until-bad* games described in Fig. 9:  $G_0(\mathcal{A})$  (without boxed code) and  $G_1(\mathcal{A})$  (with boxed code). We assume wlog that  $\mathcal{A}$  does not repeat the same HE query. Notice that  $G_0(\mathcal{A})$  is identical to  $\text{Exp}_{\text{HSIG, HE}}^{\text{UF-CMNA}}(\mathcal{A})$  in Fig. 6 and therefore  $\text{Adv}_{\text{HSIG, HE}}^{\text{UF-CMNA}}(\mathcal{A}) = \Pr[G_0(\mathcal{A})]$ . Now we obtain the following by simple

$\mathcal{B}^{\text{OSign}}(pk)$	$\text{Sim-OHSign}_1(m, n)$
$M \leftarrow \emptyset; \text{ST} \leftarrow \emptyset; \text{HET} \leftarrow \emptyset; S \leftarrow \emptyset$	If $\text{ST}[m, n] \neq \perp$ : <b>return</b> $\text{ST}[m, n]$
$(m', \sigma') \leftarrow \mathcal{A}^{\text{Sim-OHSign}_1, \text{Sim-HE}_1}(pk)$	$\sigma \leftarrow \text{OSign}(m)$
If $\text{Verify}(m', \sigma') = 1 \wedge m' \notin M$ :	$M \leftarrow M \cup \{m\}$
<b>return</b> $(m', \sigma')$	$\text{ST}[m, n] \leftarrow \sigma$
Pick some $m^* \notin M$	<b>return</b> $\sigma$
$\rho^* \leftarrow_s D_\rho$	
For $sk^* \in S$ :	<u><math>\text{Sim-HE}_1(sk', (m', n'))</math></u>
$\sigma^* \leftarrow \text{Sign}(sk^*, m^*; \rho^*)$	$S \leftarrow S \cup \{sk'\}$
If $\text{Verify}(m^*, \sigma^*) = 1$ :	$\rho \leftarrow_s D_\rho$
<b>return</b> $(m^*, \sigma^*)$	<b>return</b> $\rho$
<b>return</b> $(\perp, \perp)$	

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

transformation:

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

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

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

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

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

If the adversary  $\mathcal{A}$  wins the game  $\text{G}_1(\mathcal{A})$ , this implies that  $\mathcal{B}$  receives a valid forgery  $(m', \sigma')$  from  $\mathcal{A}$  so that  $\mathcal{B}$  can win UF-CMA game, i.e.,  $\Pr[\text{G}_1(\mathcal{A})] \leq \mathbf{Adv}_{\text{SIG}}^{\text{UF-CMA}}(\mathcal{B})$ . 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  $\text{G}_1(\mathcal{A})$ , i.e.,  $\Pr[\text{G}_1(\mathcal{A}) \text{ sets bad}] \leq \mathbf{Adv}_{\text{SIG}}^{\text{UF-CMA}}(\mathcal{B})$ .

Here we show that if bad is set either by  $\text{OHSign}_1$  or  $\text{HE}_1$ , then  $\mathcal{A}$  must have queried  $\text{HE}_1$  with  $(sk^*, (m^*, n^*))$  for some  $(m^*, n^*)$  such that  $sk^* = sk$ . First, if bad is set by  $\text{OHSign}_1$ , it must be due to a previous query to  $\text{HE}_1$  with  $(sk', (m', n'))$  such that  $sk' = sk$  holds, since  $\text{OHSign}$ , without invoking  $\text{HSign}$ , returns previously generated signature  $\sigma$  stored in  $\text{ST}$  whenever the same  $(m, n)$  pair is queried. Second, we consider the case bad is set by  $\text{HE}_1$ . Since we assumed that  $\mathcal{A}$  does not repeat the same HE query, if a query  $(sk', (m', n'))$  to  $\text{HE}_1$  sets bad then  $\text{HET}[sk', m', n']$  must have been defined by  $\text{OHSign}_1$ . Here notice that we have  $sk' = sk$  due to the definition of  $\text{OHSign}_1$ . Thus if bad is set in  $\text{G}_1(\mathcal{A})$  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  $\text{Sign}$  and  $\text{Verify}$  due to the above for-loop operations.  $\square$

### A.3 Proof of Lemma 6

**Lemma 6** (UF-KOA  $\rightarrow \{f_8, f_9, f_{10}\}$ -UF-fCMA). *If  $\text{FS} := \mathbf{FS}[\text{ID}, \text{CSF}]$  is UF-KOA secure, then  $\text{FS}$  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  $\text{FS}$  run-*

ning in time  $t$ , and making at most  $Q_s$  queries to  $\text{OFaultSign}$ ,  $Q_h$  queries to  $\text{H}$ , one can construct another adversary  $\mathcal{B}$  against  $\text{FS}$  such that

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

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  $\text{UF-fCMA}$  experiment (resp.  $\text{UF-KOA}$  experiment) by  $\text{H}$  and  $\text{HT}$  (resp.  $\text{H}'$  and  $\text{HT}'$ ).

