

Non-Interactive Atomic Swaps from Standard Signatures: Lower Bounds and Constructions

Xiangyu Liu¹ Riccardo Zanotto^{1,2} Vassilis Zikas³

¹CISPA Helmholtz Center for Information Security, Germany

²Saarland University, Germany ³Georgia Tech, USA

{xiangyu.liu, riccardo.zanotto}@cispa.de vzikas@gatech.edu

Abstract

Atomic swaps are the prototypical application of adaptor signatures. Practical constructions settle with two ordinary on-chain signatures and require no scripting, but they need two sequential online rounds: Alice samples a hard statement and sends its pre-signature, and Bob follows with his own pre-signature using the same statement. We ask whether the protocol can be compressed into a single online round (from Bob to Alice), which we call a *non-interactive* atomic swap, without relying on impractical universal adaptor signatures. In particular, we require that both parties continue to use *standard* signature schemes with unmodified verification algorithms.

Our main result is a lower bound: for statistically unlinkable adaptors with negligible faithfulness failure, Bob’s valid-signature space per message must be superpolynomially larger than the relation’s witness space. This (1) reaffirms the impossibility result by Erwig et al. [PKC 2021] for unique signatures as a trivial corollary, and (2) excludes non-interactive swaps using the same signature scheme on both sides. However, the bound does leave room for *asymmetric* instantiations. We give one in which Alice signs with BLS and Bob uses a pairing-Schnorr adaptor whose witness is Alice’s BLS signature, and prove its security under the native unforgeability of BLS in the random oracle model.

To support more signature schemes in our non-interactive swap framework, we introduce atomic signature swaps *with preprocessing*: Alice publishes message-independent public data before the messages are chosen. These data, together with her public key and the messages, later determine a statement whose witnesses assemble into her native signature. The protocol requires a single online round carrying a pre-signature by Bob and falls into our non-interactive framework. We prove a Schnorr/Schnorr instantiation under the OMDL assumption in the random oracle model, and give a conditional ECDSA/Schnorr instantiation.

1 Introduction

An *atomic swap* [TMM22, HLTW24] lets two parties exchange assets that might live on different ledgers, so that either both transfers take place or neither does. Abstractly, Alice wants a signature σ_B on a transaction m_B under Bob’s key and Bob wants a signature σ_A on m_A under Alice’s key; but neither should obtain the other’s signature without simultaneously enabling the other to obtain theirs. The first realizations used hash time-locked contracts, which require scripting support on both chains and leak the link between the two transactions. *Adaptor signatures* [Poe17, Poe18, AEE⁺21] (ASig) remove both drawbacks.

An adaptor signature scheme ASig extends a signature scheme Σ with a hard NP relation $\mathcal{R}$. The signer can create a pre-signature $\tilde{\sigma}$ bound to a statement Y in $\mathcal{R}$. This pre-signature can be

adapted into a full signature σ using a witness y such that $(Y, y) \in \mathcal{R}$, so that anyone holding $\tilde{\sigma}$ and σ learns y .

The above properties make adaptor signatures a perfect fit for atomic swaps, which are their prototypical application.¹ We recall the standard two-round atomic swap protocol from adaptor signatures [MMS⁺19, AEE⁺21, TMM22, LTZ24] (see Fig. 1(a)): Assume Alice (resp. Bob) wishes to receive a signature from Bob (resp. Alice) on a known message m_B (resp. m_A) where both signature schemes, referred to as Σ_A and Σ_B , admit an adaptor signature extension w.r.t. the same hard relation $\mathcal{R}$. The messages m_A and m_B could for example be unsigned transactions from Alice to Bob and from Bob to Alice. Neither party wants to pay the other without also getting paid. Here is how this is solved: Alice first samples a statement-witness pair $(Y, y) \in \mathcal{R}$, generates a pre-signature $\tilde{\sigma}_A$ on m_A w.r.t. Y , and sends $(Y, \tilde{\sigma}_A)$ to Bob. Bob then uses the same statement Y to produce a pre-signature $\tilde{\sigma}_B$ on m_B and sends it back. Upon receiving $\tilde{\sigma}_B$, Alice adapts it into a full signature σ_B using y , and posts σ_B on chain to claim the transaction m_B . Once the transaction is confirmed, and hence σ_B is public, Bob extracts the witness y from $(\tilde{\sigma}_B, \sigma_B)$, adapts $\tilde{\sigma}_A$ into a full signature σ_A , and posts σ_A on chain to claim m_A . By putting proper time-locks on the transactions (which are allowed even in non-Turing-complete blockchains like Bitcoin) the parties can further ensure that the swap will either occur atomically or will not occur at all.

The above swap requires two *sequential* rounds. This can be acceptable when the swap is arranged between two applications that are already in contact, but can be problematic when the swap occurs between different (unsynchronized) chains, or when the parties have different time zones, or are behind intermittent connectivity. Hence, it is natural to ask how much of this interaction is really necessary.

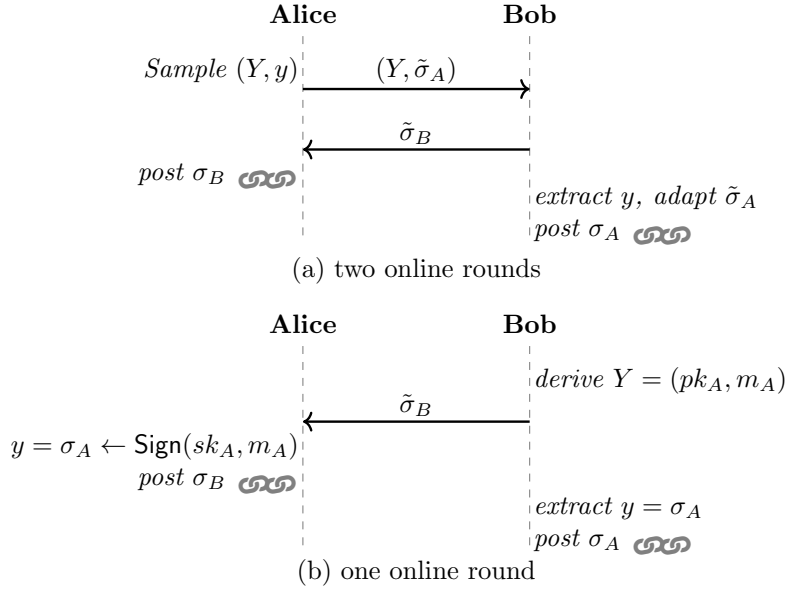


Figure 1: Simplified atomic swap protocols via adaptor signatures. (a) The two-round interactive protocol, where Alice sends $(Y, \tilde{\sigma}_A)$ and Bob sends $\tilde{\sigma}_B$. Both signature schemes have adaptor signature extension. (b) The one-round non-interactive protocol, where Bob derives Y from public data and sends $\tilde{\sigma}_B$. Only Σ_B needs an adaptor signature extension.

¹Adaptor signatures have since become a standard building block for blockchain protocols beyond swaps, including payment channels [AEE⁺21], coin mixing [GMM⁺22], and Discreet Log Contracts [Dry17, Dis24].

The Non-Interactive Framework. The idea to remove a round from such an adaptor-signature-based swap is to make the witness to the relation be itself the desired signature.² More concretely, to realize a non-interactive atomic swap (see Fig. 1(b)) we need an adaptor signature scheme ASig_B w.r.t. the following *direct signature* relation

$$\mathcal{R}_A = \{((pk_A, m_A), \sigma_A) : \text{Ver}_A(pk_A, m_A, \sigma_A) = 1\}, \quad (1)$$

where the statement Y asserts that a valid signature on m_A under pk_A of Σ_A exists, and a witness is any such signature σ_A . Given such ASig_B , Bob derives Y from the public pair (pk_A, m_A) , pre-signs m_B w.r.t. Y under pk_B of Σ_B , and sends $\tilde{\sigma}_B$ to Alice. Alice then produces a signature σ_A on m_A and uses it as a witness to adapt $\tilde{\sigma}_B$ into a full signature σ_B . After observing σ_B on the blockchain, Bob extracts a witness of Y , which is exactly Alice’s signature on m_A .

