A Machine-Checked EUF-CMA Proof for the Hybrid Fiat-Shamir Signature Scheme

Sara Zain[1]

Barkhausen Institut, Dresden, Germany

Abstract. Hybrid signatures are a practical approach to post-quantum migration, but their security analysis becomes subtle when two Fiat-Shamir (FS) components are binded through a single shared challenge. This paper presents the first machine-checked proof of EUF-CMA security for the FS-FS hybrid construction of Bindel and Hale [12], formalised in EasyCrypt in the Random Oracle Model (ROM). The proof is parametrised over abstract component interfaces and establishes the intended either-component security guarantee: for either choice of component, a hybrid forgery can be reduced to an EUF-CMA forgery against that component, together with the collision resistance of the message digest and a random-oracle guessing term. In this ROM formulation, the proof does not require an independent second-preimage-resistance assumption for the challenge hash. The mechanisation makes explicit two proof obligations arising from the hybrid structure and the abstract digest: an invariant linking the lazy-oracle state to an explicit query log at the verification point, and a module-restriction framing argument for the digest-collision reduction. We further machine-check honest-signing correctness for a concrete Schnorr-Schnorr instantiation and instantiate the abstract security proof with a Schnorr-Okamoto component pair without changing the core game-hopping argument. The formalisation thereby makes explicit the assumptions, invariants, and composition structure underlying the security argument for the FS-FS hybrid.

Keywords: Formal Verification · EasyCrypt · Hybrid Digital Signatures · Post-Quantum Cryptography.

1 Introduction

Digital signatures are designed to provide security under computational assumptions about the underlying cryptographic primitives. In the post-quantum transition, some assumptions underlying widely deployed signature schemes may no longer provide the required security in the presence of sufficiently powerful quantum computers. A natural transition strategy is therefore to combine a classical and a post-quantum component so that the resulting hybrid retains a security guarantee as long as at least one component remains secure.

However, simply combining two signatures does not provide such a guarantee. In particular, concatenation-based hybrids can be vulnerable to *signature-stripping* attacks: if the verifier can accept one component independently of the

other, an adversary may be able to remove or bypass the component that it cannot forge and rely only on the remaining scheme [13]. This motivates hybrid constructions (*hybridisation*) in which the components are cryptographically bound rather than merely verified in parallel.

Bindel and Hale [12] investigate this problem through a construction of hybrid-signature goals and propose tailored constructions designed to provide stronger hybrid guarantees. Among their constructions is the FS-FS scheme considered in this work, which combines two Fiat-Shamir-based components [20] through a single shared challenge. In this construction, the challenge is computed as $c = \mathcal{H}(w_1, w_2, \mathcal{D}(m))$, where w_1 and w_2 are the component commitments and $\mathcal{D}(m)$ is a digest of the message. The challenge simultaneously binds both component transcripts to the same message-dependent value. This shared challenge is also what makes the security argument substantially different from independently analysing two signature schemes. A valid forgery induces a joint verification point $x_V = (w_1, w_2, \mathcal{D}(m))$, and the proof must reason about whether this point has previously been queried to the random oracle, whether the digest value can be related to a previously signed message, and how the resulting hybrid forgery can be reduced to a forgery against an individual component. The two component executions therefore cannot simply be separated into independent Fiat-Shamir security arguments. Moreover, this design achieves proof composability, strong non-separability, and simultaneous verification—the three goals identified by Bindel and Hale as necessary for a *true* hybrid and these properties are actively studied in the context of post-quantum transition [9, 13, 17].

Bindel and Hale state an EUF-CMA security theorem for the FS-FS construction (Theorem 1, page 16) but leave its accompanying formal security analysis for future work [12]. This raises the question of whether the intended hybrid-security argument can be carried through mechanically while retaining both the shared challenge and abstraction over the component schemes.

This research answers this question by mechanically verifying the FS-FS security argument in EasyCrypt [6, 7]. The formalisation is carried out in the classical Random Oracle Model (ROM) [10] and formulates the game-based argument parametrically over two component interfaces. The resulting theorem gives, for each fixed component $i \in \{1, 2\}$, a reduction to the EUF-CMA security of that component, together with the random-oracle guessing term and a collision-resistance term for the message digest.

The machine-checked verification is non-trivial because the shared challenge creates proof obligations that do not arise in the corresponding single-component arguments. In particular, the proof maintains an explicit relationship between the lazy random oracle and a log of its classical queries, which is used to reason about whether the joint verification point x_V has previously been queried. The collision-resistance reduction additionally uses a tracker for digest evaluations. Since the digest is abstract in the security theorem, a module restriction is required to ensure that the digest cannot access or modify the tracker's state. These properties are established explicitly in the EasyCrypt development rather than being treated as implicit properties of the games.

Our formalisation also makes the assumption structure explicit. The challenge hash $\mathcal{H}$ is modelled as a lazy random oracle, whereas the message digest $\mathcal{D}$ remains abstract and its collision resistance is an explicit assumption. The proof consequently does not require a separate second-preimage-resistance assumption for $\mathcal{H}$: the case in which the adversary has not queried the random oracle at the verification point is handled directly by the fresh-value guessing bound $1/|\mathcal{R}|$. This is a consequence of the ROM model and of the particular game decomposition used in our formalisation; it does not claim that second-preimage resistance is unnecessary for the original construction under other models or proof approaches.

We complement the security result with a concrete correctness check. For the Schnorr-Schnorr instantiation, the correctness experiment uses a deterministic digest and mechanically verifies that an honestly generated signature is accepted. Determinism ensures that signing and verification compute the same message digest, while the concrete Schnorr reconstruction equations ensure that verification recovers the commitments used to derive the shared challenge. This establishes functional correctness for the concrete instantiation and is independent of the security game-hopping argument; the deterministic digest used in this experiment is not an additional assumption of the abstract security theorem.

The contributions are:

- *Machine-checked hybrid security proof.* This paper provides a complete EasyCrypt formalisation of the EUF-CMA game-based security argument for the FS-FS hybrid. The proof is organised as a sequence of game transitions and probability bounds and is parametrised over abstract component interfaces[1].
- *Explicit assumption and probability structure.* The formal theorem exposes the reduction to the EUF-CMA security of either component, the $1/|\mathcal{R}|$ random-oracle guessing term, and the collision-resistance term for the abstract message digest. The formalisation therefore makes explicit how the intended "either-component" security guarantee is realised in the ROM.
- *Mechanised proof invariants.* We develop and verify an explicit logging invariant for the lazy random oracle and an auxiliary-game argument for the fresh-challenge case. We additionally use module restrictions to establish the framing condition required by the abstract-digest collision reduction. These constructions address proof obligations introduced by the hybrid structure and by the abstraction of the digest.
- *Concrete correctness and instantiation.* We mechanically verify honest-signing correctness for the Schnorr-Schnorr instantiation with a deterministic digest and instantiate the abstract security result for the concrete component pair. The security assumptions concerning the abstract digest and the adversary remain explicit hypotheses of the resulting bound.
- *Interface-level generality.* We additionally instantiate one component with a structurally different Σ-protocol to check that the game-hopping argument depends on the component interface rather than on the scalar structure

[1] Full EasyCrypt development of the hybrid scheme is available at: https://gitlab.barkhauseninstitut.org/szain/pqc-classic-project

of Schnorr. The security proof itself remains unchanged; only the concrete component instantiation and its associated side-conditions differ.

Structure. Section 2 presents the FS-FS construction, the concrete Schnorr instantiation, and the formalisation setting. Section 3 positions the work with respect to hybrid signatures and machine-checked cryptographic proofs. Section 4 states the main theorem and develops the game-hopping proof. Sections 5 concludes.

2 Construction and Formalisation Setting

We first describe the FS-FS hybrid signature construction and fix the formal setting used throughout the paper. The construction combines two Fiat-Shamir-based components through a single shared challenge, with a message digest included in the challenge input. This shared challenge binds the two component transcripts and is the central structural feature that the subsequent game-based proof must preserve.

We first present the abstract construction, then give the concrete Schnorr instantiation used in our correctness check and security corollary. We finally describe the EUF-CMA experiment and its EasyCrypt representation. This separation allows the subsequent game-hopping argument to remain parametric in the component interfaces, while the algebraic reasoning needed for correctness is established separately for the concrete instantiation.

The FS-FS construction. The Fiat-Shamir (FS) transform [20] converts a Σ-protocol into a signature scheme by deriving the verifier's challenge from the prover's commitment and the message using a hash function. The FS-FS construction of Bindel and Hale [12] applies this idea to two component Σ-protocols, but uses a *single shared challenge* rather than independent challenges for the two components.

Let $\mathbf{C}_1$ and $\mathbf{C}_2$ denote the two component Σ-protocols. Each component provides procedures for key generation, commitment generation, and response computation. The hybrid construction is parameterised by the two components, a digest function $\mathcal{D}$, and a challenge hash function $\mathcal{H}$.

Key generation independently generates $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathbf{C}_i.\mathsf{keygen}()$ for $i \in \{1, 2\}$, and returns $\mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2), \mathsf{sk} = (\mathsf{sk}_1, \mathsf{sk}_2)$. To sign a message m, each component first generates a commitment: $(w_i, r_i) \leftarrow \mathbf{C}_i.\mathsf{commit}(\mathsf{sk}_i)$. The message is then compressed to $d = \mathcal{D}(m)$, and both commitments and the digest are bound to a single challenge $c = \mathcal{H}(w_1, w_2, d)$. The component responses are computed using this common challenge, $z_i = \mathbf{C}_i.\mathsf{respond}(\mathsf{sk}_i, r_i, c)$ and the resulting hybrid signature is $\sigma = (c, z_1, z_2)$. Verification reconstructs the two commitments from the public keys, the shared challenge, and the responses: $w_i' = \mathsf{recon}_i(\mathsf{pk}_i, c, z_i)$. It then recomputes the digest and checks $c = \mathcal{H}(w_1', w_2', \mathcal{D}(m))$. Thus, the same challenge simultaneously binds both component transcripts to the message digest. This shared binding is the essential structural feature of the FS-FS construction and is also the pivot of our security proof.

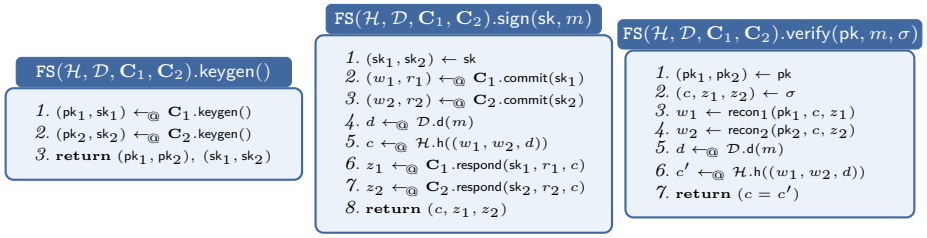


Fig. 1. The hybrid FS–FS signature scheme $\Sigma_h = \mathtt{FS}(\mathcal{H}, \mathcal{D}, \mathbf{C}_1, \mathbf{C}_2)$. Both responses share the single challenge $c = \mathcal{H}.\mathsf{h}((w_1, w_2, d))$, $d = \mathcal{D}.\mathsf{d}(m)$. Verification re-hashes the reconstructed commitments $w_i = \mathsf{recon}_i(\mathsf{pk}_i, c, z_i)$ and checks $c' = c$. The shared input $x_V = (w_1, w_2, d)$ is the pivot of the proof. $\mathcal{H}$ is `LRO` or its logging wrapper `LH` in the security games.

