Salted Fiat–Shamir at the Edge: A Modular Multi-User Security Analysis

Anonymous Authors

Anonymous Institution
contact@anonymous.invalid

Abstract. The Fiat–Shamir transform removes interaction from public-coin proofs by deriving the verifier's challenge from a hash function. In a multi-user deployment, however, the same random oracle serves many public keys and many concurrent statements. This makes domain separation and the accounting of oracle queries part of the security argument rather than mere implementation details. We give a compact modular analysis of a *salted* Fiat–Shamir transform in the classical random-oracle model. The transform hashes an explicit protocol label, public key, statement, commitment, and fresh salt. Its security reduces to three transparent quantities: the interactive protocol's special-soundness error, a salt-collision term, and the probability of guessing a programmed oracle point. The proof isolates the exact place at which the number of users enters and avoids a multiplicative loss in that number when public keys are included in the hash domain. We instantiate the framework with Schnorr identification and derive concrete parameter guidance. Our presentation is deliberately audit-oriented: all encodings are typed, the games expose their bad events, and the implementation checklist follows directly from the proof assumptions.

Keywords: Fiat–Shamir transform · random-oracle model · multi-user security · domain separation · Schnorr signatures

1 Introduction

The Fiat–Shamir (FS) transform [4] is among the most useful interfaces between interactive proofs and deployed cryptography. It replaces the verifier's random challenge with a hash of the public transcript, thereby turning a three-move identification protocol into a non-interactive proof or a signature scheme. The random-oracle methodology [2] provides a clean idealized setting for its analysis, while the forking lemma [6] explains how two accepting transcripts with a shared commitment can reveal a witness.

Real deployments differ from the single-key textbook experiment. A shared hash function is evaluated for many protocols, public keys, sessions, and messages. If these values are not represented unambiguously in the oracle input, transcripts from one context can be replayed or reinterpreted in another. Even with a sound encoding, a proof must say whether its loss grows with the number of users

Table 1. Where the proof accounts for deployment-level concerns.

Concern	Mechanism	Proof location
Cross-protocol reuse	Fixed protocol tag	Domain lemma, Lemma 1
Cross-user replay	Public key in hash input	Domain lemma, Lemma 1
Ambiguous parsing	Injective typed encoding	Syntax, Section 2.3
Repeated programmed point	Fresh ρ-bit salt	Collision bound, Lemma 2
Witness extraction	Special soundness and rewind	Main theorem, Theorem 1

and how fresh randomness affects reprogramming. Such questions are especially relevant when a conservative proof is converted into parameters.

This paper studies the following deliberately simple transform. For a three-move public-coin protocol with commitment a, the prover samples a ρ-bit salt s and sets

$$c = \mathsf{H}(\mathtt{tag} \,\|\, \mathsf{enc}(pk, x, a, s)),$$

where the encoding is injective and length-delimited. The proof is $\pi = (a, s, z)$, with z the response to c. Including pk gives each user a disjoint logical oracle domain; including s makes the point selected by the honest prover hard to predict before the commitment is fixed.

1.1 Contributions

Our contribution is not a new assumption or identification protocol. It is a small, reusable proof layer connecting standard special soundness to a multi-user deployment.

- We specify a typed salted FS transform whose transcript encoding is explicit down to the protocol tag and length prefixes.
- We present a game-based extraction theorem in which the user count is absorbed by domain separation. The remaining loss is expressed using total oracle and signing queries, rather than a union bound over keys.
- We instantiate the theorem for Schnorr identification and turn its bad events into a concise parameter table and an implementation checklist.

Scope. We work in the classical random-oracle model (ROM). Quantum queries require different reprogramming tools and are not covered by our main theorem; we return to this limitation in Section 6. Our theorem is stated for proofs of knowledge. The standard conversion to signatures hashes the message as the statement and uses the same analysis.

2 Preliminaries

For $n \in \mathbb{N}$, $[n] = \{1, \ldots, n\}$ and $u \xleftarrow{\$} S$ denotes uniform sampling from a finite set S. All algorithms receive the security parameter 1^λ implicitly. A function f is negligible if for every positive polynomial p, $f(\lambda) < 1/p(\lambda)$ for all sufficiently large λ. We write $\mathrm{Time}(\mathcal{A})$ for the running time of an adversary, including its oracle simulation overhead.