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

**Simulation of Random Oracle Queries** Upon receiving a random oracle query  $\text{H}(a, m, pk)$  from  $\mathcal{A}$ ,  $\mathcal{B}$  forwards the input  $(a, m, pk)$  to its own random oracle ( $\text{H}'$  from the  $\text{UF-KOA}$  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  $\text{ID}$  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  $\text{HT}[a_i, m_i, pk]$  is already set (via  $\mathcal{A}$ 's previous hash or signing queries) and  $\text{HT}[a_i, m_i, pk] \neq e_i$ ,  $\mathcal{B}$  aborts. Otherwise,  $\mathcal{B}$  programs the random oracle so that

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

Notice that programming makes  $\text{H}$ 's output inconsistent with that of the external random oracle  $\text{H}'$ , 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  $\phi_i$ :

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

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  $\text{H}$  has  $\alpha$  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  $\alpha$  bits due to  $\alpha$ -bit min-entropy of  $\text{ID}$ ), and  $\text{HT}$  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  $\text{OFaultHSig}$  with probability at most  $\epsilon_{\text{HVZK}}$  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  $\text{CSF}$  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  $\text{UF-fCMA}$  game with probability at most

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

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

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

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

## 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  $2\lambda$ -bit digests, and  $H''$  has  $4\lambda$ -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  $\rho$ . In XEdDSA (Algorithm 2), a single secret key is used for both computing  $\rho$  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  $\sigma$

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

---

### 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  $\sigma$

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

---

### 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  $\sigma$

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

Picnic2 Sign( $m; \rho$ )

Values  $(M, n, \tau)$  are parameters of the protocol.

Parse  $\rho$  as  $(\text{seed}_1^*, \dots, \text{seed}_M^*)$ , where each value is  $\lambda$  bits long.

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

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

Let  $a := (h_1, h'_1, \dots, h_M, h'_M)$ .

Let  $St := \{\text{seed}_j^*, \{\text{com}_{j,i}\}, \{\text{state}_{j,i}\}, \{h_j\}, \{h'_j\}$  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  $\tau$ , and  $\mathcal{P}$  is a list  $\{p_j\}_{j \in \mathcal{C}}$  with  $p_j \in [n]$ . Let  $e := (\mathcal{C}, \mathcal{P})$ .

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

**Serialize** Output  $(e, z)$ .

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

## 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  $\{\text{seed}_j^*\}_{j=1}^M$ , from a single root seed, using a tree construction. Similarly, the per-party, per-instance seeds  $\{\text{seed}_{i,j}\}$  are derived from  $\text{seed}_j^*$  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  $sk \parallel m \parallel pk \parallel n$  where  $n$  is a  $2\lambda$ -bit random value (acting as the nonce in the notation of Section 4).

**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  $\lambda$ -bit key and block size,  $p$  is a random  $\lambda$ -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 Verification Verify( $\sigma, m$ )

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

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

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

## D Technicalities in the Definition of c-HVZK

Definitions for computational honest-verifier zero-knowledge (c-HVZK) vary in the literature. Here we discuss some differences that have an impact on the analysis of Picnic in [Section 7](#).

**Does the distinguisher know  $sk$ ?** In the literature there are two main ways to define computational HVZK for an ID scheme (in the context of analyzing the security of the associated signature scheme). The first is similar to [Definition 4](#), but without auxiliary input. This style of definition states the two distributions which must be computationally indistinguishable, and is not always clear about whether the distributions include  $sk$  along with the transcript. In the context of a reduction, we need a game-based security definition, so that the inputs and outputs of a c-HVZK distinguisher algorithm are explicit. For example (as in [\[Kat10\]](#) following [Definition 8.3](#)), we require that the following is negligible for any PPT adversary  $\mathcal{A}$ :

$$\left| \Pr[(pk, sk) \leftarrow \text{Gen}(1^\kappa) : \mathcal{A}^{\text{Trans}_{sk, pk}(\cdot)}(pk) = 1] - \Pr[(pk, sk) \leftarrow \text{Gen}(1^\kappa) : \mathcal{A}^{\text{Sim}_{pk}(\cdot)}(pk) = 1] \right| \quad (4)$$

One can also find some game-based definitions in the literature where adversary  $\mathcal{A}$  submits  $(pk, sk)$  to the c-HVZK challenger [\[BS17, Attack Game 20.4\]](#), and thus knows  $sk$ , while other definitions initialize  $\mathcal{A}$  with a randomly chosen  $pk$  (as in the example above from [\[Kat10\]](#)).