Beyond non-interactivity, this framework enjoys a second advantage of *asymmetry*. Namely, only Σ_B is required to admit an adaptor signature extension, while Σ_A is kept entirely unmodified. Alice may therefore use a signature scheme without any adaptor extension, for example BLS signatures [BLS01], which are compact and widely deployed but admit no adaptor extension by the impossibility of [EFH⁺21].

But how can we realize ASig_B ? A straightforward way would be to use universal adaptor signatures [DOY22, LTZ24, CLTZ25, GH26, ABC⁺26], which support arbitrary NP relations, including the relation $\mathcal{R}_A$ defined via Σ_A ’s verification. This approach has two drawbacks. First, evaluating the relation in a general-purpose zero-knowledge proof incurs a substantial efficiency loss. The cost is particularly high when Σ_A ’s verification invokes a hash function that is modeled as a random oracle, as for Schnorr and ECDSA. Recently, Abe et al. [ABC⁺26] mitigate this cost using on-line/offline NIZKs and obtain more practical parameters for the universal approach. Second, and more importantly for deployment, the adaptor scheme ASig_B changes the form of the signatures produced by Σ_B : a signature obtained by pre-signing and adapting differs from one produced by standard signing, and verification must be extended to accept it. Giunta and Hostáková [GH26] formalize this as the *scriptlessness* of adaptor signatures, a core desirable property for blockchain applications: the conditional logic of the adaptor scheme should live entirely between the two parties; the verifier never needs to see it, and what it checks is an ordinary signature under the ordinary verification algorithm. Döttling et al. [DTS⁺26] achieve scriptlessness for arbitrary NP relations without modifying the underlying signature scheme, but require an additional advertisement message, making the resulting swap two-round.

In contrast to universal adaptor signatures, adaptor signatures for a specific simple relation, such as the discrete-logarithm relation, typically achieve perfect scriptlessness (the verification algorithm remains unchanged from the underlying one) and perfect unlinkability (a pre-sign-then-adapt signature distributes identically to one produced by standard signing). This raises the question we address in this work:

Can we build a non-interactive atomic swap where both σ_A and σ_B are standard (scriptless and unlinkable) signatures?

1.1 Our Contributions

We study non-interactive atomic swaps from adaptor signatures and prove both negative and positive results.

²The same notion is called a one-way atomic swap in [ABC⁺26].

A Lower Bound on Signature Expansion. Our main technical result quantifies how large Bob’s signature space must be to support an adaptor for a relation $\mathcal{R}$. Let Q_B be the number of valid Σ_B signatures per message and let N be the size of the witness set per statement. For an ϵ_{ow} -one-way relation, Theorem 3.8 implies

$$Q_B \geq \frac{N(1 - \delta)^2}{\epsilon_{ow} + \mu},$$

Here μ measures statistical unlinkability, and δ bounds the probability that extraction returns a witness different from the one used for adaptation. When μ , δ , and ϵ_{ow} are negligible, the bound forces Q_B to exceed N by a superpolynomial factor. We obtain the following consequences.

- Unique signature schemes, such as BLS, have $Q_B = 1$ and therefore admit no adaptor signature extension for any hard relation. This reaffirms (as a corollary of our bound) the impossibility from [EFH⁺21].
- For a non-interactive swap using the direct signature relation $\mathcal{R}_A$ from Eq. (1), $N = Q_A$. Our theorem implies that Q_B/Q_A is superpolynomial, and hence schemes with comparable signature-space sizes, including the same scheme, are excluded.

BLS Becomes Compatible with Atomic Signature Swaps. On the positive side, the above cardinality bound leaves room for a construction with $Q_A = 1$. We instantiate Alice’s scheme with BLS and Bob’s with pairing-Schnorr over $(\mathbb{G}_1, \mathbb{G}_T)$. Bob’s adaptor uses the relation $Y = e(W, g_2)$, whose witness W is Alice’s BLS signature. We give explicit algorithms and a reduction to native BLS unforgeability, with Bob’s challenge hash in the random-oracle model. The resulting Bob signature has a group-element response and a pairing-based verifier. This construction does not contradict the lower bound and the impossibility of [EFH⁺21], since BLS is only used on Alice’s side, which requires no adaptor extension (cf. Fig. 1(b)).

Atomic Signature Swaps with Preprocessing. To extend the range of signature schemes our non-interactive paradigm admits, we introduce offline preprocessing. In the direct approach, Bob’s adaptor must support Alice’s verifier, whose witnesses are complete signatures and whose form may be arbitrarily complicated. Our approach replaces the full verification relation $\mathcal{R}_A$ of Eq. (1) by a simpler relation that Bob’s adaptor can support, as shown in Fig. 2. In a pre-processing phase, Alice publishes data ρ (independently of the message); the triple (pk_A, ρ, m_A) then determines a statement $Y = \text{Stmt}(pk_A, \rho, m_A)$ whose witness w is her online signing response. Public assembly turns every valid witness into a native Alice signature, even when ρ is maliciously generated.

For Schnorr and ECDSA, preprocessing fixes signing randomness (so that $N = 1$) to derive a fixed-base discrete-logarithm statement. In Schnorr, the slot and message determine R and $Y = Rpk_A^{H(R, pk_A, m_A)}$, and the unique scalar witness s gives the signature (R, s) . This changes the relation from q witnesses to one, rather than beating the bound for the same relation. BLS fits the same framework with empty preprocessing and $\mathcal{R}_{\text{pair}} = \{(Y, W) : e(W, g_2) = Y\}$. We formalize *statement-derivable signatures*, define unforgeability with public preprocessing (PP-EUF) and sound assembly, and prove a generic composition theorem without requiring unique witnesses. The resulting exchanges appear in Fig. 2 and Table 1.

- For Schnorr, a bundle of fixed public nonces permits a generalized-birthday forgery attack, as described in Sect. 4.7. We use a message-dependent two-nonce derivation and prove its security under the OMDL assumption in the random-oracle model.

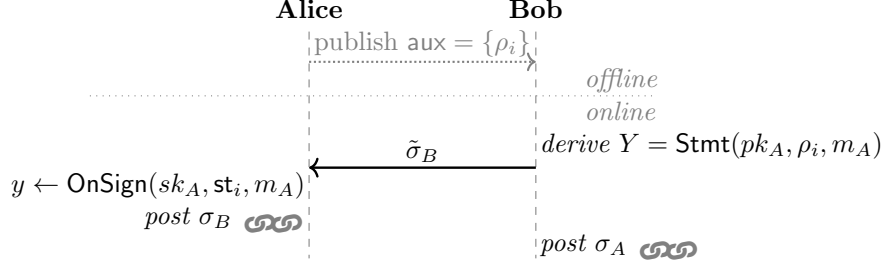


Figure 2: The atomic signature swap with preprocessing from statement-derivable signatures. Alice publishes a bundle of message-independent offline outputs once. Given m_A and m_B , Bob derives Y from Alice’s public key and aux , and sends a pre-signature to Alice. The witness is Alice’s online signing output. The instantiations are Schnorr/Schnorr and, conditionally, ECDSA/Schnorr.

- For ECDSA, the direct verification equation uses a nonce-dependent discrete-logarithm base. Two additional public group elements and consistency proofs give a statement in the fixed-base relation used by scalar Schnorr. We prove sound assembly; unforgeability with this extra public information remains an explicit assumption.