2.1 Three-move public-coin protocols

Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be an NP relation, with instance x and witness w. A canonical three-move protocol $\Pi = (\mathsf{Prove}, \mathsf{Vfy})$ has the form

$$(a, st) \leftarrow \mathsf{Prove}_1(pk, x, w), \qquad c \xleftarrow{\$} \mathcal{C}, \qquad z \leftarrow \mathsf{Prove}_2(st, c),$$

after which the verifier outputs $\mathsf{Vfy}(pk, x, a, c, z) \in \{\mathsf{acc}, \mathsf{rej}\}$. The challenge set $\mathcal{C}$ has size 2^t. Perfect completeness means that honest transcripts are always accepted.

Definition 1 (Special soundness). *The protocol Π has special-soundness error $\varepsilon_{\mathrm{ss}}$ if there is an extractor Ext such that, given two accepting transcripts (a, c, z) and (a, c', z') for the same (pk, x) with $c \neq c'$, it outputs w satisfying $(x, w) \in R$, except with probability at most $\varepsilon_{\mathrm{ss}}$.*

For Schnorr identification in a prime-order group, extraction is algebraic and $\varepsilon_{\mathrm{ss}} = 0$. For protocols with an admissibility condition or computational extraction, keeping the error explicit is useful.

2.2 Random oracles and lazy sampling

A random oracle $\mathsf{H} : \{0,1\}^* \to \mathcal{C}$ is simulated by a table T. On a new input X, the simulator samples $c \xleftarrow{\$} \mathcal{C}$, stores $T[X] = c$, and returns c; repeated inputs return the stored value. Programming $\mathsf{H}(X) := c$ is distribution-preserving when X has not previously been queried. Consequently, our games record the event

$$\mathsf{PreQ} := [X \in \mathrm{dom}(T)]$$

whenever an honest transcript attempts to program X.

2.3 Typed encodings

We require a canonical injective encoding. For a byte string v, let $L(v)$ be its fixed-width length followed by v, and define

$$\mathsf{Enc}(pk, x, a, s) = \mathtt{FSv1} \,\|\, L(\mathsf{id}(\Pi)) \,\|\, L(pk) \,\|\, L(x) \,\|\, L(a) \,\|\, L(s). \tag{1}$$

The literal version byte prevents a future format from silently sharing the same domain. Fixed-length encodings may omit a length prefix only when the type and length are uniquely determined by the protocol identifier.

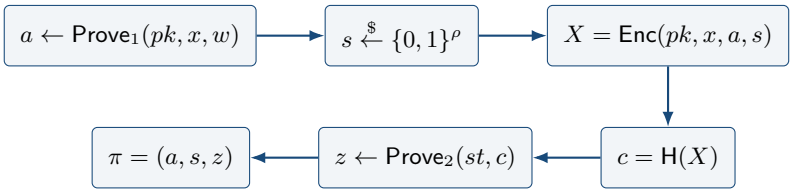

Fig. 1. Data flow of the salted transform. Every value entering the random oracle is public, typed, and bound to the public key.

Lemma 1 (Disjoint logical domains). *If* Enc *is injective, two equal oracle inputs identify the same protocol version, public key, instance, commitment, and salt. In particular, an oracle query under one public key cannot be the programmed point of a different public key.*

Proof. Parse the fixed tag and then each length-delimited field from left to right. Equality of the resulting strings implies equality of every field. The last claim follows by contraposition on the public-key field.

3 The Salted Transform

Let Π be a protocol as above and let ρ be a salt length. The salted transform $\mathsf{Trans}^{\mathsf{H}}_{\Pi,\rho}$ is defined by the following algorithms.

Prover $\mathsf{Prove}^{\mathsf{H}}$**.** On input (pk, x, w), compute $(a, st) \leftarrow \mathsf{Prove}_1(pk, x, w)$, sample $s \xleftarrow{\$} \{0,1\}^\rho$, set $c = \mathsf{H}(\mathsf{Enc}(pk, x, a, s))$, compute $z \leftarrow \mathsf{Prove}_2(st, c)$, and return $\pi = (a, s, z)$.

Verifier $\mathsf{Vfy}^{\mathsf{H}}$**.** On input (pk, x, π), parse $\pi = (a, s, z)$ and reject unless $s \in \{0,1\}^\rho$. Set $c = \mathsf{H}(\mathsf{Enc}(pk, x, a, s))$ and return $\mathsf{Vfy}(pk, x, a, c, z)$.

The salt is public and carries no entropy after publication. Its role is to make the future programming point unpredictable when an honest transcript is simulated. It does not replace commitment randomness, and it is never used as a secret key.