In the context of signature schemes, it's natural for  $\mathcal{A}$  to only have  $pk$ , since it's only important that simulated transcripts be indistinguishable when  $sk$  is unknown. However, as noted in [\[Kat10\]](#), going from indistinguishability of transcripts to [Eq. \(4\)](#) requires that the distributions be indistinguishable even when  $sk$  is included with the transcript. To see why, suppose that a protocol is not c-HVZK when  $sk$  is known to  $\mathcal{A}$ . Now since  $pk$  is computed as a function of  $sk$  and is given to  $\mathcal{A}$ , without some additional assumption on  $\text{Gen}$ , it may be that  $pk$  leaks enough about  $sk$  to allow  $\mathcal{A}$  to distinguish the two distributions.

For a specific ID scheme, one can sometimes directly prove it satisfies the game-based definition [Eq. \(4\)](#) under the additional assumption that  $\text{Gen}$  produces the pair  $(pk, sk)$  as  $pk = \text{OWF}(sk)$ , and  $\text{OWF}$  is a secure one-way function.

Until now, this difference seems to have been mostly been esoteric, as the majority of ID schemes are  $p/s$ -HVZK, and this holds when  $sk$  is known to  $\mathcal{A}$  (since  $\mathcal{A}$  is computationally unbounded, whether  $sk$  is explicitly provided is moot, as it can be recovered from  $pk$ ). However, by derandomizing ID schemes we can naturally construct a protocol that is c-HVZK only when  $sk$  not known to  $\mathcal{A}$ . As an example, take the Schnorr protocol where  $r = H(sk||t)$  where  $t$  is a public timestamp (or nonce) unique per run of the protocol and  $H$  is a RO. Simulated transcripts will not have  $r$  of this form, and  $r$  can be computed given  $sk$ , so it is essential that  $sk$  remain hidden from  $\mathcal{A}$ . The optimization in the KKW proof protocol to avoid randomizing commitments (used by Picnic) is similar in nature; the proof contains a hash digest computed

from  $sk$  and random values that are made public as part of the proof. The derandomization described above is also one difference separating EdDSA from Schnorr; therefore the ID scheme associated with EdDSA is not p/s-HVZK, but is c-HVZK for definitions that ensure  $sk$  is hidden from the adversary.

**One or multiple transcripts?** Again one can find definitions of c-HVZK in the literature with a single transcript [BS17], or multiple transcripts [Kat10]. With Definition 4, by adding auxiliary input, sequential composition of c-HVZK is possible, as implicitly shown in Appendix A.1.

When c-HVZK does not hold when  $sk$  is known to  $\mathcal{A}$ , c-HVZK with multiple transcripts is not guaranteed. For a contrived example, suppose an ID scheme  $\Pi$  has transcripts  $(a, e, z)$  and is c-HVZK when  $sk$  is hidden. Compute an additive secret sharing of  $sk$ ,  $sk = sk_0 \oplus sk_1$ . Now create a new ID scheme  $\Pi'$  from  $\Pi$ , which replaces the first message  $a$  with  $a' = (a, sk_i)$ , where  $i \in \{0, 1\}$  is chosen randomly. Clearly,  $\Pi'$  remains c-HVZK for one transcript, but after a small number of transcripts are given to  $\mathcal{A}$ , she may recover  $sk$  in the real case, but not the simulated case. A slightly less contrived example may be constructed when  $\Pi'$  uses  $sk$  as the secret key in a  $k$ -time signature scheme that leaks  $sk$  if  $k + 1$  signatures are output.

Therefore, when proving c-HVZK for an ID scheme directly using a game-based definition like Eq. (4) it's important to allow  $\mathcal{A}$  to have multiple transcripts. In the context of EUF-CMA secure signatures, there are multiple signing queries, so multiple transcripts are a requirement. A good game-based definition of c-HVZK with multiple transcripts, as required for signature schemes is given in [GHHM20, Definition 2]. Lemma 18 shows that Picnic2 meets this definition.