Protocol	Online rounds	ASig ext. required for	σ_A, σ_B standard	Instantiations ($\Sigma_A \times \Sigma_B$)
AEE+21 [AEE ⁺ 21]	2	Σ_A, Σ_B	✓	Schnorr $\times$ Schnorr ECDSA $\times$ ECDSA
ASig with advertisement [DTS ⁺ 26]	2	Σ_B	✓	any $\times$ ASig _{AD}
Universal ASig [DOY22]	1	Σ_B	$\times$	any $\times$ UASig
This work	1	Σ_B	✓	BLS $\times$ pairing-Schnorr
This work (preprocessing)	1	Σ_B	✓	Schnorr $\times$ Schnorr ECDSA $\times$ Schnorr (conditional)

Table 1: Atomic swaps from adaptor signatures. Online rounds count messages between Alice and Bob. “ σ_A, σ_B standard” indicates whether both σ_A and σ_B are accepted by the unmodified verification algorithms of Σ_A and Σ_B , respectively.

1.2 Related Work

Atomic Swaps and Adaptor Signatures. Atomic swaps were classically realized with hash time-locked contracts, which require scripting support on both chains and publicly link the two transactions. The concept of adaptor signatures was first introduced as scriptless scripts by Poelstra in [Poe17, Poe18] and later formalized by Aumayr et al. [AEE⁺21]. Recently numerous concrete constructions have been proposed under a variety of assumptions, including Schnorr-based schemes [AEE⁺21], ECDSA-based schemes [AEE⁺21, TZY22, HMR26], LWE/SIS-based (LAS) [EEE20], group action-(isogeny-)based IAS, ORCAS, and GATOR [TMM21, CDG⁺26, DMZ26], and code-based schemes [KH22]. Generic constructions have been shown from identification schemes with suitable homomorphic properties by Erwig et al. [EFH⁺21], and from dichotomic signatures by Gerhart et al. [GSST24].

Universal Adaptor Signatures. A separate line of work constructs universal adaptor signatures for arbitrary NP relations. Dai et al. [DOY22] gave a general construction from arbitrary signatures and NP languages, later refined by Liu et al. [LTZ24] and Ciampi et al. [CLTZ25] to additionally achieve witness hiding. Abe et al. [ABC⁺26] take a different route through online/offline NIZKs: the offline prover randomness enables extraction from the online proof, leading to a substantially more efficient post-quantum construction. Despite this efficiency improvement, the resulting complete signature includes the NIZK proof and requires an augmented verifier; hence it is not scriptless with respect to the underlying signature scheme. Giunta and Hostáková [GH26] formalize this deployment requirement, and constructing universal adaptor signatures that are fully scriptless remains a challenging open problem. An alternative way to achieve scriptlessness for arbitrary NP relations is to add protocol communication. Döttling et al. [DTS⁺26] give two general compilers supporting diverse signature schemes while achieving *statement truth privacy*; both require an additional advertisement message, making the resulting swap two-round.

Comparison with [GMTMSS26]. The impossibility of [EFH⁺21] has motivated several attempts to recover adaptor-like functionality for unique signatures like BLS. Concurrently and independently of this work, Gomez-Martinez et al. [GMTMSS26] introduce *two-party asymmetric-input solitary-output adaptor signatures* (2P-AISO-AS), in which PreSign is an interactive protocol between two parties. They instantiate the notion efficiently from BLS in two ways and obtain BLS-compatible contingent payments, atomic swaps, and coin mixing. Both [GMTMSS26] and Section 4.9 give BLS-compatible swaps. The pre-signing in [GMTMSS26] is interactive, while our construction uses BLS for Alice and non-interactive pairing-Schnorr pre-signing for Bob. The lower-bound adversary locally pre-signs its supplied challenge, so Theorem 3.8 does not apply to their interactive primitive.

2 Preliminaries

Let $\lambda \in \mathbb{N}$ denote the security parameter throughout the paper. For $n \in \mathbb{Z}_{>0}$, define $[n] := \{1, 2, \dots, n\}$. We write $x := y$ for the assignment of y to x . Let $x \xleftarrow{\$} S$ denote the operation of sampling x uniformly at random from a set S . For a distribution $\mathcal{D}$, we write $x \leftarrow \mathcal{D}$ to denote sampling x according to $\mathcal{D}$, and write $\mathcal{D}(x)$ to denote the probability that the sample equals x . For an algorithm $\mathcal{A}$, we write $y \leftarrow \mathcal{A}(x)$ to denote running $\mathcal{A}$ on input x and assigning the output to y . We write $y := \mathcal{A}(x; r)$ if the randomness r is specified.

We use $\llbracket P \rrbracket$ to denote the bit 1 if the Boolean predicate P evaluates to true, and 0 otherwise. A function $f(\lambda)$ is said to be negligible if, for every polynomial $p(\lambda)$, there exists $\lambda_0 \in \mathbb{Z}$ such that $f(\lambda) < \frac{1}{p(\lambda)}$ for all $\lambda > \lambda_0$.

For random variables $\mathcal{X}$ and $\mathcal{Y}$ over S , define their min entropy and statistical distance, respectively, as

$$\mathbf{H}_{\infty}(\mathcal{X}) := -\log \left(\max_{s \in S} \Pr[\mathcal{X} = s] \right),$$

$$\Delta(\mathcal{X}, \mathcal{Y}) := \frac{1}{2} \sum_{s \in S} |\Pr[\mathcal{X} = s] - \Pr[\mathcal{Y} = s]|.$$

2.1 Signatures

Definition 2.1 (Signatures). A signature scheme $\Sigma = (\text{Gen}, \text{Sign}, \text{Ver})$ consists of the following three algorithms.

- $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$. The key generation algorithm takes as input the security parameter λ , and outputs a public key pk and a secret key sk . We use $[\text{Gen}(1^\lambda)]$ to denote the set of all key pairs generated via $\text{Gen}(1^\lambda)$.
- $\sigma \leftarrow \text{Sign}(sk, m)$. The signing algorithm takes as input sk and a message m , and outputs a signature σ .
- $0/1 \leftarrow \text{Ver}(pk, m, \sigma)$. The (deterministic) verification algorithm takes as input pk , m , and σ , and outputs a bit.

Correctness. For any $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$, any message m and $\sigma \leftarrow \text{Sign}(sk, m)$, it holds that $\text{Ver}(pk, m, \sigma) = 1$.

Definition 2.2 ((Strong) unforgeability of signatures). A signature scheme **Sig** is (strongly) unforgeable under chosen message attacks, if for any PPT adversary $\mathcal{A}$, it holds that

$$\text{Adv}_{\Sigma, \mathcal{A}}^{(\text{s})\text{uf}}(\lambda) := \Pr[(\text{S})\text{UF}_{\Sigma, \mathcal{A}} = 1] \leq \text{negl}(\lambda),$$

where the experiments $(\text{S})\text{UF}_{\Sigma, \mathcal{A}}$ are defined in Fig. 3.

Experiment $(\text{S})\text{UF}_{\Sigma, \mathcal{A}}$:	Oracle $\text{SIGN}(m)$:
1. $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$; $\mathcal{S} := \emptyset$	5. $\sigma \leftarrow \text{Sign}(sk, m)$
2. $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\text{SIGN}}(pk)$	6. $\mathcal{S} := \mathcal{S} \cup \{(m, \sigma)\}$
3. return $\llbracket \text{Ver}(pk, m^*, \sigma^*) \wedge (m^*, \cdot) \notin \mathcal{S} \rrbracket$ // <i>UF</i>	7. return σ
4. return $\llbracket \text{Ver}(pk, m^*, \sigma^*) \wedge (m^*, \sigma^*) \notin \mathcal{S} \rrbracket$ // <i>SUF</i>	

Figure 3: The (strong) unforgeability experiment for signature scheme Σ .

We recall some well-known signature schemes like Schnorr, ECDSA, and BLS in Sect. 2.2.

2.2 Schnorr, ECDSA, Okamoto, and BLS Signatures

In this section, we present Schnorr [Sch90], ECDSA [Nat23, JMV01], Okamoto [Oka93], and BLS [BLS01] signatures. The first three are operated over a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q together with a hash function modelled as a random oracle, whereas BLS additionally requires a bilinear pairing.