3.1 Multi-user knowledge experiment

The challenger generates keys $(pk_i, sk_i) \leftarrow \mathsf{KGen}(1^\lambda)$ for $i \in [U]$ and gives all public keys to $\mathcal{A}$. The adversary may query the common oracle H at most q_H times and request at most q_P honest proofs in total, adaptively choosing (i, x) for each request. It wins by returning (i^*, x^*, π^*) such that verification accepts, the tuple is fresh, and the designated extractor fails to output a witness for x^*. Freshness means that (i^*, x^*, π^*) was not returned verbatim by the proof oracle; an application may impose a stronger, message-level notion.

We denote the winning probability by $\mathrm{Adv}^{\mathrm{muk}}_{\mathsf{Trans},\mathcal{A}}(\lambda, U, q_H, q_P)$. The experiment uses one global lazy-sampled table. This matches deployed hash functions

and ensures that any accounting benefit comes from the encoding, not from granting each user an independent oracle.

4 Security Analysis

The proof has two independent components. First, salts control collisions among points that the simulator chooses. Second, a forking step obtains a second challenge for the adversary's decisive oracle query.

Lemma 2 (Salt-collision bound). *Among at most q_P honest proof queries, the probability that two queries with identical (pk, x, a) also select the same salt is at most*

$$\delta_{\mathrm{salt}} \le \frac{q_P(q_P-1)}{2^{\rho+1}}.$$

The same bound holds if the adversary adaptively selects the public key and instance before each salt is sampled.

Proof. Condition on the entire execution before a salt is sampled. For each pair of proof queries with matching prefixes (pk, x, a), the later independent salt equals the earlier one with probability $2^{-\rho}$. There are at most $\binom{q_P}{2}$ pairs. Adaptivity changes which pairs have matching prefixes but not the conditional probability for their fresh salts.

Lemma 3 (Pre-query bound). *Suppose an honest proof's commitment a is fixed before its salt is sampled. For any adversary making at most q_H oracle queries overall, the probability that its table already contains $\mathsf{Enc}(pk, x, a, s)$ when a fresh salt s is sampled is at most $q_H/2^{\rho}$ per proof query. Thus $\Pr[\mathsf{PreQ}] \le q_P q_H/2^{\rho}$.*

Proof. After conditioning on (pk, x, a) and the adversary's current view, at most q_H table entries can have the required prefix. Each determines at most one bad ρ-bit salt by injectivity of the encoding. A union bound over the q_P proof queries completes the argument.

To state the extraction term cleanly, let ε be the adversary's success probability after excluding the two bad events above. A standard forking experiment reruns $\mathcal{A}$ with identical random tape and identical oracle answers before its decisive query, then supplies an independent challenge at that query. Write $q = q_H + q_P + 1$ for an upper bound on the number of candidate transcript-defining oracle points, including those created while answering proof queries.

Theorem 1 (Multi-user extraction). *Let Π have challenge space of size 2^t and special-soundness error $\varepsilon_{\mathrm{ss}}$. For every classical adversary $\mathcal{A}$ against $\mathsf{Trans}^{\mathsf{H}}_{\Pi,\rho}$, there is a black-box extractor $\mathcal{B}$ such that*

$$\mathrm{Adv}^{\mathrm{muk}}_{\mathsf{Trans},\mathcal{A}} \le \sqrt{q\big(\mathrm{Adv}^{\mathrm{ext}}_{\Pi,\mathcal{B}} + 2^{-t} + \varepsilon_{\mathrm{ss}}\big)} + \frac{q_P q_H}{2^{\rho}} + \frac{q_P(q_P-1)}{2^{\rho+1}}, \tag{2}$$

where $\mathrm{Time}(\mathcal{B})$ is at most two executions of $\mathcal{A}$ plus $O(q)$ table operations and one invocation of the special-soundness extractor. The bound has no additional multiplicative factor U.

Proof. We proceed through games.

Game G_0. This is the real multi-user experiment with one global lazy-sampled oracle.

Game G_1. The challenger answers honest proof requests by choosing the salt and programming the transcript-defining oracle point consistently. If that point was previously queried, it sets PreQ and aborts. By Lemma 3, the statistical difference from G_0 is at most $q_P q_H / 2^{\rho}$.

Game G_2. Abort when two simulated proofs reuse an oracle point. By Lemma 2, this costs at most $q_P(q_P - 1)/2^{\rho+1}$. Conditioned on no abort, every programmed point is fresh and has exactly the random-oracle distribution.

Now fix a successful execution of G_2. Except with probability 2^{-t}, the adversary queried the oracle at $X^* = \mathsf{Enc}(pk_{i^*}, x^*, a^*, s^*)$ before outputting its proof: without that query, its response predicts a uniform challenge. Select the decisive query among at most q candidates and apply the classical forking argument of Pointcheval and Stern [6]. Replaying with the same prefix and an independent answer yields, with probability at least $\varepsilon^2/q - 2^{-t}$, two accepting transcripts sharing (pk_{i^*}, x^*, a^*) and having distinct challenges. Applying special soundness extracts a witness, apart from $\varepsilon_{\mathrm{ss}}$.