We write $x_V = (w_1, w_2, d)$, where $d = \mathcal{D}(m)$, for the verification hash input. For an honestly generated signature, correctness of the underlying Σ-protocols gives

$$\mathsf{recon}_i\big(\mathsf{pk}_i, c, \mathbf{C}_i.\mathsf{respond}(\mathsf{sk}_i, r_i, c)\big) = w_i.$$

Figure 1 shows the three algorithms. Consequently, when the digest evaluation is consistent between signing and verification, the verification hash input is exactly the same value x_V used to generate the challenge.

In our EasyCrypt development, $\mathbf{C}_1$ and $\mathbf{C}_2$ are abstract module interfaces, and the hybrid construction is defined parametrically over conforming implementations. This abstraction allows the game-based security argument to be separated from the internal algebraic structure of the component protocols.

Schnorr instantiation. We first instantiate both components with Schnorr's Σ-protocol. For each component $\mathsf{SchnorrComp}_i$, the secret key is sampled as $\mathsf{sk}_i \leftarrow \$\mathbb{Z}_q$ and the public key is $\mathsf{pk}_i = g^{\mathsf{sk}_i}$. A commitment samples $k \leftarrow \$\mathbb{Z}_q$ and returns (g^k, k). Given challenge c, the response is $z_i = k + c\mathsf{sk}_i$, and reconstruction is $\mathsf{recon}_i(\mathsf{pk}_i, c, z_i) = g^{z_i}\mathsf{pk}_i^{-c}$. For an honestly generated response,

$$g^{z_i}\mathsf{pk}_i^{-c} = g^{k+c\mathsf{sk}_i}(g^{\mathsf{sk}_i})^{-c} = g^k,$$

so reconstruction recovers the original commitment.

This instantiation recovers the FS-FS construction considered by Bindel and Hale [12] as a concrete special case of our abstract construction. More generally, the security proof is parametric in the component interfaces and does not depend on the specific algebraic representation of the keys, commitments, or responses.

Schemes with explicit abort/reject branches (e.g., FS with aborts [18]) are not covered natively by the present interface. Supporting them requires extending the component model with an explicit abort outcome and adapting the game sequence accordingly.

Throughout, q denotes the group order of $\mathbb{G}$; since the Schnorr response space is $\mathcal{R} = \mathbb{Z}_q$, this coincides with $|\mathcal{R}|$ appearing in Theorem 1 and Corollary 1.

Concrete correctness check.

In addition to the security proof, we mechanically verify correctness for the concrete Schnorr-Schnorr instantiation. In particular, the correctness experiment generates a key pair, signs an arbitrary message using the hybrid signing algorithm, and immediately verifies the resulting signature. For this experiment, the digest is instantiated by a deterministic function. Hence, the signing and verification procedures compute the same value $\mathcal{D}(m)$ for the same message. Moreover, for each component, the concrete Schnorr response reconstructs the commitment that was used to derive the shared challenge.

Consequently, verification reconstructs the same pair of commitments and the same digest as in the signing procedure. It therefore evaluates the lazy random oracle at the same input used during signing and recovers the previously sampled challenge contained in the signature.

The complete honest-signing-verification experiment is mechanically verified in EasyCrypt, establishing that every honestly generated signature is accepted for the concrete instantiation. This correctness result is separate from the EUF-CMA game-hopping proof: it establishes functional correctness of the concrete Schnorr-Schnorr instantiation, rather than introducing an additional security assumption or being used as a premise of the main security theorem.

Generality check via Schnorr-Okamoto. To check that the abstract proof is not tied to scalar-valued Schnorr components, we additionally instantiate the second component with the Okamoto Σ-protocol. Here the secret key, commitment randomness, and response are pairs of elements of $\mathbb{Z}_q$, while the corresponding group elements are represented as products of two generators. The resulting component nevertheless conforms to the same abstract interface.

This instantiation changes the internal algebraic structure of the second component but leaves the hybrid construction and its game-hopping argument unchanged. It therefore provides a concrete check that the security proof is parametrised by the component interface rather than by the scalar structure of Schnorr.

Security model. We prove EUF-CMA security in the Random Oracle Model (ROM) [10]. The adversary $\mathcal{A}$ receives the hybrid public key and has adaptive access to the hybrid signing oracle and to the random oracle $\mathcal{H}$. It wins if it outputs a pair (m^*, σ^*) such that $\mathsf{FS.verify}(\mathsf{pk}, m^*, \sigma^*) = \mathsf{true}$ and m^* was not previously submitted to the signing oracle.

The random oracle is modelled explicitly in EasyCrypt by a lazy sampling map. On a query x, the oracle returns the previously sampled value if x has already been queried; otherwise it samples a fresh value uniformly from the challenge space $\mathcal{R} = \mathbb{Z}_q$ and records the resulting pair. This explicit representation is important for the proof because the security argument distinguishes whether the adversary has previously queried the verification point x_V.

The digest $\mathcal{D}$ is deliberately not modelled as a random oracle. Instead, it is treated abstractly and its collision resistance is captured by a separate assumption. This distinction is essential: the random-oracle treatment of $\mathcal{H}$ permits us

to reason about a fresh challenge at an unqueried verification point, whereas the security of the message digest must be established independently.

The main theorem therefore considers an abstract digest and abstract component implementations subject to the required interface and losslessness conditions. The concrete Schnorr instantiation discharges the corresponding component-side conditions, while the digest and adversarial assumptions remain explicit hypotheses of the security bound.

EasyCrypt formalisation. The security proof is formalised in EasyCrypt [6,7], a proof assistant for game-based [25] cryptographic proofs based on probabilistic relational Hoare logic (pRHL) [5]. We represent each security experiment as an EasyCrypt game whose main procedure returns a boolean result, and write $\Pr[G : \mathsf{res}]$ for the probability that the experiment returns `true`.

The proof uses two complementary forms of reasoning. Relational equivalence is used to establish that successive games preserve the relevant output distribution, while probabilistic Hoare reasoning is used for direct probability bounds. In particular, the latter is used to bound the probability that a fresh random-oracle value equals the challenge contained in a forgery.

A central mechanisation issue is that the lazy random oracle maintains its sampling map as mutable game state. We therefore introduce an explicit logging layer in the security games to record oracle queries and prove an invariant relating the log to the oracle's internal sampling map. This invariant allows the informal statement "the oracle value at an unqueried point is fresh" to be established as a property of the formal game rather than taken as an implicit global fact.

A second mechanisation issue arises from treating the digest abstractly. The collision-reduction game maintains a tracker for digest evaluations. Since the digest is abstract, the formal development additionally uses an EasyCrypt module restriction to ensure that the digest cannot access or modify the tracker's state. This permits the collision witnesses recorded during the adversary's execution to be transferred soundly to the collision-resistance experiment.

Lazy random oracle. In EasyCrypt, the ROM is realised as a lazy sampling map: the oracle module LRO maintains an internal program variable `HRO.RO.m`, where HRO denotes the random oracle module and `HRO.RO.m` is its internal sampling map, a finite map from inputs to outputs. A query $\mathcal{H}.\mathsf{h}(x)$ returns the stored value if x has been seen before; otherwise it draws a fresh uniform element of $\mathcal{R} = \mathbb{Z}_q$, records it in `HRO.RO.m`, and returns it. Each fresh output matches any fixed target with probability $1/q$—the `dp1E` lemma[2] used to bound the guessing branch (Lemma 4). Because the oracle's map (`HRO.RO.m`) is a program variable updated at runtime, the standard ROM argument—that an unqueried point returns a fresh uniform value—must be derived from the program's execution history rather than assumed globally. This is precisely what the logging invariant Inv (Lemma 4) establishes. We write LRO for the lazy-oracle module and `HRO.RO.m` for its internal sampling map; the two are distinct objects in EasyCrypt.

[2] `lemma dp1E (z:chall) : mu1 dp z = 1%r / q%r.` Here `mu1 dp z` is the probability that `dp` outputs z; this is used in the guessing branch to confirm that a fresh oracle output matches any fixed value with probability exactly $1/q$.

Module restrictions. EasyCrypt modules can be declared with restrictions $\{-M\}$, mechanically enforcing that a module cannot read or write the state of M. We use this in Lemma 8 to enforce that the abstract digest $\mathcal{D}$ cannot alter the internal state of the collision-resistance tracker TD (proof pattern P3). This condition is a consequence of abstracting $\mathcal{D}$: a concrete hash function has no mutable state and no mechanism to interact with game-internal structures, so the condition simply does not arise when $\mathcal{D}$ is concrete, as in [12]. It is not that the condition is overlooked in the informal argument—it does not exist there at all; abstracting $\mathcal{D}$ to make the construction parametric introduces a new proof obligation that the module restriction $\{-\mathsf{TrackD}\}$ discharges mechanically.

Formalisation assumptions. We adopt standard losslessness [2, 4, 21]: $\mathcal{A}$, $\mathcal{D}$, and all procedures of $\mathbf{C}_1, \mathbf{C}_2$ terminate with probability 1 when their oracles do; discharged concretely for $\mathsf{SchnorrComp}_i$ in Corollary 1.

Our component interface does not model explicit abort/rejection outcomes. Consequently, Fiat-Shamir-with-aborts constructions are outside the present formalisation. Supporting such schemes would require extending the component interface and adapting the signing and security games to account explicitly for rejection and restart behaviour.

3 Related Work

Hybrid signatures. Hybrid signatures have been studied as a means of combining cryptographic components while retaining security when one of the components remains secure. Bindel et al. [13] formalised security concerns arising from hybridisation, including signature-stripping attacks on simple concatenation-based constructions.

Bindel and Hale [12] subsequently developed a richer construction of (true) hybrid-signature goals and proposed tailored constructions designed to achieve stronger forms of hybrid security. Among these is the FS-FS construction considered in this work, which combines two Fiat-Shamir-based components through a single shared challenge. Their work presents the constructions and their intended security properties, but explicitly leaves the accompanying formal security analysis for future work [12].

Our work addresses this gap for the FS-FS construction by mechanically verifying its game-based security argument in EasyCrypt [6, 7]. In particular, we formalise the security proof at the level of the abstract component interfaces rather than only proving a concrete Schnorr instance. The resulting theorem exposes the same essential "either-component" security structure while making explicit the probability loss associated with the random-oracle guessing case and the collision-resistance assumption on the message digest. The concrete Schnorr-Schnorr instantiation is then used to establish functional correctness and to instantiate the abstract security result.

This distinction is important from a formal-verification perspective: the shared challenge $c = \mathcal{H}(w_1, w_2, \mathcal{D}(m))$ binds the two component transcripts at the same

random-oracle point. Consequently, the security argument cannot simply be obtained by composing two independently verified Fiat-Shamir proofs. Our formalisation instead follows the hybrid proof structure and mechanically establishes the game transitions and invariants required to relate the joint verification point to the component security games.