Schnorr Signatures. Let $H : \{0, 1\}^* \rightarrow \mathbb{Z}_q$. The key generation algorithm samples $x \xleftarrow{\$} \mathbb{Z}_q$ and outputs $sk = x$, $pk = X := g^x$. To sign m , the signer samples a nonce $r \xleftarrow{\$} \mathbb{Z}_q$, sets $R := g^r$ and $c := H(R, X, m)$, and outputs

$$\sigma = (R, s), \quad s := r + c \cdot x \bmod q.$$

Verification accepts (R, s) if $g^s = R \cdot X^{H(R, X, m)}$.

For every fixed $R \in \mathbb{G}$ the verification equation determines s uniquely, and conversely every R admits such an s . Hence $|\mathcal{S}_{pk, m}| = q$ and Schnorr is q -regular. Since **Sign** samples R uniformly from $\mathbb{G}$, its output is uniform on $\mathcal{S}_{pk, m}$. Schnorr signatures are strongly unforgeable in the random oracle model under the discrete logarithm assumption. The Schnorr adaptor signature was introduced by Aumayr et al. [AEE⁺21], which supports the discrete logarithm relation $(Y = g^y, y) \in \mathcal{R}$.

ECDSA Signatures. Let $H : \{0, 1\}^* \rightarrow \mathbb{Z}_q$ and let $f : \mathbb{G} \rightarrow \mathbb{Z}_q$ be the usual conversion function mapping a group element to its x -coordinate modulo q . Key generation is as for Schnorr: $sk = x$, $pk = X = g^x$. To sign m , the signer samples $k \xleftarrow{\$} \mathbb{Z}_q^*$, sets $R := g^k$ and $r := f(R)$, and outputs

$$\sigma = (r, s), \quad s := k^{-1}(H(m) + r \cdot x) \bmod q,$$

restarting if $r = 0$ or $s = 0$. Verification accepts (r, s) if $f(g^{H(m)/s} \cdot X^{r/s}) = r$.

A valid (r, s) determines the reconstructed nonce point. In a raw, idealized counting model, ECDSA therefore has $\Theta(q)$ (approximately q) valid signatures and signing is approximately uniform over its support. The exact count depends on retries for $r = 0$ or $s = 0$, coordinate reduction, the $(r, s)/(r, -s)$ symmetry, and low- s normalization in some deployments. The ECDSA adaptor signature was introduced by Aumayr et al. [AEE⁺21], which also supports the discrete logarithm relation $(Y = g^y, y) \in \mathcal{R}$.

Unlike Schnorr, ECDSA is *not* strongly unforgeable: the nonces k and $-k$ yield the same r , since g^k and g^{-k} share an x -coordinate, so (r, s) and $(r, -s)$ are both valid and either can be computed from the other. Nevertheless, known ECDSA adaptors can have perfectly faithful extraction; this is why Sect. 3.4 treats faithfulness separately from strong one-wayness.

Okamoto Signatures. Okamoto's scheme [Oka93] uses two independent generators $g_1, g_2 \in \mathbb{G}$ whose relative discrete logarithm is unknown. Key generation samples $x_1, x_2 \xleftarrow{\$} \mathbb{Z}_q$ and outputs $sk = (x_1, x_2)$, $pk = X := g_1^{x_1} g_2^{x_2}$. To sign m , the signer samples $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_q$, sets $R := g_1^{r_1} g_2^{r_2}$ and $c := H(R, X, m)$, and outputs

$$\sigma = (R, s_1, s_2), \quad s_1 := r_1 + c \cdot x_1, \quad s_2 := r_2 + c \cdot x_2.$$

Verification accepts if $g_1^{s_1} g_2^{s_2} = R \cdot X^c$.

The verification equation is a single linear constraint on the pair $(s_1, s_2) \in \mathbb{Z}_q^2$, so each of the q possible values of R admits exactly q valid pairs. Hence $|\mathcal{S}_{pk, m}| = q^2$ and Okamoto is q^2 -regular, with **Sign** uniform on $\mathcal{S}_{pk, m}$. The Okamoto signature scheme is strongly unforgeable in the random oracle model under the discrete logarithm assumption.

BLS Signatures. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear pairing over groups of prime order q with generators g_1, g_2 , and let $H : \{0, 1\}^* \rightarrow \mathbb{G}_1$ be a hash function modelled as a random oracle. Key generation samples $x \xleftarrow{\$} \mathbb{Z}_q$ and outputs $sk = x$, $pk = X := g_2^x$. Signing is deterministic,

$$\sigma := H(m)^x \in \mathbb{G}_1,$$

and verification accepts if $e(\sigma, g_2) = e(H(m), X)$.

BLS is a *unique* signature scheme: the verification equation admits exactly one solution σ for each (pk, m) , so $|\mathcal{S}_{pk, m}| = 1$ and BLS is 1-regular. Uniqueness makes BLS strongly unforgeable as soon as it is existentially unforgeable, which holds in the random oracle model under the co-CDH assumption. On the other hand, uniqueness is precisely what prevents BLS from carrying an adaptor signature extension [EFH⁺21]. Our Theorem 3.8 recovers this impossibility as a special case.

2.3 Hardness Assumptions

Definition 2.3 (One-more discrete logarithm [NRS21]). We say the (ℓ) -one-more discrete logarithm $((\ell)$ OMDL) assumption holds over $\mathbb{G}$, if for any PPT adversary $\mathcal{A}$, it holds that

$$\text{Adv}_{\ell, \mathbb{G}, \mathcal{A}}^{\text{omdl}}(\lambda) := \Pr[\text{OMDL}_{\ell, \mathbb{G}, \mathcal{A}}(\lambda) = 1] \leq \text{negl}(\lambda)$$

where the experiment $\text{OMDL}_{\ell, \mathbb{G}, \mathcal{A}}(\lambda)$ is defined in Fig. 4.

Experiment $\text{OMDL}_{\ell, \mathbb{G}, \mathcal{A}}(\lambda)$:	Oracle $\text{CHAL}()$: // ℓ queries in total
1. $\ell := 0$; $\text{ctr} := 0$	5. $\ell := \ell + 1$
2. $(\{x'_i\}_{i \in [\ell]}) \leftarrow \mathcal{A}^{\text{CHAL}, \text{DLOG}}(\mathbb{G}, g, q)$	6. $x_\ell \xleftarrow{\$} \mathbb{Z}_q$; $X_\ell := g^{x_\ell}$
3. require $\text{ctr} < \ell$	7. return X_ℓ
4. return $\llbracket \forall i \in [\ell] : X_i = g^{x'_i} \rrbracket$	
	Oracle $\text{DLOG}(X \in \mathbb{G})$:
	8. $\text{ctr} := \text{ctr} + 1$
	9. return $\log_g X$

Figure 4: The ℓ -(algebraic) one-more discrete logarithm game.

2.4 NP Languages

Let $\{\mathcal{R}_\lambda\} \subseteq (\{0, 1\}^* \times \{0, 1\}^*)_\lambda$ be a series of binary relations indexed by parameter λ . If λ is fixed then we simply denote $\mathcal{R}$. We call $\mathcal{R}$ an NP relation if there is an efficient algorithm to check whether $(Y, y) \in \mathcal{R}$. The relation $\mathcal{R}$ defines an NP language $\mathcal{L}_\mathcal{R} := \{Y \in \{0, 1\}^* \mid \exists y \in \{0, 1\}^* \text{ s.t. } (Y, y) \in \mathcal{R}\}$. We call Y the statement (or instance, not necessarily in $\mathcal{L}_\mathcal{R}$), and y a witness of Y if $(Y, y) \in \mathcal{R}$. We assume there are efficient sample algorithms that returns an instance-witness pair (Y, y) and y is uniformly sampled from the witness set $\text{Wit}(Y)$. Formally, $(Y, y) \leftarrow \text{Sample}(\mathcal{R})$.