The public key is part of X^*. Hence the fork stays in the same user's domain by Lemma 1; the extractor neither guesses i^* in advance nor loses a factor U. Rearranging the resulting inequality for ε and adding the two game-transition terms proves Equation (2).

Remark 1 (What "no factor U" means). The adversary's resources q_H and q_P may naturally grow with the number of users. The theorem does not make a system with more traffic free. It states that after total traffic is fixed, the reduction does not pay a second loss merely to guess which public key appears in the forgery.

5 Schnorr Instantiation

Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order p, written multiplicatively. A secret key is $y \xleftarrow{\$} \mathbb{Z}_p$ and the public key is $Y = g^y$. To prove knowledge of y, the prover samples $r \xleftarrow{\$} \mathbb{Z}_p$, sends $A = g^r$, receives $c \in \mathbb{Z}_p$, and returns $z = r + cy \bmod p$. Verification checks

$$g^z \stackrel{?}{=} A\,Y^c. \tag{3}$$

Given accepting transcripts (A, c, z) and (A, c', z') with $c \neq c'$, the witness is

$$y = (z - z')(c - c')^{-1} \bmod p. \tag{4}$$

Thus special soundness is perfect. For signatures, set $x = \mathsf{Digest}(m)$ and include the complete, canonical public-key encoding in Equation (1). The response remains one scalar, and the transform adds exactly ρ salt bits to an ordinary Schnorr signature.

Table 2. Upper bounds on salt-related bad events for $q_H = 2^{40}$.

Salt ρ	Proof queries q_P	Pre-query bound	Collision bound
128	2^{20}	2^{-68}	$< 2^{-89}$
128	2^{32}	2^{-56}	$< 2^{-65}$
192	2^{32}	2^{-120}	$< 2^{-129}$
192	2^{48}	2^{-104}	$< 2^{-97}$
256	2^{64}	2^{-152}	$< 2^{-129}$

5.1 Concrete accounting

The salt terms are information-theoretic and easy to size independently of the group assumption. Table 2 reports conservative upper bounds for $q_H = 2^{40}$ total hash queries and several proof-query budgets. Values are computed directly from

$$\delta_{\text{salt}} + \delta_{\text{pre}} \leq \frac{q_P(q_P - 1)}{2^{\rho+1}} + \frac{q_P q_H}{2^{\rho}}.$$

They are bounds, not measured failure rates.

With a 256-bit challenge, the guessing term 2^{-t} is dominated by either line of the table. A 128-bit salt is adequate for many bounded deployments, but a 192-bit salt gives more comfortable separation when the lifetime proof count is uncertain. The right choice should follow the system-wide query budget, not the security level of a single key in isolation.

5.2 Implementation profile

The proof assumptions translate into the following interoperable profile.

1. Serialize group elements canonically and reject non-canonical or invalid encodings before hashing or verification.
2. Use a fixed ASCII protocol identifier and version byte. Do not allow callers to select the tag dynamically.
3. Encode all variable-length fields with fixed-width lengths, and bind the public key even when it is available from surrounding context.
4. Draw each salt from the operating system's cryptographic random-number generator. Treat RNG failure as a signing failure.
5. Reduce the hash output to $\mathbb{Z}_p$ with a specified unbiased method, such as rejection sampling or a standard hash-to-field procedure.
6. Verify lengths, group membership, and the equation in Equation (3) before returning success.

These checks are not cosmetic. Non-canonical group encodings, for example, can violate injectivity even when the high-level tuple in Equation (1) looks unambiguous.

6 Discussion and Limitations

Classical versus quantum access. The rewind in Theorem 1 observes and reprograms a classical query. An adversary with superposition access to the oracle cannot be handled by that argument. QROM analyses use techniques such as measure-and-reprogram [3] or transforms designed for quantum extraction [8]. Our syntax remains useful in that model, but the theorem and its concrete loss must be replaced.

State and randomness. Salting reduces the probability that a simulator encounters a pre-queried point. It does not repair biased commitment randomness, secret-dependent timing, nonce reuse, or state rollback. In the Schnorr instantiation, reusing r across two challenges reveals the secret key regardless of the salt.

Proof tightness. The square-root loss in Equation (2) is the familiar cost of a generic classical fork. Algebraic assumptions, an algebraic-group model, or a protocol-specific reduction can yield better concrete bounds. Our goal is modularity: the salt and domain-separation terms can be reused when a sharper extraction lemma is available.