Other work has considered hybrid combiner pen-and-paper cryptographic constructions, including hybrid key-exchange mechanisms [11] and standardisation oriented approaches to composite cryptography [9, 23]. These works motivate the need for security arguments that remain meaningful across different component choices; our contribution is complementary in that it addresses the machine-checked verification of a non-trivial hybrid signature proof.

Machine-checked proofs. Machine-checked game-based cryptographic proofs have a substantial history. EasyCrypt [6, 7] provides language and proof techniques for expressing cryptographic game-hopping [25] and reasoning about their equivalence and probability bounds. Earlier machine-checked work on digital signatures was carried out in CertiCrypt [8], establishing the feasibility of mechanically verifying standard signature-security arguments [27]. Subsequent work has demonstrated reusable EasyCrypt techniques for machine-checked EUF-CMA proofs of discrete-logarithm-based signatures, including tight reductions [19, 26].

Machine-checked proofs have also been developed for post-quantum cryptographic primitives. For example, Hülsing et al. [21] formally verified security and correctness properties of Saber in EasyCrypt, while Barbosa et al. [2] identified and repaired a subtle gap in the security analysis of Dilithium and provided a mechanised ROM proof.

Our setting differs from these single-scheme proofs in the structure of the security argument rather than merely in the underlying primitive. The adversary interacts with a hybrid signing oracle whose two component executions are coupled (or binded together) through one random-oracle challenge. The formalisation therefore has to preserve information about the common oracle transcript while reasoning about both component paths. In addition, because the message digest is abstract in our security theorem, the collision reduction requires a tracker whose state must be protected from the abstract digest module. These requirements lead to the explicit oracle-query invariant and module restrictions described in Section 4.

The contribution is consequently complementary to previous machine-checked proofs of individual signature schemes: rather than re-verifying the security of a particular primitive, we mechanically verify a security argument for a construction whose security depends on the interaction of multiple component proofs through a shared challenge.

Fiat-Shamir and the random-oracle model. The Fiat-Shamir (FS) transform [20] and its security analysis in the random-oracle model [10, 24] provide the cryptographic foundation for the component schemes considered. Our work does not introduce a new Fiat-Shamir security argument; instead, it verifies the hybrid composition in which the Fiat-Shamir challenges of two components are derived jointly.

This distinction is central to the scope of our contribution. The random-oracle reasoning used in the proof is not applied independently to two component signatures. Instead, the verification point contains both reconstructed commitments and the message digest. The formal proof must therefore establish when this joint point has previously been queried and, when it has, relate the resulting challenge to the component-level forgery reductions.

QROM. The quantum random-oracle model (QROM) extends the random-oracle model to adversaries that can query the oracle in quantum superposition [14]. Security of Fiat-Shamir-type constructions in the QROM has been studied by, among others, Kiltz et al. [22] and Don et al. [16]. Techniques such as compressed oracles provide a way to reason about quantum oracle queries while retaining information about the oracle's queried points [28].

The present work is restricted to the classical ROM. In particular, our EasyCrypt proof represents the random oracle by an explicit lazy sampling map and maintains an explicit query log. This representation is appropriate for the classical game-based argument developed here, but it does not directly provide a QROM proof. Extending our mechanisation to the QROM would therefore require replacing the classical query-log invariant and re-establishing the corresponding freshness and verification-point arguments in a quantum-compatible formal model. The EasyPQC [3] provides a framework for QROM proofs in EasyCrypt, but it does not yet support the multi-component hybrid setting considered here and also it is under development. We therefore regard QROM formalisation as outside the scope of this work rather than as a property established by the present development.

4 Game-Based Security Proof Structure

The security argument for the FS-FS construction hinges on the shared Fiat-Shamir challenge

$$c = \mathcal{H}(x_V), \qquad x_V = (w_1, w_2, \mathcal{D}(m)),$$

which jointly binds the two component transcripts to the message digest. This coupling is the central difference from the standard security proofs of individual Fiat-Shamir signatures: the two components cannot be analysed independently until the verification point has first been related to the random-oracle transcript and to the digest. Consequently, a hybrid forgery cannot be analysed by considering the two component signatures independently. So, the proof must first establish how the forged verification point relates to the random-oracle transcript and to previously signed message digests.

We formalise the security argument of Bindel and Hale [12] using a sequence of games in EasyCrypt. Before presenting the individual game transitions, we first state the security claim that our mechanisation addresses. This also fixes the correspondence between the theorem stated in [12] and the formal statement proved in our formalisation (Theorem 1).

Bindel–Hale security claim. Bindel and Hale (Theorem 1, p.16) [12] state that the FS-FS construction is EUF-CMA secure provided that the underlying Fiat-Shamir components satisfy the required EUF-CMA security assumption, the digest function is collision resistant, and the shared hash function satisfies the required second-preimage property. In particular, the theorem gives an "either component" guarantee: a forgery against the hybrid can be reduced to a forgery against either component, together with the corresponding digest and hash assumptions.

Our formalisation makes this claim precise in the ROM. We model $\mathcal{H}$ as a lazy random oracle and therefore do not introduce second-preimage resistance for $\mathcal{H}$ as a separate computational assumption. Instead, the case in which the adversary never queries the random oracle at the verification point x_V is handled explicitly by a guessing bound of $1/|\mathcal{R}|$. The digest $\mathcal{D}$, in contrast, remains an abstract deterministic function and is therefore accounted for by an explicit collision-resistance term. The resulting formal bound is stated in Theorem 1.

Proof at a glance. Let (m, σ) be a successful forgery, with $\sigma = (c, z_1, z_2)$. The verifier reconstructs

$$w_i = \mathsf{recon}_i(\mathsf{pk}_i, c, z_i), \qquad i \in \{1, 2\},$$

and evaluates the shared hash on

$$x_V = (w_1, w_2, \mathcal{D}(m)).$$

Thus, acceptance requires

$$c = \mathcal{H}(x_V).$$

The proof separates successful forgeries according to whether the verification point x_V has previously appeared in the random-oracle transcript and whether a collision under the message digest has been detected.

First, if x_V was not previously queried to the random oracle, then the value $\mathcal{H}(x_V)$ is a fresh uniform element of $\mathcal{R}$. Consequently, the adversary must guess the challenge value, which gives the $1/|\mathcal{R}|$ contribution.

Second, when x_V has been queried, the proof tracks all digest evaluations performed during the experiment. If the tracker detects two distinct messages $m \neq m'$ with $\mathcal{D}(m) = \mathcal{D}(m')$, the recorded messages form a valid collision for $\mathcal{D}$, yielding a reduction to the collision-resistance game.

In the remaining good branch, neither of these events occurs. The shared challenge is then fixed by a random-oracle query at the verification point, while the absence of a digest collision prevents the message digest from being shared by two distinct messages. In this branch, a successful hybrid forgery can be transformed into a valid forgery against either component scheme. More precisely, for each fixed $i \in \{1, 2\}$, a symmetric reduction embeds the hybrid experiment into the EUF-CMA experiment of Σ_i and reconstructs the commitment of the other component from the hybrid forgery.

The remainder of this section formalises these steps as a sequence of game transitions. The first games expose the verification point and the random-oracle

transcript; the subsequent games introduce the digest-collision tracker and split on the corresponding bad event. The final reductions establish the component-security bound and the collision-resistance bound.

The proof is organised around the shared hash input $x_V = (w_1, w_2, d)$, where $d = \mathcal{D}(m)$, at which both component transcripts are jointly committed. A successful forgery either corresponds to an oracle query at x_V, or falls into the fresh-point guessing case handled by badGuess. In the ROM games, $\mathcal{H}$ is instantiated internally as `LRO` (whose program variable `HRO.RO.m` stores the sampling map) or its logging wrapper `LH` (which additionally maintains the query log `LH.`qH as a program variable), so $\mathcal{H}$ is not a free parameter at the game level; the games are written $G_i(\mathcal{A}, \mathcal{D}, \mathbf{C}_1, \mathbf{C}_2)$. Two structural equivalences ($G_0 \leftrightarrow G_1 \leftrightarrow G_2$) rewrite the real game to expose two bad events. Splitting on badGuess ($\mathcal{H}$ was never queried at x_V during the execution) and badD (a collision under $\mathcal{D}$) partitions success probability into three contributions, each bounded by an independent reduction. The nine-lemma argument is summarised in Figure 2.

Adversarial interfaces. For each fixed $i \in \{1, 2\}$, the component reduction receives the public key and signing interface of Σ_i from its EUF-CMA experiment and generates the key pair of the other component independently. It then simulates the hybrid signing oracle using the shared challenge returned by the component experiment. Given a hybrid forgery, the reduction reconstructs the commitment of the other component and returns the corresponding component transcript as its forgery. The two reductions are symmetric under exchanging the component indices.

The digest reduction $\mathcal{B}_D$ instead runs the hybrid adversary with the collision-tracking digest wrapper. Whenever the tracker detects a collision, it returns the recorded pair of distinct messages with equal digests to the collision-resistance experiment.

SPR in the ROM. Bindel and Hale [12] state their security theorem under three assumptions: EUF-CMA security of the componentsignatures, collision resistance of the digest $\mathcal{D}$, and second-preimage resistance (SPR) of the shared hash function $\mathcal{H}$. Our formalisation uses a different treatment of $\mathcal{H}$: rather than leaving its security as an independent assumption, we model $\mathcal{H}$ explicitly as a lazy random oracle.

This modelling choice changes how the corresponding proof obligation appears in the game sequence. Let x_V denote the verification point associated with a candidate forgery. If x_V has not been queried to the random oracle, the value $\mathcal{H}(x_V)$ used as the verification challenge is a fresh uniform element of $\mathcal{R}$. Consequently, a successful forgery in this branch requires the adversary to guess this value, which gives $\Pr[\mathsf{res} \wedge \mathsf{badGuess}] \leq \frac{1}{|\mathcal{R}|}$. This is the guessing bound established by our mechanised badGuess branch.

If x_V has been queried, the challenge is instead determined by the random-oracle transcript, and the proof proceeds through the queried branch of the game sequence. In particular, this branch is not bounded by a separate SPR experiment: after introducing the digest tracker, the proof splits on digest collisions and, in the collision-free case, derives the component-level EUF-CMA reduction.