Definition 2.4 (One-wayness). A relation $\mathcal{R}$ is ϵ_{ow} -one-way if for any PPT adversary $\mathcal{A}$,

$$\text{Adv}_{\mathcal{R}, \mathcal{A}}^{\text{ow}}(\lambda) := \Pr[(Y, y) \leftarrow \text{Sample}(\mathcal{R}); y' \leftarrow \mathcal{A}(Y) : (Y, y') \in \mathcal{R}] \leq \epsilon_{ow}.$$

Definition 2.5 (Strong one-wayness). A relation $\mathcal{R}$ is ϵ_{sow} -strongly one-way if for every PPT $\mathcal{A}$,

$$\Pr[(Y, y) \leftarrow \text{Sample}(\mathcal{R}); y' \leftarrow \mathcal{A}(Y, y) : y' \in \text{Wit}(Y) \wedge y' \neq y] \leq \epsilon_{sow}.$$

For $\mathcal{R}_A := \{((pk_A, m_A), \sigma) \mid \text{Ver}_A(pk_A, m_A, \sigma) = 1\}$, one-wayness corresponds to key-only unforgeability of Σ_A , and strong one-wayness to its key-only strong unforgeability variant. EUF-CMA and SUF-CMA imply the respective key-only notions. The native signing algorithm is a uniform witness sampler whenever it is uniform on $\mathcal{S}_{pk_A, m_A}$, as for Schnorr and BLS and for the raw, idealized ECDSA model up to the conventions above.

2.5 Adaptor Signatures

Definition 2.6 (Adaptor signatures). An adaptor signature (ASig) scheme w.r.t. a relation $\mathcal{R}$ consists of seven algorithms $\text{ASig} = (\text{Gen}, \text{Sign}, \text{Ver}, \text{pSign}, \text{pVer}, \text{Adapt}, \text{Extract})$, where the first three algorithms are defined in the same manner as for a standard signature scheme $\Sigma = (\text{Gen}, \text{Sign}, \text{Ver})$ (cf. Def. 2.1), and the last four are defined as follows.

- $\tilde{\sigma} \leftarrow \text{pSign}(sk, m, Y)$. The pre-sign algorithm takes as input sk , m and an instance Y , and outputs a pre-signature $\tilde{\sigma}$.
- $b \leftarrow \text{pVer}(pk, m, Y, \tilde{\sigma})$. The pre-verification algorithm takes as input pk , m , Y and $\tilde{\sigma}$, and outputs a bit indicating the validity of $\tilde{\sigma}$.
- $\sigma \leftarrow \text{Adapt}(pk, m, \tilde{\sigma}, y)$. The adaptation algorithm takes as input pk , m , $\tilde{\sigma}$ and a witness y as input, and outputs an adapted signature σ .
- $y/\perp \leftarrow \text{Extract}(pk, m, Y, \tilde{\sigma}, \sigma)$. The (deterministic) extraction algorithm takes as input pk , m , Y , $\tilde{\sigma}$ and σ , and outputs a witness y , or a failure symbol $\perp$.

Correctness. We require the pre-signature correctness, the adaptation correctness, and the extraction correctness.

For any $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$, any message m , any $(Y, y) \in \mathcal{R}$, any $\tilde{\sigma} \leftarrow \text{pSign}(sk, m, Y)$, $\sigma \leftarrow \text{Adapt}(pk, m, \tilde{\sigma}, y)$, and $y' \leftarrow \text{Extract}(pk, m, Y, \tilde{\sigma}, \sigma)$, it holds that

1. (Pre-signature correctness) $\text{pVer}(pk, m, Y, \tilde{\sigma}) = 1$;
2. (Adaptation correctness) $\text{Ver}(pk, m, \sigma) = 1$;
3. (Extraction correctness) $(Y, y') \in \mathcal{R}$.

We consider the following security properties for adaptor signatures, following [DOY22, GSST24, LTZ24].

Full extractability. The signer can extract a witness from the pre-signature and the adapted signature. Note that if the relation $\mathcal{R}$ is one-way, then full extractability implies unforgeability (the adversary cannot generate a valid pre-signature or a regular signature on a fresh message) [DOY22].

Pre-signature adaptability. The receiver can adapt a valid pre-signature into a valid (full) signature with the knowledge of a witness.

Unlinkability. The presign-then-adapt signature is indistinguishable from a standard signature.

We formalize full extractability, pre-signature adaptability, and unlinkability below. We also give the optional witness-hiding and pre-verify-soundness notions immediately afterward.

Definition 2.7 (Full extractability of ASig). An adaptor signature scheme ASig w.r.t. relation R is fully extractable, if for any PPT adversary $\mathcal{A}$,

$$\text{Adv}_{\text{ASig}, \mathcal{A}}^{\text{fext}}(\lambda) := \Pr[\text{FEXT}_{\text{ASig}, \mathcal{A}}(\lambda) = 1] \leq \text{negl}(\lambda),$$

where the experiment $\text{FEXT}_{\text{ASig}, \mathcal{A}}(\lambda)$ is defined in Fig. 5.

<u>Experiment FEXT_{ASig,A}(λ):</u>	<u>Oracle SIGN(m):</u>
1. $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$	7. $\sigma \leftarrow \text{Sign}(sk, m)$
2. $\mathcal{S} := \emptyset; \mathcal{T}[m] := \emptyset$	8. $\mathcal{S} := \mathcal{S} \cup \{m\}$
3. $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\text{SIGN}, \text{PSIGN}}(pk)$	9. return σ
4. $b_1 := \llbracket \text{Ver}(pk, m^*, \sigma^*) \wedge m^* \notin \mathcal{S} \rrbracket$	
5. $b_2 := \llbracket \forall (Y, \tilde{\sigma}) \in \mathcal{T}[m^*] : (Y, \text{Extract}(pk, m^*, Y, \tilde{\sigma}, \sigma^*)) \notin R \rrbracket$	<u>Oracle PSIGN(m, Y):</u>
6. return $\llbracket b_1 \wedge b_2 \rrbracket$	10. $\tilde{\sigma} \leftarrow \text{pSign}(sk, m, Y)$
	11. $\mathcal{T}[m] := \mathcal{T}[m] \cup \{(Y, \tilde{\sigma})\}$
	12. return $\tilde{\sigma}$

Figure 5: Full-extractability experiment for adaptor signature scheme ASig.

Definition 2.8 (Pre-signature adaptability of ASig). An adaptor signature scheme ASig w.r.t. relation R has pre-signature adaptability, if for any public key pk , any message m , any $(Y, y) \in R$ and pre-signature $\tilde{\sigma}$ s.t. $\text{pVer}(pk, m, Y, \tilde{\sigma}) = 1$, it holds that $\text{Ver}(pk, m, \text{Adapt}(pk, m, \tilde{\sigma}, y)) = 1$.

Definition 2.9 (Computational unlinkability of ASig). An adaptor signature scheme ASig w.r.t. relation $\mathcal{R}$ is computationally unlinkable, if for any PPT adversary $\mathcal{A}$,

$$\text{Adv}_{\text{ASig}, \mathcal{A}}^{\text{unlink}}(\lambda) := |2 \cdot \Pr[\text{UNLINK}_{\text{ASig}, \mathcal{A}}(\lambda) = 1] - 1| \leq \text{negl}(\lambda),$$

where the experiment $\text{UNLINK}_{\text{ASig}, \mathcal{A}}(\lambda)$ is defined in Fig. 6.

<u>Experiment UNLINK_{ASig,A}(λ):</u>	<u>Oracle CHALL(m, Y, y):</u>
1. $b \xleftarrow{\$} \{0, 1\}$	5. require $(Y, y) \in R$
2. $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$	6. $\tilde{\sigma} \leftarrow \text{pSign}(sk, m, Y)$
3. $b' \leftarrow \mathcal{A}^{\text{CHALL}, \text{SIGN}, \text{PSIGN}}(pk)$	7. $\sigma_0 \leftarrow \text{Adapt}(pk, m, \tilde{\sigma}, y)$
4. return $\llbracket b = b' \rrbracket$	8. $\sigma_1 \leftarrow \text{Sign}(sk, m)$
	9. return σ_b
	The SIGN and PSIGN oracles are as in Fig. 5.