Chosen-key settings. The experiment uses honestly generated public keys. If adversaries may register malformed or related keys, the system also needs proof of possession or a key-validation rule. Binding pk into the hash prevents transcript confusion but does not establish that the registrant knows the corresponding secret.

7 Related Work

Fiat and Shamir introduced the heuristic transformation from identification to signatures [4]. Bellare and Rogaway formalized the random oracle as a design and analysis methodology [2], while Pointcheval and Stern developed the forking technique for security proofs of signature schemes [6]. Schnorr's discrete-logarithm identification and signature construction is the canonical algebraic example of the transform [7].

Later work sharpened the understanding of Fiat–Shamir in both classical and quantum settings. Abdalla, An, Bellare, and Namprempre studied transformations and security notions for identification schemes [1]. Unruh gave a quantum-resistant transform for proofs of knowledge [8], and Don, Fehr, Majenz, and Schaffner developed a measure-and-reprogram technique for QROM proofs [3]. Kiltz, Lyubashevsky, and Schaffner analyzed the concrete security of Fiat–Shamir signatures under active attacks [5]. Our narrower focus is the transparent separation of multi-user encoding and salt-related bad events from whichever extraction result is plugged into the final step.

8 Conclusion

For a shared random oracle, domain separation is a security boundary. By hashing a protocol tag, public key, statement, commitment, and fresh salt using an injective encoding, the salted Fiat–Shamir transform admits a modular multi-user analysis. The public-key field prevents an avoidable user-guessing loss; the salt exposes two elementary bad-event bounds; and special soundness handles the remaining extraction step. The resulting theorem is intentionally simple enough to audit and to specialize. Its main practical lesson is equally simple: specify the bytes that are hashed, size randomness against system-wide traffic, and make every proof assumption visible at the API boundary.

References

1. Abdalla, M., An, J.H., Bellare, M., Namprempre, C.: From identification to signatures via the Fiat–Shamir transform: Minimizing assumptions for security and forward-security. In: EUROCRYPT 2002. LNCS, vol. 2332, pp. 418–433. Springer (2002)
2. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM CCS 1993. pp. 62–73. ACM (1993)
3. Don, J., Fehr, S., Majenz, C., Schaffner, C.: Security of the Fiat–Shamir transformation in the quantum random-oracle model. In: CRYPTO 2019, Part II. LNCS, vol. 11693, pp. 356–383. Springer (2019)
4. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer (1987)
5. Kiltz, E., Lyubashevsky, V., Schaffner, C.: A concrete treatment of Fiat–Shamir signatures in the quantum random-oracle model. In: EUROCRYPT 2018, Part III. LNCS, vol. 10822, pp. 552–586. Springer (2018)
6. Pointcheval, D., Stern, J.: Security proofs for signature schemes. In: EUROCRYPT 1996. LNCS, vol. 1070, pp. 387–398. Springer (1996)
7. Schnorr, C.P.: Efficient signature generation by smart cards. Journal of Cryptology 4(3), 161–174 (1991)
8. Unruh, D.: Non-interactive zero-knowledge proofs in the quantum random oracle model. In: EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 755–784. Springer (2015)

A Game Definitions

For completeness, we spell out the two abort flags used in the proof. The simulator stores tuples (X, c, origin) in its oracle table, where $\mathsf{origin} \in \{\mathsf{adversary}, \mathsf{proof}\}$. Before inserting a proof-origin entry $X = \mathsf{Enc}(pk, x, a, s)$, it executes

$$\mathsf{PreQ} \leftarrow \mathsf{PreQ} \vee [X \in \mathrm{dom}(T)].$$

After insertion, it sets $\mathsf{SaltColl}$ if a different proof request has the same X. The proof of Theorem 1 aborts on either flag. Keeping the flags separate matters

because they rely on different independence claims: the pre-query argument conditions on the adversary's current table, whereas the collision argument compares pairs of simulator-generated salts.

B Extraction Algebra for Schnorr

Suppose both $g^z = AY^c$ and $g^{z'} = AY^{c'}$ hold. Dividing the equations gives $g^{z-z'} = Y^{c-c'} = g^{y(c-c')}$. Since p is prime and $c \neq c'$, the element $c - c'$ has an inverse in $\mathbb{Z}_p$. Therefore $y = (z - z')(c - c')^{-1} \bmod p$, establishing Equation (4). A verifier must first check that $A, Y \in \mathbb{G}$; otherwise the cancellation can take place in an unintended subgroup and the extraction statement no longer matches the claimed relation.