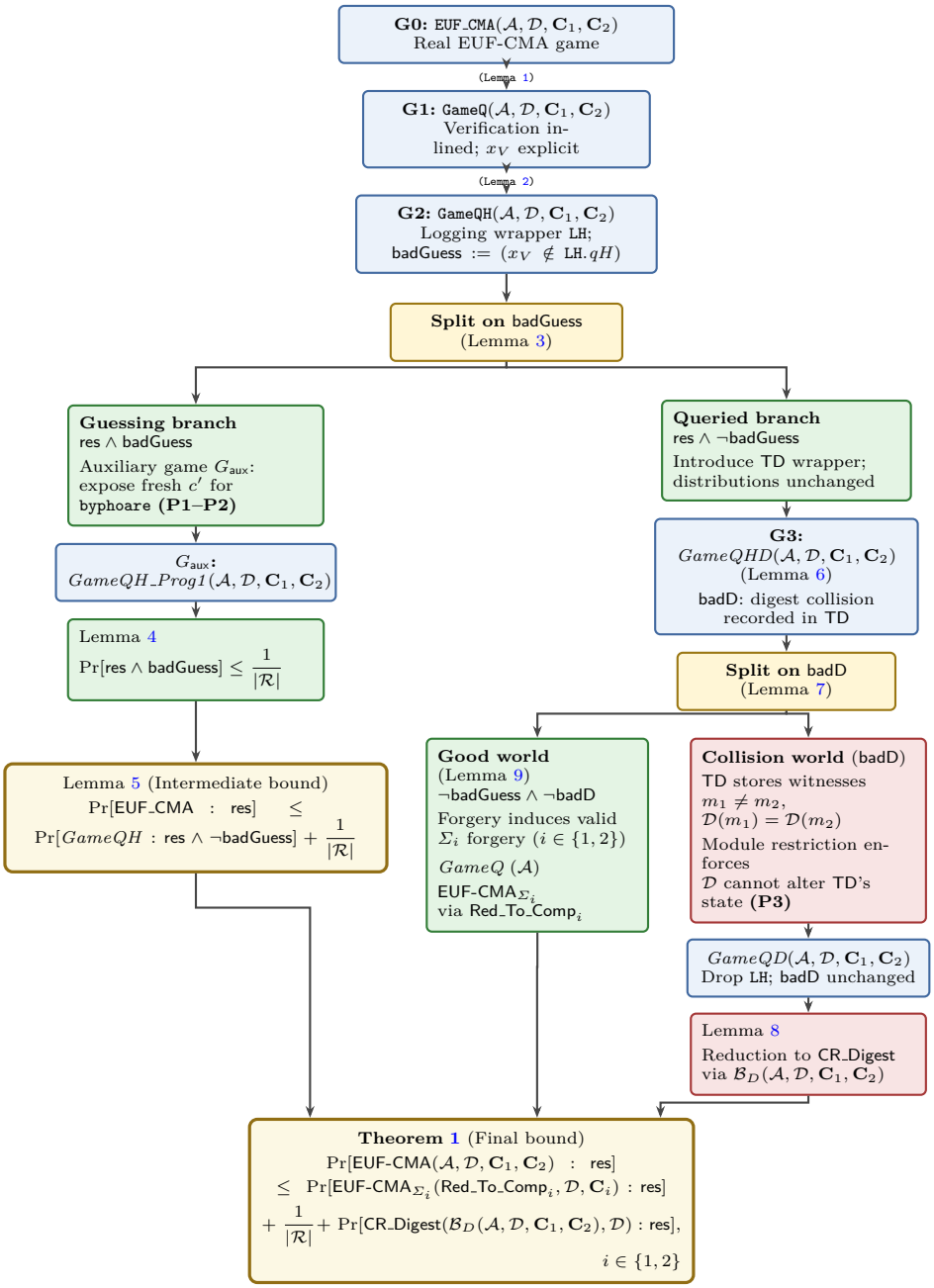


Fig. 2. Overview of the nine-lemma game-hopping proof for the hybrid FS-FS signature scheme (Theorem 1). The top chain ($G_0 \to G_1 \to G_2$) rewrites the real game to expose the verification hash input (x_V) and record whether it was queried. Splitting on badGuess yields a guessing branch, bounded by $1/|\mathcal{R}|$, and a queried branch. In the queried branch, G_3 introduces the digest tracker TD. Splitting on badD separates a good world, in which the hybrid forgery can be reduced to a forgery against either component (via a corresponding component-specific reduction), from a collision world, in which TD provides two distinct messages with the same digest and hence yields a collision-resistance reduction. Combining both branches yields the final bound.
Colours: games, bad events, reductions, branching points, bounds.

Thus, SPR does not occur as an independent computational assumption in our formal statement. More precisely, the role played by the corresponding exceptional case in the original theorem is discharged in our ROM formalisation by the explicit fresh-oracle-output argument and its $1/|\mathcal{R}|$ bound. This is a consequence of our modelling choice for $\mathcal{H}$; it does not assert that the original Bindel and Hale theorem can simply omit SPR without specifying the random-oracle model.

The asymmetry with the digest $\mathcal{D}$ is deliberate. We model $\mathcal{D}$ as an abstract deterministic function, rather than as a random oracle. Hence, the corresponding digest-collision event cannot be discharged by the same fresh-output argument and remains an explicit collision-resistance term in Theorem 1. The resulting formalisation therefore has two computational assumptions—EUF-CMA security of either component and collision resistance of $\mathcal{D}$—plus the explicit ROM guessing term $1/|\mathcal{R}|$.

Schnorr corollary. Instantiating Theorem 1 with $\mathbf{C}_i = \mathsf{SchnorrComp}_i$ discharges all component losslessness assumptions concretely; the adversary and digest assumptions remain as hypotheses. A concrete instantiation (Corollary 1) confirms the bound is non-vacuous for the FS-FS hybrid of [12].

4.1 Sequence of games

Each game is parameterised by adversary $\mathcal{A}$, digest $\mathcal{D}$, and abstract sigma components $\mathbf{C}_1$, $\mathbf{C}_2$. Throughout, $\leftarrow_{@}$ denotes a call to a probabilistic procedure. We assume $\mathcal{A}$, $\mathcal{D}$, and all procedures of $\mathbf{C}_1$, $\mathbf{C}_2$ are lossless when their oracles are lossless, as assumed in Section 2.

Game G_0: EUF_CMA. The initial game G_0 is the real EUF-CMA experiment for hybrid `FS-FS` in the ROM:

$$G_0(\mathcal{A}, \mathcal{D}, \mathbf{C}_1, \mathbf{C}_2) := \mathsf{EUF_CMA}(\mathcal{A}, \mathcal{D}, \mathbf{C}_1, \mathbf{C}_2).$$

It initialises `LRO`[3] as a lazy map, provides $\mathcal{A}$ with the hybrid signing oracle $O.\mathsf{sign}$ and the random oracle $\mathcal{H} = \mathtt{LRO}$, and sets $res = true$ when $\mathcal{A}$ returns a forgery (m, σ) that verifies under pk with $m \notin Q_S$. Figure 3 (left) shows G_0.

Game G_1: GameQ. In G_1, we inline the verification procedure so that the verification hash input becomes explicit as

$$x_V = (w_1, w_2, d), \quad \text{where } d = \mathcal{D}(m).$$

This step is purely structural: it exposes internal computations without changing any values returned to the adversary or any random choices made by the experiment.

[3] The lazy random oracle is implemented as a sampling map: a query returns the previously stored value if the input has been seen, and otherwise samples a fresh uniform challenge and stores it for future reuse.

Lemma 1 (Equivalence $G_0 \leftrightarrow G_1$).

$$\Pr\big[G_0(\mathcal{A},\mathcal{D},\mathbf{C}_1,\mathbf{C}_2) : \mathsf{res}\big] = \Pr\big[G_1(\mathcal{A},\mathcal{D},\mathbf{C}_1,\mathbf{C}_2) : \mathsf{res}\big].$$

Proof. G_1 inlines $\mathsf{FS}(\mathtt{LRO},\mathcal{D},\mathbf{C}_1,\mathbf{C}_2).\mathsf{verify}$, decomposing σ, computing w_1, w_2 via the abstract reconstruction operators, and re-hashing to expose $x_V = (w_1, w_2, d)$, without altering any probability distribution.

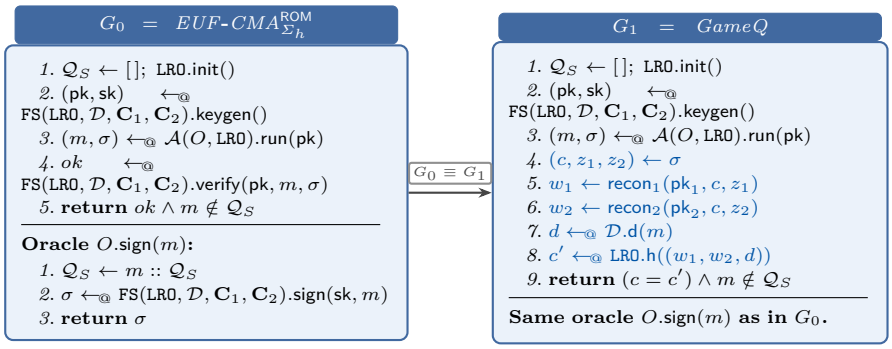


Fig. 3. Lemma 1: In G_0, verification is an opaque call to $\mathsf{FS}(\mathtt{LRO},\mathcal{D},\mathbf{C}_1,\mathbf{C}_2).\mathsf{verify}$ (line 4); G_0 does not inspect σ. G_1 inlines the verify body (lines 4–8, blue), applying the abstract reconstruction operators $\mathsf{recon}_i(\mathsf{pk}_i, c, z_i)$ to recover the commitments and making $x_V = (w_1, w_2, d)$ explicit. No distribution changes; the step is purely structural.

Game G_2: GameQH. G_2 replaces LRO with the logging wrapper $\mathtt{LH} = \mathsf{LogH}(\mathtt{LRO})$, which forwards every query to LRO unchanged while appending the input to the log $\mathtt{LH}.qH$. Using this log, we define the event

$$\mathit{badGuess} \overset{\text{def}}{\iff} x_V \notin \mathtt{LH}.qH,$$

which captures the case where the adversary never queried $\mathcal{H}$ at the verification point (x_V). Figure 4 shows G_2, the LH oracle, the invariant Inv, and the probability split.

Lemma 2 (Equivalence $G_1 \leftrightarrow G_2$).

$$\Pr\big[G_1(\mathcal{A},\mathcal{D},\mathbf{C}_1,\mathbf{C}_2) : \mathsf{res}\big] = \Pr\big[G_2(\mathcal{A},\mathcal{D},\mathbf{C}_1,\mathbf{C}_2) : \mathsf{res}\big].$$

Proof. LH appends x to the log before forwarding to LRO, returning the same value; every query produces the same distribution as in G_1. The only change is the availability of the event *badGuess*.

Lemma 3 (Probability split on badGuess).

$$\Pr\big[G_2(\mathcal{A},\mathcal{D},\mathbf{C}_1,\mathbf{C}_2) : \mathsf{res}\big] = \Pr\big[G_2 : \mathsf{res} \wedge \mathsf{badGuess}\big] + \Pr\big[G_2 : \mathsf{res} \wedge \neg\mathsf{badGuess}\big].$$

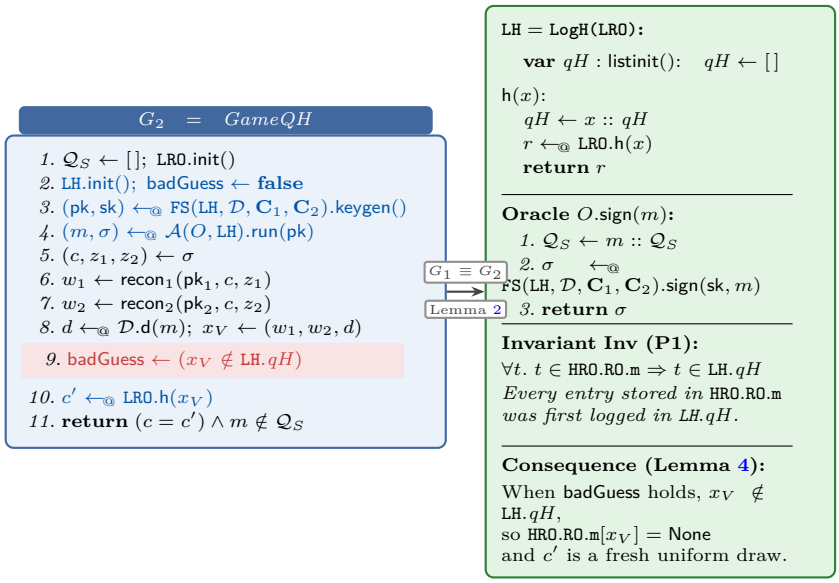


Fig. 4. G_2 replaces LRO with LH = LogH(LRO) (lines 2–4), which maintains the query log LH.qH alongside the sampling map and forwards every query to LRO unchanged. The final verification query (line 10) uses LRO directly; thus $x_V \notin$ LH.qH iff the adversary did not query $\mathcal{H}$ at x_V. Line 9 (red) assigns only badGuess; the badD flag is absent in G_2. By invariant Inv (proof pattern P1), HRO.RO.m$[x_V]$ = None under badGuess, so c' is a fresh uniform draw from $\mathcal{R}$.