Figure 6: Unlinkability experiment for adaptor signature scheme ASig.

Definition 2.10 (Witness hiding of ASig). An adaptor signature scheme ASig w.r.t. relation R is witness hiding, if there exists a simulator Sim such that, for any PPT adversary $\mathcal{A}$, it holds that

$$\text{Adv}_{\text{ASig}, \text{Sim}, \mathcal{A}}^{\text{wh}}(\lambda) := |2 \cdot \Pr[\text{WitH}_{\text{ASig}, \text{Sim}, \mathcal{A}}(\lambda) = 1] - 1| \leq \text{negl}(\lambda),$$

where the experiment $\text{WitH}_{\text{ASig}, \text{Sim}, \mathcal{A}}(\lambda)$ is defined in Fig. 7.

<u>WitH_{ASig,Sim,A}(λ):</u>	<u>CHALL(pk, sk, m):</u>
1. $b \xleftarrow{\$} \{0, 1\}$	5. require $f_{\text{ASig}}(pk, sk) = 1$
2. $(Y, y) \leftarrow \text{Sample}(R)$	6. <i>// check the validity of (pk, sk)</i>
3. $b' \leftarrow \mathcal{A}^{\text{CHALL}}(Y)$	7. $\tilde{\sigma} \leftarrow \text{pSign}(sk, m, Y)$
4. return $\llbracket b = b' \rrbracket$	8. $\sigma_0 \leftarrow \text{Adapt}(pk, m, \tilde{\sigma}, y)$
	9. $\sigma_1 \leftarrow \text{Sim}(sk, m, Y)$
	10. return σ_b

Figure 7: Witness-hiding experiment for adaptor signature scheme ASig.

Note that witness hiding is implied by unlinkability [LTZ24].

Definition 2.11 (Pre-verify soundness of ASig). An adaptor signature scheme ASig w.r.t. binary relation R has pre-verify soundness, if for any PPT adversary and any $Y \notin \mathcal{L}_R$, it holds that

$$\text{Adv}_{\text{ASig}, \mathcal{A}}^{\text{snd}}(\lambda) := \Pr \left[\begin{array}{l} (pk, sk) \leftarrow \text{Gen}(1^\lambda) \\ (m, \tilde{\sigma}) \leftarrow \mathcal{A}(sk) \end{array} : \text{pVer}(pk, m, Y, \tilde{\sigma}) = 1 \right] \leq \text{negl}(\lambda).$$

3 Lower Bound for Non-Interactive Atomic Swaps

In this section we study the lower bound for non-interactive atomic swap from ASig. Throughout this section, we abbreviate $\text{Adapt}(pk, m, \tilde{\sigma}, y)$ as $\text{Adapt}(\tilde{\sigma}, y; r)$ and $\text{Extract}(pk, m, Y, \tilde{\sigma}, \sigma)$ as $\text{Extract}(\tilde{\sigma}, \sigma, Y)$ when the omitted inputs are clear. We assume perfect correctness and deterministic extraction for

$$\text{ASig}_B = (\text{Gen}, \text{Sign}, \text{Ver}, \text{pSign}, \text{pVer}, \text{Adapt}, \text{Extract}).$$

3.1 Definitions and Useful Lemmas

Definition 3.1 (Regularity). Let $\Sigma = (\text{Gen}, \text{Sign}, \text{Ver})$ be a signature scheme. For $(pk, sk) \in [\text{Gen}(1^\lambda)]$ and message m , define $\mathcal{S}_{pk, m} := \{\sigma \mid \text{Ver}(pk, m, \sigma) = 1\}$. We say Σ is Q -regular if $|\mathcal{S}_{pk, m}| = Q$ for all such pk and m .

Analogously, a binary relation $\mathcal{R}$ is W -witness-regular if $|\text{Wit}(Y)| = W$ for all $Y \in \mathcal{L}_{\mathcal{R}}$.

Schnorr is q -regular over a prime-order group, and unique signature schemes such as BLS are 1-regular. For ECDSA we use the raw, idealized counting model with $\Theta(q)$ (approximately q) valid signatures; retry conditions, coordinate reduction, sign symmetry, and low- s normalization affect the exact deployed count.

Definition 3.2 (Extractable set). For all $(pk, sk) \in [\text{Gen}(1^\lambda)]$, messages m , statements $Y \in \mathcal{L}_{\mathcal{R}}$, and $\tilde{\sigma} \in [\text{pSign}(sk, m, Y)]$, define the *extractable set*

$$\mathcal{E}(\tilde{\sigma}, Y) := \{\sigma \in \mathcal{S}_{pk, m} \mid \text{Extract}(\tilde{\sigma}, \sigma, Y) \in \text{Wit}(Y)\}.$$

Definition 3.3 (Pre-sign-then-adapt distribution). Fix an honest key pair (pk, sk) , a message m , and $(Y, y) \in \mathcal{R}$. Let $\mathcal{D}_y(\sigma)$ be the probability over $\mathcal{S}_{pk, m}$ obtained by sampling $\tilde{\sigma} \leftarrow \text{pSign}(sk, m, Y)$ and outputting $\sigma = \text{Adapt}(\tilde{\sigma}, y)$, i.e.,

$$\mathcal{D}_y(\sigma) := \Pr[\tilde{\sigma} \leftarrow \text{pSign}(sk, m, Y); \sigma' \leftarrow \text{Adapt}(\tilde{\sigma}, y) : \sigma' = \sigma].$$

Let $\mathcal{D}$ be the distribution obtained by the same experiment with $y \xleftarrow{\$} \text{Wit}(Y)$ drawn uniformly. By total probability,

$$\mathcal{D}(\sigma) = \frac{1}{|\text{Wit}(Y)|} \cdot \sum_{y \in \text{Wit}(Y)} \mathcal{D}_y(\sigma), \text{ equivalently } \sum_{y \in \text{Wit}(Y)} \mathcal{D}_y(\sigma) = |\text{Wit}(Y)| \cdot \mathcal{D}(\sigma). \quad (2)$$

We emphasize that $\mathcal{D}$ need not be efficiently samplable, and it occurs only in the analysis. Both $\mathcal{D}_y$ and $\mathcal{D}$ are supported on $\mathcal{S}_{pk, m}$.

Definition 3.4 (Statistical unlinkability). An adaptor signature scheme ASig based on Σ has μ -unlinkability if for all $(pk, sk) \in [\text{Gen}(1^\lambda)]$, message m and $(Y, y) \in \mathcal{R}$,

$$\Delta(\text{Sign}(sk, m), \mathcal{D}_y) \leq \mu.$$

Our lower bound needs to know how often adapting a pre-signature with a witness and then extracting fails to return *that same* witness. The classical extraction correctness in Definition 2.6 only guarantees that Ext returns *some* valid witness. We isolate the stronger property as faithful extraction.

Definition 3.5 (Faithful extraction). An adaptor signature scheme ASig w.r.t. a relation $\mathcal{R}$ is δ -faithful if for all $(pk, sk) \in [\text{Gen}(1^\lambda)]$ and message m ,

$$\Pr \left[\begin{array}{l} (Y, y) \leftarrow \text{Sample}(\mathcal{R}); \tilde{\sigma} \leftarrow \text{pSign}(sk, m, Y); \\ \sigma \leftarrow \text{Adapt}(\tilde{\sigma}, y); y' := \text{Ext}(\tilde{\sigma}, \sigma, Y) \end{array} : y' \neq y \right] \leq \delta.$$

We say ASig has *perfectly faithful extraction* if $\delta = 0$.

Faithfulness is a property of the scheme rather than of the relation, and it is immediate for every construction we are aware of, including Schnorr [AEE+21], ECDSA [AEE+21], and lattice-based schemes [EEE20]. It is also achieved by all existing universal adaptor signatures [DOY22, LTZ24, CLTZ25].

Lemma 3.6 (Disjoint fibres). *For every $\tilde{\sigma}$, the sets*

$$\mathcal{E}_y(\tilde{\sigma}) := \{\sigma \in \mathcal{S}_{pk, m} \mid \text{Extract}(\tilde{\sigma}, \sigma, Y) = y\}, \quad y \in \text{Wit}(Y),$$

are pairwise disjoint, and $\mathcal{E}(\tilde{\sigma}, Y) = \bigcup_{y \in \text{Wit}(Y)} \mathcal{E}_y(\tilde{\sigma})$.

Proof. Since $\text{Extract}(\tilde{\sigma}, \cdot, Y)$ is deterministic, every σ lies in at most one $\mathcal{E}_y(\tilde{\sigma})$, and $\sigma \in \mathcal{E}(\tilde{\sigma}, Y)$ holds exactly when its image is a witness. $\square$

The next lemma bounds faithfulness failure averaged over honest keys, which suffices for the lower bound. The converse fails, as Remark 3.10 illustrates.

Lemma 3.7 ($\delta \leq \epsilon_{\text{sow}}$). *If $\mathcal{R}$ is ϵ_{sow} -strongly one-way, then every adaptor signature scheme for $\mathcal{R}$ fails to recover the adaptation witness with probability at most ϵ_{sow} , averaged over honest key generation, for every efficiently chosen message fixed before sampling (Y, y) .*

Proof. Consider the adversary $\mathcal{A}$ that generates $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$, fixes any message m , samples $\tilde{\sigma} \leftarrow \text{pSign}(sk, m, Y)$ and $\sigma \leftarrow \text{Adapt}(\tilde{\sigma}, y)$, and outputs $y' := \text{Extract}(\tilde{\sigma}, \sigma, Y)$. By extraction correctness its output satisfies $y' \in \text{Wit}(Y)$. Hence $\mathcal{A}$ wins the game of Definition 2.5 exactly when $y' \neq y$, which is the failure event of Definition 3.5, now averaged over $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$. $\square$

3.2 Lower Bound

Theorem 3.8. *Let $\mathcal{R}$ be N -witness-regular, let Σ_B be Q_B -regular, and let ASig_B be an adaptor signature scheme for Σ_B w.r.t. $\mathcal{R}$ that is μ -statistically unlinkable and δ -faithful ($0 \leq \delta \leq 1$). Then there exists a PPT adversary $\mathcal{A}$ against the one-wayness of $\mathcal{R}$ such that*

$$\text{Adv}_{\mathcal{R}}^{\text{ow}}(\mathcal{A}) \geq \frac{N}{Q_B}(1 - \delta)^2 - \mu.$$

Proof. The adversary $\mathcal{A}$ against the one-wayness of $\mathcal{R}$ works as follows. On input Y , it generates $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$ and fixes an efficiently chosen message m . It computes $\tilde{\sigma} \leftarrow \text{pSign}(sk, m, Y)$ and $\sigma \leftarrow \text{Sign}(sk, m)$, and outputs $\text{Extract}(\tilde{\sigma}, \sigma, Y)$. Since $\sigma \in \mathcal{S}_{pk, m}$, it wins precisely when $\sigma \in \mathcal{E}(\tilde{\sigma}, Y)$. For fixed (pk, sk) and $Y \in \mathcal{L}_{\mathcal{R}}$, write $W := \text{Wit}(Y)$ and $|W| = N$.

By unlinkability $\mathcal{D}$ is μ -close to $\text{Sign}(sk, m)$. For each fixed outer pre-signature $\tilde{\sigma}$, the set $\mathcal{E}(\tilde{\sigma}, Y)$ is fixed, so statistical distance applies to this event; the sample defining $\mathcal{D}$ uses an independent fresh pre-signature. Therefore,

$$\Pr[\sigma \leftarrow \text{Sign}(sk, m) : \sigma \in \mathcal{E}(\tilde{\sigma}, Y)] \geq \Pr[\sigma \leftarrow \mathcal{D} : \sigma \in \mathcal{E}(\tilde{\sigma}, Y)] - \mu.$$

By Lemma 3.6, for every $\tilde{\sigma}$,

$$\Pr[\sigma \leftarrow \mathcal{D} : \sigma \in \mathcal{E}(\tilde{\sigma}, Y)] = \sum_{y \in W} \mathcal{D}(\mathcal{E}_y(\tilde{\sigma})).$$

Let $p_{\tilde{\sigma}, y}(\sigma) := \Pr_r[\text{Adapt}(\tilde{\sigma}, y; r) = \sigma] \in [0, 1]$, we get

$$\begin{aligned} \mathcal{D}(\mathcal{E}_y(\tilde{\sigma})) &= \sum_{\sigma \in \mathcal{E}_y(\tilde{\sigma})} \mathcal{D}(\sigma) \\ &\geq \sum_{\sigma \in \mathcal{E}_y(\tilde{\sigma})} \mathcal{D}(\sigma) \cdot p_{\tilde{\sigma}, y}(\sigma) \quad // p_{\tilde{\sigma}, y}(\sigma) \leq 1 \\ &= \sum_{\sigma} \mathcal{D}(\sigma) \cdot p_{\tilde{\sigma}, y}(\sigma) \cdot \llbracket \text{Extract}(\tilde{\sigma}, \sigma, Y) = y \rrbracket. \end{aligned}$$

Since Extract is deterministic, the bracket is a deterministic function of $\tilde{\sigma}$. Taking the expectation over $\tilde{\sigma} \leftarrow \text{pSign}(sk, m, Y)$ turns the summand into a joint probability:

$$\begin{aligned} \mathbb{E}_{\tilde{\sigma}}[p_{\tilde{\sigma}, y}(\sigma) \cdot \llbracket \text{Extract}(\tilde{\sigma}, \sigma, Y) = y \rrbracket] \\ = \Pr_{\tilde{\sigma}, r}[\text{Adapt}(\tilde{\sigma}, y; r) = \sigma \wedge \text{Extract}(\tilde{\sigma}, \sigma, Y) = y]. \end{aligned}$$

Define $\eta_y(\sigma) := \Pr_{\tilde{\sigma}, r}[\text{Adapt}(\tilde{\sigma}, y; r) = \sigma \wedge \text{Extract}(\tilde{\sigma}, \sigma, Y) \neq y]$ and recall that $\Pr_{\tilde{\sigma}, r}[\text{Adapt}(\tilde{\sigma}, y; r) = \sigma] = \mathcal{D}_y(\sigma)$. The right-hand side equals $\mathcal{D}_y(\sigma) - \eta_y(\sigma)$. Define the subdistribution of faithfully extracted outputs by

$$G_Y(\sigma) := \frac{1}{N} \sum_{y \in W} (\mathcal{D}_y(\sigma) - \eta_y(\sigma)).$$

By Equation (2), $0 \leq G_Y(\sigma) \leq \mathcal{D}(\sigma)$. Let δ_Y be the faithfulness-failure probability conditioned on this key and statement Y , with y uniform in W . Then $\sum_{\sigma} G_Y(\sigma) = 1 - \delta_Y$. Summing the preceding fibre inequality over witnesses gives

$$\begin{aligned} \mathbb{E}_{\tilde{\sigma}}[\mathcal{D}(\mathcal{E}(\tilde{\sigma}, Y))] &\geq N \sum_{\sigma} \mathcal{D}(\sigma) G_Y(\sigma) \\ &\geq N \sum_{\sigma} G_Y(\sigma)^2 \\ &\geq \frac{N}{Q_B} (1 - \delta_Y)^2. \end{aligned}$$