Proof. The events $\{\mathsf{res} \wedge \mathsf{badGuess}\}$ and $\{\mathsf{res} \wedge \neg\mathsf{badGuess}\}$ are disjoint and their union is $\{\mathsf{res}\}$. The equality therefore follows by finite additivity of probability, formalised in EasyCrypt by `Pr[mu_split badGuess]`.[4]

The first branch is the fresh-challenge case. We next bound $\Pr[G_2 : \mathsf{res} \wedge \mathsf{badGuess}]$ by showing that, under badGuess, the evaluation of the random oracle at the verification point is fresh. This requires relating the lazy oracle's internal state to the explicit query log maintained by the formalisation.

(P1) Logging invariant. We maintain the invariant

$$\mathsf{Inv}(\mathtt{HRO.RO.m}, \mathtt{LH}.qH) \;:=\; \forall t.\ t \in \mathtt{HRO.RO.m} \Rightarrow t \in \mathtt{LH}.qH$$

which states that every input recorded in the random oracle's internal sampling map also occurs in the explicit query log LH.qH.

The invariant holds initially, when both structures are empty, and is preserved by oracle calls: LH.h records an input in LH.qH before forwarding the query to the lazy random oracle, while the signing oracle routes its hash calls through

[4] `Pr[mu_split e]` decomposes $\Pr[G : \varphi]$ into $\Pr[G : \varphi \wedge e] + \Pr[G : \varphi \wedge \neg e]$ (exact equality). The companion rule `Pr[mu_sub]` formalises monotonicity $A \subseteq B \Rightarrow \Pr[G : A] \leq \Pr[G : B]$, used only in Lemma 7.

LH. Together with the adversary-preservation condition stated in Section 2, this establishes the invariant throughout the execution.

Consequently, when badGuess holds, i.e., $x_V \notin \mathtt{LH}.qH$, the invariant implies that x_V has no entry in HRO.RO.m. Hence the subsequent evaluation of the lazy random oracle at x_V samples a fresh uniform challenge.

(P2) Auxiliary game. The auxiliary game G_{aux} (Figure 5) is identical to G_2 except that it stores the challenge contained in the forged signature in a fresh variable cc immediately after parsing σ: $cc \leftarrow c$. This transformation exposes the forged challenge as a fixed target for the subsequent oracle evaluation. Under badGuess, P1 ensures that the evaluation of the random oracle at x_V is fresh. Thus the relevant event is that this fresh uniform oracle output equals the already fixed value cc, which yields the corresponding $1/|\mathcal{R}|$ guessing bound.

The equivalence between G_2 and G_{aux} on the badGuess branch, together with the fresh-sampling bound, is formalised in the subsequent guessing lemma.

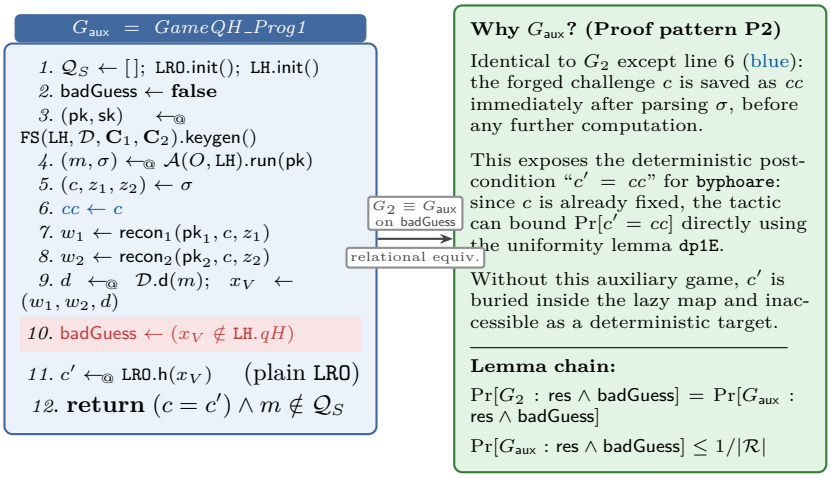


Fig. 5. Auxiliary game G_{aux} (Lemma 4, P2): identical to G_2 except line 6 (blue), which stores c as cc before any further computation. Under badGuess, $\mathtt{HRO.RO.m}[x_V] = \mathsf{None}$, so c' is fresh and $\Pr[c' = cc] = 1/q$. This enables byphoare by exposing a deterministic target value, which is not directly accessible in G_2.

Lemma 4 (Guessing bound).

$$\Pr\left[G_2(\mathcal{A}, \mathcal{D}, \mathbf{C}_1, \mathbf{C}_2) : \mathsf{res} \wedge \mathsf{badGuess}\right] \leq \frac{1}{q}, \quad \textit{where } q = |\mathcal{R}|.$$

Proof. Patterns P1–P2 above give $\Pr[G_2 : \mathsf{res} \wedge \mathsf{badGuess}] = \Pr[G_{\mathsf{aux}} : \mathsf{res} \wedge \mathsf{badGuess}]$. Applying byphoare in G_{aux} with $\Pr[c' = cc] = 1/|\mathcal{R}|$ (from dp1E) gives the bound, which is then transported back to G_2.

Lemma 5 (Intermediate bound).

$$\Pr[G_0 : \mathsf{res}] \le \Pr[G_2 : \mathsf{res} \wedge \neg\mathsf{badGuess}] + 1/|\mathcal{R}|.$$

Proof. Immediate from Lemmas 1–3 and Lemma 4.

Game G_3: GameQHD (digest tracking). On the other branch $\neg badGuess$, $\mathcal{A}$ did query $\mathcal{H}$ at x_V, so the proof introduces a digest-tracking wrapper. G_3 (Figure 6) wraps $\mathcal{D}$ with $\mathsf{TD} = \mathsf{TrackD}(\mathcal{D})$, which records the first message seen for each digest value in its map *seen*, and raises
$\mathsf{badD} \overset{\text{def}}{\iff} \exists m \neq m' \text{ s.t. } \mathcal{D}(m) = \mathcal{D}(m')$.

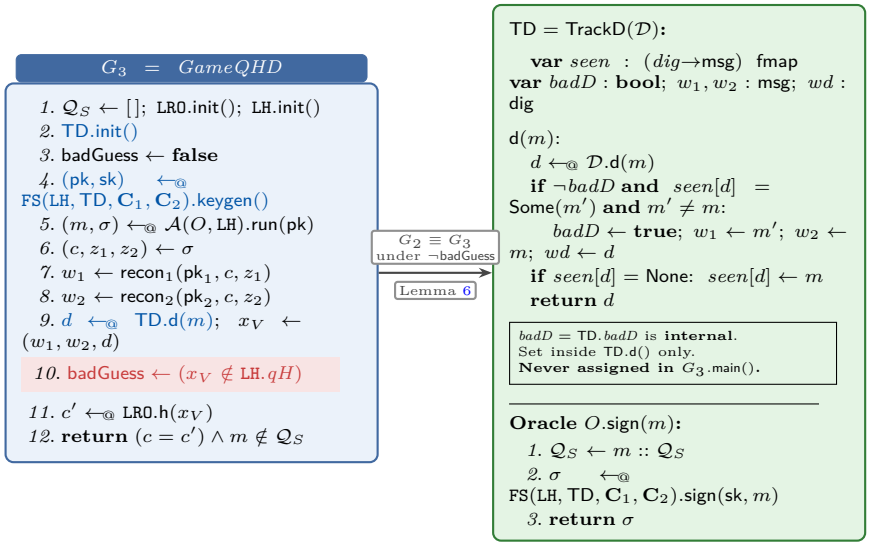


Fig. 6. G_3 introduces $\mathsf{TD} = \mathsf{TrackD}(\mathcal{D})$ (lines 2,4,9), a transparent wrapper that records digest evaluations in its internal map *seen* and raises badD upon the first collision. TD never modifies any return value of $\mathcal{D}.d$, so the output distributions of G_2 and G_3 are identical conditionally on ¬badGuess. Line 10 (red) assigns only badGuess; TD.*badD* is set exclusively inside TD.d() and is never assigned in G_3.main().

Lemma 6 (Equivalence $G_2 \leftrightarrow G_3$ under ¬badGuess).

$$\Pr\big[G_2(\mathcal{A}, \mathcal{D}, \mathbf{C}_1, \mathbf{C}_2) : \mathsf{res} \wedge \neg\mathsf{badGuess}\big] = \Pr\big[G_3(\mathcal{A}, \mathcal{D}, \mathbf{C}_1, \mathbf{C}_2) : \mathsf{res} \wedge \neg\mathsf{badGuess}\big].$$

Proof. TD calls $\mathcal{D}.\mathsf{d}(m)$ first and returns that exact value; it only observes and never modifies any output. Joint distributions are identical in G_2 and G_3, and the equivalence holds conditionally on ¬badGuess since G_3 retains LH.

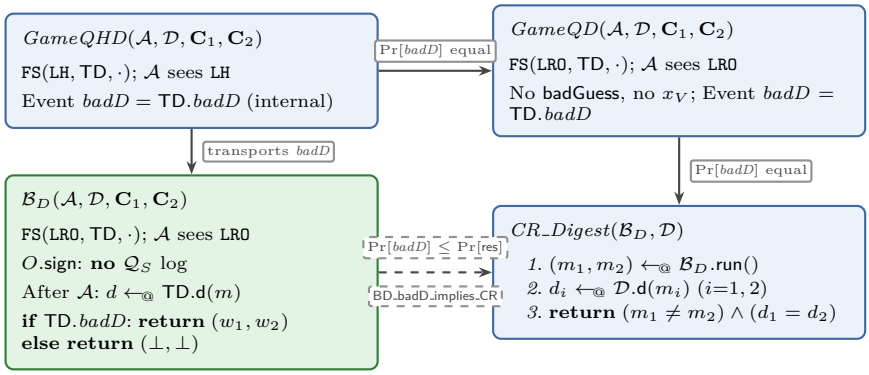
Lemma 7 (Probability split on badD).

$$\Pr\big[G_3 : res \wedge \neg badGuess\big] = \Pr\big[G_3 : res \wedge \neg badGuess \wedge \neg badD\big] + \Pr\big[G_3 : res \wedge \neg badGuess \wedge badD\big].$$

Since $\{res \wedge \neg badGuess \wedge badD\} \subseteq \{badD\}$*, probability monotonicity gives:*

$$\Pr\big[G_3 : res \wedge \neg badGuess\big] \;\leq\; \Pr\big[G_3 : res \wedge \neg badGuess \wedge \neg badD\big] + \Pr\big[G_3 : badD\big].$$

Proof. The events $\{res \wedge \neg badGuess \wedge badD\}$ and $\{res \wedge \neg badGuess \wedge \neg badD\}$ are disjoint and their union is $\{res \wedge \neg badGuess\}$. Hence the equality follows by `Pr[mu_split badD]`[4]. Monotonicity then yields the inequality.



$$\Pr[G_3 : badD] \;=\; \Pr[GameQD : badD] \;=\; \Pr[\mathcal{B}_D : badD] \;\leq\; \Pr[CR_Digest(\mathcal{B}_D, \mathcal{D}) : \mathsf{res}]$$

Fig. 7. Lemma 8: badD is transported from G_3 through GameQD and $\mathcal{B}_D$ to CR_Digest. Solid arrows are probability equalities (Steps 1–2); the dashed arrow is the soundness inequality (Step 3), using the collision invariant of TD and the module-restriction framing argument (proof pattern P3): the declaration $\{-\mathsf{TrackD}\}$ on $\mathcal{D}$ in EasyCrypt's type system mechanically prevents $\mathcal{D}$ from accessing any program variable of TD, ensuring the CR challenger's re-calls to $\mathcal{D}$ cannot retroactively alter the recorded collision witnesses.

Collision-world reduction. When badD occurs, the internal state of TD records two distinct messages $w_1 \neq w_2$ satisfying $D(w_1) = D(w_2)$, together with the collision digest w_d. Reduction $\mathcal{B}_D(\mathcal{A}, \mathcal{D}, \mathbf{C}_1, \mathbf{C}_2)$ runs $\mathcal{A}$ with the TD-wrapped digest, calls TD.d once more on the forgery message after $\mathcal{A}$ outputs, then returns (w_1, w_2) to the CR challenger; its signing oracle carries no query log. Figure 7 shows the reduction chain.

Lemma 8 (CR bound).

$$\Pr[G_3 : \mathsf{badD}] \;\le\; \Pr[\mathit{CR_Digest}(\mathcal{B}_D(\mathcal{A}, \mathcal{D}, \mathbf{C}_1, \mathbf{C}_2), \mathcal{D}) : \mathit{res}].$$

Proof. The proof proceeds in three steps, corresponding to the three arrows in Figure 7.

Step 1. The logging wrapper `LH` is dropped by replacing $\mathtt{FS}(\mathtt{LH}, \mathsf{TD}, \mathbf{C}_1, \mathbf{C}_2)$ with $\mathtt{FS}(\mathtt{LRO}, \mathsf{TD}, \mathbf{C}_1, \mathbf{C}_2)$ and giving $\mathcal{A}$ `LRO` directly. Since `LH` never wrote to `TD`'s state, the event badD is unaffected (two games are equivalent).

Step 2. `GameQD` and $\mathcal{B}_D$ are structurally identical on badD: the only differences are that `GameQD` additionally tracks signed messages in Q_S and performs a forgery-freshness check, neither of which can affect whether `TD` detected a collision.

Step 3 (soundness)**.** We show $\Pr[\mathcal{B}_D : \mathsf{badD}] \le \Pr[\mathtt{CR_Digest}(\mathcal{B}_D, \mathcal{D}) : \mathsf{res}]$. Three ingredients are required, none of which is visible in the informal argument.

Collision invariant: whenever badD holds, the witnesses in TD's state satisfy $\mathcal{D}(w_1) = \mathcal{D}(w_2)$, established at initialisation and maintained by the adversary-preservation assumption for TD's invariant.

Framing (P3): the additional calls to $\mathcal{D}$ by the CR game after $\mathcal{B}_D$ terminates cannot alter $\mathsf{TD}.\mathit{badD}$. This is non-trivial because $\mathcal{D}$ is abstract; in EasyCrypt, the module restriction $\{-\mathsf{TrackD}\}$ on $\mathcal{D}$ mechanically enforces at the type level that $\mathcal{D}$ cannot read or write any program variable of TD — a framing condition that holds by construction, requiring no further assumption.

Determinism: $\mathcal{D}$ returns consistent digests for seen messages and the collision witness (standard for any deterministic digest), so $\mathcal{B}_D$'s witnesses pass the CR check when $\mathcal{D}$ is re-called.

Core reduction to a component forgery. We now bound the good-world branch by reducing a successful hybrid forgery to a forgery against either component. The reduction is formally established in two parts. First, the good-world event in G_3 is related to the success event of $G_1 = \mathsf{GameQ}$ using the preceding game equivalences. Second, the EasyCrypt development establishes a relational simulation from GameQ to the corresponding component EUF-CMA game.

For each $i \in \{1, 2\}$, let $j = 3 - i$. We use the component EUF-CMA game $\mathsf{EUF\text{-}CMA}_{\Sigma_i}$ instantiated with the shared-hash verification procedure. For $i = 1$, a candidate forgery has the form $(m, (w_2, c, z_1))$, where w_2 is the commitment supplied for the other component. The component-1 verifier reconstructs $w_1 = \mathsf{recon}_1(\mathsf{pk}_1, c, z_1)$ and checks the shared challenge by evaluating $H(w_1, w_2, \mathcal{D}(m))$. The $i = 2$ game is defined symmetrically.

Consider first $i = 1$. The reduction $\mathcal{B}_1$ receives pk_1 from the Σ_1 challenger and generates the other component's key pair $(\mathsf{pk}_2, \mathsf{sk}_2)$ using $\mathbf{C}_2.\mathsf{keygen}()$. It gives $(\mathsf{pk}_1, \mathsf{pk}_2)$ to $\mathcal{A}$ and simulates the hybrid signing oracle. For a signing query m, it computes $(w_2, r_2) \leftarrow \mathbf{C}_2.\mathsf{commit}(\mathsf{sk}_2)$, forwards (m, w_2) to the Σ_1 signing oracle, which returns (c, z_1), and then computes $z_2 \leftarrow \mathbf{C}_2.\mathsf{respond}(\mathsf{sk}_2, r_2, c)$. It returns (c, z_1, z_2) to $\mathcal{A}$. Thus, the component-1 part of the hybrid signing procedure is delegated to the Σ_1 signing oracle, while the component-2 part is generated locally. Figure 8 illustrates this component reduction for $i = 1$.

When $\mathcal{A}$ outputs a candidate hybrid forgery $(m, (c, z_1, z_2))$, the reduction computes $w_2 = \mathsf{recon}_2(\mathsf{pk}_2, c, z_2)$ and returns $(m, (w_2, c, z_1))$ as its Σ_1 forgery.

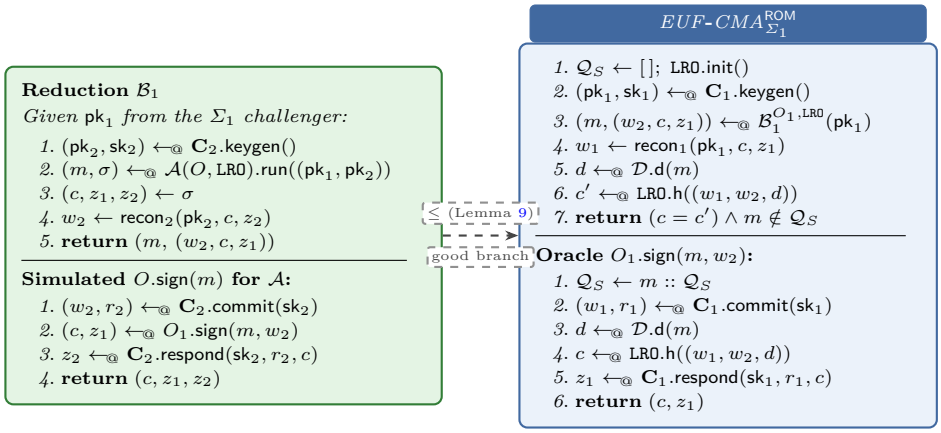


Fig. 8. Component reduction $\mathcal{B}_1$ used in Lemma 9. The reduction generates $(\mathsf{pk}_2, \mathsf{sk}_2)$ independently, simulates the hybrid signing oracle by delegating the Σ_1 component to its signing oracle, and returns (w_2, c, z_1) as the component forgery.

The reduction for $i = 2$ is symmetric: it generates $(\mathsf{pk}_1, \mathsf{sk}_1)$ locally, delegates the component-2 part of each signing query to the Σ_2 signing oracle, and returns $(m, (w_1, c, z_2)), w_1 = \mathsf{recon}_1(\mathsf{pk}_1, c, z_1)$, as the component-2 forgery.

The key formal property is that the reductions establish not only the validity of the extracted transcript, but also the correspondence between the hybrid and component games. For $i = 1$, the EasyCrypt development proves that whenever the GameQ execution returns *res*, the corresponding component execution also returns *res*. The relational invariant preserves the relevant global state, including the adversary, digest, component, and random-oracle states, and explicitly maintains equality of the signing-query lists. Consequently, the message-freshness condition checked by the component EUF-CMA experiment is preserved by the reduction. The corresponding simulation is established symmetrically for $i = 2$.

For the Σ_1 branch, suppose that $G_1 = \mathsf{GameQ}$ accepts a forgery $(m, (c, z_1, z_2))$, and define

$$w_1 = \mathsf{recon}_1(\mathsf{pk}_1, c, z_1), \qquad w_2 = \mathsf{recon}_2(\mathsf{pk}_2, c, z_2).$$

The reduction returns $(m, (w_2, c, z_1))$. The Σ_1 verifier reconstructs

$$w_1' = \mathsf{recon}_1(\mathsf{pk}_1, c, z_1) = w_1$$

and therefore evaluates the same hash input

$$(w_1', w_2, \mathcal{D}(m)) = (w_1, w_2, \mathcal{D}(m))$$

as the hybrid verifier. Therefore, the verification result is preserved by the reduction. The Σ_2 case is symmetric. The EasyCrypt relational proofs establish this verification correspondence together with the signing-oracle simulation and the preservation of the relevant state.

Lemma 9 (Core soundness). *For each $i \in \{1, 2\}$, let $j = 3 - i$. There exists a reduction $\mathcal{B}_i$ that generates $(\mathsf{pk}_j, \mathsf{sk}_j)$ independently using $\mathbf{C}_j.\mathsf{keygen}()$ and simulates the hybrid signing oracle for $\mathcal{A}$ using the Σ_i signing oracle, such that*

$$\Pr\big[G_3 : \mathsf{res} \wedge \neg\mathsf{badGuess} \wedge \neg\mathsf{badD}\big] \;\le\; \Pr\big[\mathsf{EUF\text{-}CMA}_{\Sigma_i}(\mathcal{B}_i(\mathcal{A}), \mathcal{D}, \mathbf{C}_i) : \mathsf{res}\big].$$

Both cases $i = 1$ and $i = 2$ are mechanically verified in EasyCrypt.

Proof. We first relate the good-world event in G_3 to the success event of $G_1 = \mathsf{GameQ}$. By Lemma 6, the success probabilities of G_2 and G_3 coincide when restricted to $\neg\mathsf{badGuess}$. Hence, by monotonicity,

$$\begin{aligned}\Pr\big[G_3 : \mathit{res} \wedge \neg\mathsf{badGuess} \wedge \neg\mathsf{badD}\big] &\le \Pr\big[G_3 : \mathit{res} \wedge \neg\mathsf{badGuess}\big] \\ &= \Pr\big[G_2 : \mathit{res} \wedge \neg\mathsf{badGuess}\big] \\ &\le \Pr[G_2 : \mathit{res}] \\ &= \Pr[G_1 : \mathit{res}],\end{aligned}$$

where the last equality follows from the equivalence $G_1 \leftrightarrow G_2$ of Lemma 2.

It remains to reduce $G_1 = \mathsf{GameQ}$ to the corresponding component EUF-CMA game. For $i = 1$, the EasyCrypt relational proof establishes

$$\mathsf{G_1.res} \Longrightarrow \mathsf{EUF\text{-}CMA}_{\Sigma_1}.\mathsf{res},$$

while preserving the relevant global state and the signing-query list. In particular, the simulation uses the same digest and random-oracle state on both sides, and the signing-query lists remain equal throughout the execution. Therefore, a fresh hybrid forgery remains fresh in the component game. The corresponding probability inequality gives

$$\Pr[G_1 : \mathit{res}] \le \Pr\big[\mathsf{EUF\text{-}CMA}_{\Sigma_1}(\mathcal{B}_1, \mathcal{D}, \mathbf{C}_1) : \mathit{res}\big].$$

The $i = 2$ case is established symmetrically, yielding

$$\Pr[G_1 : \mathit{res}] \le \Pr\big[\mathsf{EUF\text{-}CMA}_{\Sigma_2}(\mathcal{B}_2, \mathcal{D}, \mathbf{C}_2) : \mathit{res}\big].$$

Combining the appropriate component reduction with the preceding inequalities proves the lemma.

Remark. The component simulation is established from the full success event of $G_1 = \mathsf{GameQ}$ and does not require conditioning on $\neg\mathsf{badGuess}$ or $\neg\mathsf{badD}$. These conditions arise only in the preceding game-hopping argument, where they partition the hybrid success probability into the guessing, collision, and good-world branches. Lemma 9 therefore follows by composing the good-world-to-G_1 inequalities with the unconditional component simulation.

Final bound. Combining the guessing, collision, and good-world bounds of Lemmas 1–9 yields the following theorem.

Theorem 1 (EUF-CMA security of Σ_h). *For every adversary $\mathcal{A}$, abstract sigma-protocol components $\mathbf{C}_1, \mathbf{C}_2$, and digest $\mathcal{D}$, and for each $i \in \{1,2\}$, there exist reductions $\mathcal{B}_i$ and $\mathcal{B}_D$ such that*

$$\Pr\big[\mathsf{EUF\text{-}CMA}(\mathcal{A},\mathcal{D},\mathbf{C}_1,\mathbf{C}_2) : res\big] \;\le\; \frac{1}{|\mathcal{R}|} + \Pr\big[\mathsf{EUF\text{-}CMA}_{\Sigma_i}(\mathcal{B}_i(\mathcal{A}),\mathcal{D},\mathbf{C}_i) : res\big] + \Pr\big[\mathsf{CR}(\mathcal{B}_D(\mathcal{A},\mathcal{D},\mathbf{C}_1,\mathbf{C}_2),\mathcal{D}) : res\big].$$

Here, $\mathsf{EUF\text{-}CMA}_{\Sigma_i}$ denotes the component game with the shared hybrid-hash verification procedure. Consequently, Σ_h is EUF-CMA secure if, for at least one $i \in \{1,2\}$, the corresponding shared-hash component game is EUF-CMA secure and $\mathcal{D}$ is collision-resistant. Both symmetric instantiations of the component reduction ($i = 1$ and $i = 2$) are mechanically verified in EasyCrypt. The term $1/|\mathcal{R}|$ accounts for the guessing branch in the ROM, where the verification point was not queried by the adversary. Thus, no separate second-preimage-resistance assumption on the random oracle is required.

Corollary 1 (FS-FS Schnorr bound). *Let $Sch_i = \mathsf{SchnorrComp}_i$ for $i \in \{1,2\}$ and let $q = |\mathcal{R}|$. For every adversary $\mathcal{A}_S$ and digest $\mathcal{D}_S$, Theorem 1 gives, for each $i \in \{1,2\}$,*

$$\Pr\big[\mathsf{EUF\text{-}CMA}(\mathcal{A}_S,\mathcal{D}_S,Sch_1,Sch_2) : res\big] \;\le\; \frac{1}{q} + \Pr\big[\mathsf{EUF\text{-}CMA}_{\Sigma_i}(\mathcal{B}_i(\mathcal{A}_S),\mathcal{D}_S,Sch_i) : res\big] + \Pr\big[\mathsf{CR}(\mathcal{B}_D,\mathcal{D}_S) : res\big]. \quad (1)$$

For this instantiation, all losslessness obligations of the Schnorr component procedures are discharged in EasyCrypt. The losslessness of $\mathcal{A}_S$ and $\mathcal{D}_S$, as well as the component EUF-CMA and digest collision-resistance assumptions appearing in the bound, remain hypotheses.

Schnorr-Okamoto corollary. Instantiating Theorem 1 with $C_1 = \mathsf{SchnorrComp}_1$ and $C_2 = \mathsf{OkamotoComp}_2$ discharges all component losslessness assumptions concretely; the adversary and digest assumptions remain as hypotheses. This instantiation re-establishes Theorem 1 via the identical nine-lemma proof script of Section 4.1–no lemma statement or proof tactic differs from the Schnorr-Schnorr development; only the concrete component instantiation and its losslessness side-conditions differ from Corollary 1. This confirms that the nine-lemma argument is genuinely parametric rather than tailored to the Schnorr case.

Corollary 2 (Schnorr-Okamoto bound). *For every adversary $\mathcal{A}_S$ and digest $\mathcal{D}_S$, instantiate $C_1 = Sch_1 = \mathsf{SchnorrComp}_1$ and $C_2 = Ok_2 = \mathsf{OkamotoComp}_2$, and letq $= |\mathcal{R}|$. Theorem 1 gives, for each $i \in \{1,2\}$,*

$$\Pr[\mathsf{EUF\text{-}CMA}(\mathcal{A}_S,\mathcal{D}_S,\mathsf{Sch}_1,\mathsf{Ok}_2) : \mathsf{res}] \le \frac{1}{q} + \Pr[\mathsf{EUF\text{-}CMA}_{\Sigma_i}(\mathcal{B}_i(\mathcal{A}_S),\mathcal{D}_S,C_i) : \mathsf{res}] + \Pr[\mathsf{CR}(\mathcal{B}_D,\mathcal{D}_S) : \mathsf{res}]. \quad (2)$$

5 Conclusion and future work

We presented, to the best of our knowledge, the first machine-checked EUF-CMA security proof of the hybrid Fiat-Shamir (FS-FS) signature construction of Bindel and Hale [12]. The construction is formalised in EasyCrypt in the ROM, with the two component signatures represented through abstract interfaces capturing the operations required by the security proof. Theorem 1 provides two symmetric security bounds, each reducing the security of the hybrid to the EUF-CMA security of either component under the shared hybrid-hash challenge, together with collision resistance of the message digest. The proof exposes three possible branches for a successful hybrid forgery. A forgery for which the verification point has not been queried to the random oracle is bounded by the probability of guessing the challenge, namely $1/|\mathcal{R}|$. If the verification point was queried and the forgery is accompanied by a digest collision, the event is reduced to collision resistance of the abstract digest $\mathcal{D}$. In the remaining good-world branch, the hybrid forgery is transformed into a forgery against the selected component EUF-CMA game. Since $\mathcal{H}$ is modelled as a random oracle, the proof obtains the corresponding guessing bound directly from the unpredictability of the random-oracle output at an unqueried verification point. Consequently, the resulting theorem does not require the separate second-preimage-resistance assumption on $\mathcal{H}$ used in the security argument of Bindel and Hale [12]. The digest $\mathcal{D}$, in contrast, is kept abstract and is not instantiated as a random oracle; its role therefore remains captured by an explicit collision-resistance assumption.

Beyond the resulting security bound, the mechanisation makes explicit proof obligations that can remain implicit in a pen-and-paper game-hopping argument. In particular, the ROM part of the proof requires a logging invariant relating the lazy random oracle to its query history. The collision branch requires a module-restriction invariant ensuring that the abstract digest cannot alter the state maintained by the digest-collision tracker. These obligations illustrate reusable proof patterns for EasyCrypt developments in which lazy-oracle state interacts with auxiliary state maintained by a game or reduction.

The concrete instantiations further demonstrate the separation between the abstract security argument and the representation of the component signatures. The Schnorr-Schnorr instantiation provides a uniform setting in which the required component losslessness obligations are discharged concretely. The Schnorr-Okamoto instantiation changes the internal representation of one component: Okamoto uses paired secret keys, commitment randomness, and corresponding responses rather than the scalar structures used for Schnorr. Nevertheless, the abstract game-hopping argument and its lemma statements remain unchanged; the additional work is confined to providing the appropriate concrete component interface and proving the associated losslessness obligations. These instantiations therefore provide evidence that the formal security argument is not tied to a single algebraic representation, while remaining within the class of component interfaces supported by the present formalisation.

Future work. The primary direction is to extend the formalisation to the quantum random-oracle model (QROM) [3,14,16,22]. The existing EasyPQC framework [3] is under development. Our present proof relies on a classical logging wrapper to determine whether the adversary queried the verification point $x_V = (w_1, w_2, \mathcal{D}(m))$. This query-logging mechanism does not directly extend to quantum superposition queries. A QROM formalisation would therefore require a different proof mechanism, such as a compressed oracle [28], one-way-to-hiding lemma [1], or a measure-and-reprogram technique [15], adapted to the shared verification structure of the FS-FS construction. A second direction is to extend the component interface to Fiat-Shamir schemes with abort or rejection behaviour. This would require explicitly modelling the corresponding failure behaviour and adapting the game sequence and losslessness arguments. A lattice-based Fiat-Shamir-with-aborts scheme such as Dilithium [18] would then provide a useful cross-family instantiation for assessing whether the proof patterns transfer beyond the discrete-logarithm-based components considered here. A further direction is to formalise the remaining true hybrid constructions identified by Bindel and Hale [12], including FS-RSA, Falcon-RSA, and FS-DSA. Finally, deriving concrete bounds that explicitly account for adversarial query counts would connect the asymptotic theorem more directly to concrete security estimation and parameter selection for hybrid deployments.

Acknowledgements The author thanks Dr. Nina Bindel, Dr. Long Meng, Dr. Carmine Abate, and Dr. Denis Firsov for their helpful comments and feedback on drafts of this paper and its formalisation.

References

1. Ambainis, A., Hamburg, M., Unruh, D.: Quantum security proofs using semi-classical oracles. In: Annual International Cryptology Conference. pp. 269–295. Springer (2019)
2. Barbosa, M., Barthe, G., Doczkal, C., Don, J., Fehr, S., Grégoire, B., Huang, Y.H., Hülsing, A., Lee, Y., Wu, X.: Fixing and mechanizing the security proof of Fiat-Shamir with aborts and Dilithium. In: Annual International Cryptology Conference. pp. 358–389. Springer (2023)
3. Barbosa, M., Barthe, G., Fan, X., Grégoire, B., Hung, S.H., Katz, J., Strub, P.Y., Wu, X., Zhou, L.: EasyPQC: Verifying post-quantum cryptography. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 2564–2586 (2021)
4. Barbosa, M., Dupressoir, F., Hülsing, A., Meijers, M., Strub, P.Y.: A tight security proof for SPHINCS+, formally verified. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 35–67. Springer (2024)
5. Barthe, G., Crespo, J.M., Kunz, C.: Beyond 2-safety: Asymmetric product programs for relational program verification. In: International Symposium on Logical Foundations of Computer Science. pp. 29–43. Springer (2013)
6. Barthe, G., Dupressoir, F., Grégoire, B., Kunz, C., Schmidt, B., Strub, P.Y.: Easycrypt: A tutorial. International School on Foundations of Security Analysis and Design pp. 146–166 (2012)

7. Barthe, G., Grégoire, B., Heraud, S., Béguelin, S.Z.: Computer-aided security proofs for the working cryptographer. In: Annual Cryptology Conference. pp. 71–90. Springer (2011)
8. Barthe, G., Grégoire, B., Zanella Béguelin, S.: Formal certification of code-based cryptographic proofs. In: Proceedings of the 36th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages. pp. 90–101 (2009)
9. Becker, A., Guthrie, R., Jenkins, M.: Non-composite hybrid authentication in PKIX and applications to internet protocols. Internet-Draft, IETF (2022), https://datatracker.ietf.org/doc/draft-becker-guthrie-noncomposite-hybrid-auth/
10. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Proceedings of the 1st ACM Conference on Computer and Communications Security. pp. 62–73 (1993)
11. Bindel, N., Brendel, J., Fischlin, M., Goncalves, B., Stebila, D.: Hybrid key encapsulation mechanisms and authenticated key exchange. In: International Conference on Post-Quantum Cryptography. pp. 206–226. Springer (2019)
12. Bindel, N., Hale, B.: A note on hybrid signature schemes. Cryptology ePrint Archive (2023)
13. Bindel, N., Herath, U., McKague, M., Stebila, D.: Transitioning to a quantum-resistant public key infrastructure. In: International Workshop on Post-Quantum Cryptography. pp. 384–405. Springer (2017)
14. Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C., Zhandry, M.: Random oracles in a quantum world. In: International conference on the theory and application of cryptology and information security. pp. 41–69. Springer (2011)
15. Don, J., Fehr, S., Majenz, C.: The measure-and-reprogram technique 2.0: Multi-round Fiat-Shamir and more. In: Annual International Cryptology Conference. pp. 602–631. Springer (2020)
16. Don, J., Fehr, S., Majenz, C., Schaffner, C.: Security of the Fiat-Shamir transformation in the quantum random-oracle model. In: Annual International Cryptology Conference. pp. 356–383. Springer (2019)
17. Driscoll, F., Parsons, M., Hale, B.: Terminology for post-quantum traditional hybrid schemes. RFC 9794, IETF (Jun 2025), https://www.rfc-editor.org/rfc/rfc9794
18. Ducas, L., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: Crystals–dilithium: Digital signatures from module lattices (2018)
19. Dupressoir, F., Zain, S.: Machine-checking unforgeability proofs for signature schemes with tight reductions to the computational diffie-hellman problem. In: 2021 IEEE 34th Computer Security Foundations Symposium (CSF). pp. 1–15. IEEE (2021)
20. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Conference on the theory and application of cryptographic techniques. pp. 186–194. Springer (1986)
21. Hülsing, A., Meijers, M., Strub, P.Y.: Formal verification of saber's public-key encryption scheme in easycrypt. In: Annual International Cryptology Conference. pp. 622–653. Springer (2022)
22. Kiltz, E., Lyubashevsky, V., Schaffner, C.: A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 552–586. Springer (2018)

23. National Institute of Standards and Technology: Module-lattice-based digital signature standard. Tech. Rep. FIPS 204, NIST (2024), https://doi.org/10.6028/NIST.FIPS.204
24. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. Journal of cryptology **13**(3), 361–396 (2000)
25. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs. cryptology eprint archive (2004)
26. Zain, S.: Machine-checked verification of digital signature schemes in EasyCrypt. Ph.D. thesis, University of Bristol (2023)
27. Zanella-Béguelin, S., Barthe, G., Grégoire, B., Olmedo, F.: Formally certifying the security of digital signature schemes. In: 2009 30th IEEE Symposium on Security and Privacy. pp. 237–250. IEEE (2009)
28. Zhandry, M.: How to record quantum queries, and applications to quantum indifferentiability. In: Annual International Cryptology Conference. pp. 239–268. Springer (2019)

Appendix: Supplementary Material

Mechanisation Effort. An early attempt to adapt the concatenation-based framework of [13]—where each component is hashed separately, yielding two independent oracle queries—revealed a structural mismatch: the FS-FS construction of [12] commits both components through a *single* shared oracle query. This invalidates the per-component oracle-freshness argument of [13] and demands a new logging structure that tracks the joint hash input x_V across both signing paths. Redesigning the game-hopping around this shared structure, while keeping the shared oracle consistent across reductions, was the first major proof effort.

A second effort focused on structural abstraction. An initial concrete instantiation with the two Schnorr components was generalized to abstract module types Σ_1 and Σ_2, making Theorem 1 parametric over any conforming FS-based pair. The nine-lemma proof structure was preserved with only localized modifications, while the concrete Schnorr instantiation is retained as Corollary 1.

A third effort introduced the Schnorr-Okamoto instantiation (Corollary 2). Since Theorem 1 is already parametric over C_1 and C_2, this required no changes to the proof skeleton. Only the concrete module $\mathsf{OkamotoComp}_2$ and its four losslessness lemmas were added, demonstrating that the abstraction extends to a component with a structurally different paired key and response type.

Complete Reduction Chain. Table 1 details the game-hopping proof of Theorem 1. Parenthesised names denote the corresponding EasyCrypt lemmas.

L.	Statement	Key idea
1	$\Pr[G_0 : res] = \Pr[G_1 : res]$ (`EUF_ROM_GameQ_equiv`)	Inline `verify`; purely structural
2	$\Pr[G_1 : res] = \Pr[G_2 : res]$ (`GameQ_GameQH_equiv`)	Replace LRO with logging wrapper LH; **byequiv**
3	$\Pr[G_2 : res] = \Pr[G_2 : res \wedge badGuess] + \Pr[G_2 : res \wedge \neg badGuess]$ (`GameQH_split`)	Law of total probability: $\{res \wedge badGuess\}$ and $\{res \wedge \neg badGuess\}$ are mutually exclusive and jointly exhaustive within $\{res\}$; `Pr[mu_split badGuess]`
4	$\Pr[G_2 : res \wedge badGuess] \leq 1/q$ (`GameQH_guess_bound1`)	Invariant Inv (P1) gives $\mathsf{HRO.RO.m}[x_V]{=}\mathsf{None}$ under $badGuess$; auxiliary G_{aux} exposes fresh c' (P2); **byphoare** + `dp1E`
5	$\Pr[G_0 : res] \leq \Pr[G_2 : res \wedge \neg badGuess] + 1/q$ (`EUF_leq_GameQH`)	Combine L. 1–4
6	$\Pr[G_2 : res \wedge \neg badGuess] = \Pr[G_3 : res \wedge \neg badGuess]$ (`GameQH_GameQHD_equiv_res_badGuess`)	Introduce $\mathcal{T}_D{=}\mathsf{TrackD}(D)$; $\mathcal{T}_D$ observes but never alters output; **byequiv** conditioned on $\neg badGuess$
7	$\Pr[G_3 : res \wedge \neg badGuess] \leq \Pr[G_3 : res \wedge \neg badGuess \wedge \neg badD] + \Pr[G_3 : badD]$ (`GameQHD_good_split_badD`)	Total probability on $badD$ (equality via `Pr[mu_split badD]`), then monotonicity $\{res \wedge \neg badGuess \wedge badD\} \subseteq \{badD\}$ via `Pr[mu_sub]`
8	$\Pr[G_3 : badD] \leq \Pr[\mathsf{CR}(\mathcal{B}_D, \mathcal{D}) : res]$ (`badD_bound`)	(i) Drop LH: badD unaffected. (ii) $\mathsf{GameQD} \equiv \mathcal{B}_D$ on badD. (iii) Soundness: collision invariant + module restriction $\{-\mathsf{TrackD}\}$ on $\mathcal{D}$ (P3)
9	$\Pr[G_3 : res \wedge \neg\mathsf{badGuess} \wedge \neg\mathsf{badD}] \leq \Pr[\mathsf{EUF\text{-}CMA}_{\Sigma_i}(\mathcal{B}_i(\mathcal{A}), \mathcal{D}, \mathbf{C}_i) : res]$, $i \in \{1,2\}$ (`Hybrid_success_implies_Comp1`); (`Hybrid_success_implies_Comp2`)	Remove TD via $\neg\mathsf{badD}$; $G_3 \equiv G_2$ conditionally, then $G_2 \equiv G_1$; hybrid forgery (c, z_1, z_2) yields (w_j, c, z_i) as a valid Σ_i forgery; $\mathcal{B}_i$ supplies correct $w_j{=}\mathsf{recon}_j(pk_j, c, z_j)$

Assembly. The Σ_1 branch combines L. 5 with L. 6–9 ($i{=}1$); the Σ_2 branch combines L. 5 with L. 6–9 ($i{=}2$). $\mathcal{B}_1{=}$`Red_To_Comp1`; $\mathcal{B}_2{=}$`Red_To_Comp2`.

Final bound (Theorem 1). For $i \in \{1,2\}$: $\Pr[\mathsf{EUF\text{-}CMA}_{\Sigma_h}(\mathcal{A}, \mathcal{D}, \mathbf{C}_1, \mathbf{C}_2) : res] \leq \frac{1}{|\mathcal{R}|} + \Pr[\mathsf{EUF\text{-}CMA}_{\Sigma_i}(\mathcal{B}_i(\mathcal{A}), \mathcal{D}, \mathbf{C}_i) : res] + \Pr[\mathsf{CR}(\mathcal{B}_D(\mathcal{A}, \mathcal{D}, \mathbf{C}_1, \mathbf{C}_2), \mathcal{D}) : res]$

Table 1. $G_0{=}\mathsf{EUF_CMA}$, $G_1{=}\mathsf{GameQ}$, $G_2{=}\mathsf{GameQH}$, $G_3{=}\mathsf{GameQHD}$; $|\mathcal{R}|$ is the challenge-space size. Security of Σ_h reduces to (*i*) $\mathsf{EUF\text{-}CMA}$ of Σ_i ($i \in \{1,2\}$) under the shared hybrid-hash challenge (L. 9), (*ii*) collision resistance of $\mathcal{D}$ (L. 8), and (*iii*) the ROM guessing term $1/|\mathcal{R}|$ (L. 4), which subsumes the SPR assumption of [12].