The last step is Cauchy-Schwarz on the Q_B accepted signatures. Finally, average over honest keys and sampled statements. Witness regularity fixes N , and Definition 3.5 gives $\bar{\delta} := \mathbb{E}_{pk, sk, Y}[\delta_Y] \leq \delta$. The statistical hop and Jensen's inequality yield

$$\begin{aligned} \text{Adv}_{\mathcal{R}}^{\text{ow}}(\mathcal{A}) &\geq \frac{N}{Q_B} \mathbb{E}_{pk, sk, Y}[(1 - \delta_Y)^2] - \mu \\ &\geq \frac{N}{Q_B} (1 - \bar{\delta})^2 - \mu \geq \frac{N}{Q_B} (1 - \delta)^2 - \mu, \end{aligned}$$

which proves the claim. $\square$

3.3 Corollaries and Implications

Corollary 3.9. *Under the regularity and statistical unlinkability hypotheses of Theorem 3.8, if $\mathcal{R}$ is ϵ_{sow} -strongly one-way with $0 \leq \epsilon_{\text{sow}} \leq 1$, then there exists a PPT adversary $\mathcal{A}$ against its one-wayness such that*

$$\text{Adv}_{\mathcal{R}}^{\text{ow}}(\mathcal{A}) \geq \frac{N}{Q_B}(1 - \epsilon_{\text{sow}})^2 - \mu.$$

If ASig_B has perfectly faithful extraction, then

$$\text{Adv}_{\mathcal{R}}^{\text{ow}}(\mathcal{A}) \geq \frac{N}{Q_B} - \mu.$$

Proof. By Lemma 3.7, the proof of Theorem 3.8 applies with $\delta = \epsilon_{\text{sow}}$. Perfect faithfulness gives $\delta = 0$. $\square$

Remark 3.10 (Faithfulness vs. strong one-wayness in the lower bound). Theorem 3.8 uses faithfulness directly, which can hold even when strong one-wayness fails. For example, raw ECDSA is malleable because (r, s) and $(r, -s)$ both verify,³ while known ECDSA adaptors use inverse adaptation and extraction operations and are perfectly faithful [AEE⁺21, TZY22, HMR26]. Error-bearing lattice relations give another separation: they need not be strongly one-way, while the schemes of [EEE20] have perfectly faithful extraction.

Corollary 3.11. *Under the hypotheses of Theorem 3.8, assume that $\mathcal{R}$ is ϵ_{ow} -one-way and $\epsilon_{\text{ow}} + \mu > 0$. Then*

$$\frac{Q_B}{N} \geq \frac{(1 - \delta)^2}{\epsilon_{\text{ow}} + \mu}.$$

Consequently, if $\delta, \epsilon_{\text{ow}}, \mu$ are negligible, then Q_B/N is superpolynomial in λ . By the preceding corollary, the same conclusion holds when $\epsilon_{\text{sow}}, \epsilon_{\text{ow}}, \mu$ are negligible.

Remark 3.12. Faithfulness and statistical unlinkability remain important in the lower bound.

- For $N \gg Q_B$, pad the discrete-logarithm relation with a redundant component:

$$\text{Wit}(Y) = \{(y, z) : g^y = Y, z \in \mathbb{Z}_N\}.$$

If adaptation ignores z and extraction returns $z_0 = 0$, then $\delta = 1 - 1/N$. The bound becomes $Q_B \geq 1/(N(\epsilon_{\text{ow}} + \mu))$ and does not force expansion relative to the padded witness set.

- The Okamoto signature scheme has q^2 verifier-accepted signatures per message (see Sect. 2.2), although a modified signer fixing $r_1 = 0$ produces only q of them. Such concentration does not invalidate Theorem 3.8, and its bound requires no uniformity over the full accepted space.
- Large μ can make the bound vacuous. For example, append a bit set to 0 by signing and to 1 by adaptation, with verification accepting both and extraction rejecting bit 0. Then $\mu = 1$, and ordinary signatures never lie in the extractable set.

Corollary 3.13 ([EFH⁺21]). *A unique signature scheme, i.e., $Q_B = 1$, admits no adaptor signature extension for a hard witness-regular relation.*

Proof. Use the adversary from Theorem 3.8. Uniqueness makes its ordinary signature identical to the output of adapting its pre-signature with any valid witness. Extraction correctness therefore guarantees a valid witness, so its one-wayness advantage is 1. $\square$

³Low- s deployments reject one member of each pair.

Corollary 3.14. *Under the hypotheses of Corollary 3.11, if $\mathcal{R} = \mathcal{R}_A$ and Σ_A is Q_A -regular, then $N = Q_A$ and Q_B/Q_A is superpolynomial. Thus two regular schemes with comparable signature-space sizes, including the same scheme, are excluded under those hypotheses.*

3.4 Which Signature Pairs Are Feasible

We now read Corollaries 3.11 and 3.14 as a constraint on (Σ_A, Σ_B) in non-interactive swaps. If ASig_B is perfectly faithful, as are all constructions we are aware of, then

$$Q_B \geq \frac{N}{\epsilon_{ow} + \mu}.$$

For a concrete reading, suppose $\mu \approx 0$, extraction is perfectly faithful, and key-only one-way advantage is about $\epsilon_{ow} \approx 2^{-128}$. For $Y = (pk_A, m_A)$ and a Q_A -regular Σ_A , the requirement becomes

$$Q_B \geq Q_A \cdot 2^{128}.$$

BLS, Schnorr, and Okamoto are respectively 1-, q -, and q^2 -regular. The ECDSA row uses a raw, idealized counting model, with $\Theta(q)$ valid signatures and the approximation $Q \approx q$; rejection, coordinate reduction, sign symmetry, and low- s normalization affect the exact count in deployed variants. Table 2 summarizes the resulting conditional reading. Formally, its ECDSA entries apply the exact-regularity theorem to a raw, regularized model with $Q_A = \Theta(q)$; they do not assert that deployed ECDSA is exactly regular.

Table 2: Conditional feasibility in the direct signature-relation framework, at the 128-bit security level. $\times$ means that Corollary 3.11 excludes the direct signature-relation instantiation under its unlinkability, faithfulness, hardness, and regularity hypotheses. $\checkmark^*$ marks a construction proved here, and $\checkmark^?$ means not excluded by the bound but no proved construction here. The BLS $\times$ Schnorr star is realized by the pairing-Schnorr variant of Sect. 4.9, whose native response is a group element and whose commitment lies in the pairing target group; it is not ordinary scalar Schnorr.

Σ_A (adapter)	Q_A	Σ_B (pre-signer)			
		BLS 1	Schnorr $q \approx 2^{256}$	ECDSA $q \approx 2^{256}$	Okamoto $q^2 \approx 2^{512}$
BLS	1	$\times$	$\checkmark^*$	$\checkmark^?$	$\checkmark^?$
Schnorr	$q \approx 2^{256}$	$\times$	$\times$	$\times$	$\checkmark^?$
ECDSA	$\Theta(q)$	$\times$	$\times$	$\times$	$\checkmark^?$
Okamoto	$q^2 \approx 2^{512}$	$\times$	$\times$	$\times$	$\times$

The table concerns only the direct relation

$$\mathcal{R}_A = \{((pk_A, m_A), \sigma_A) : \text{Ver}_A(pk_A, m_A, \sigma_A) = 1\},$$

for which $N = |\text{Wit}(Y)| = Q_A$ when Alice's scheme is Q_A -regular.

Accordingly, in the raw, regularized ECDSA model, direct ECDSA $\times$ Schnorr has $N = Q_A = \Theta(q)$ and $Q_B = q$, hence $Q_B/N = \Theta(1)$. Corollary 3.11 excludes this pair under its stated hypotheses because its right-hand side is superpolynomial. If extraction is perfectly faithful, the required inequality is simply $Q_B/N \geq 1/(\epsilon_{ow} + \mu)$. This conclusion uses statistical unlinkability and the regularized counting model; it is unrelated to the PP-EUF assumption for the ECDSA preprocessing construction in Sect. 4.

- *Bob cannot use a unique signature scheme.* This is due to the impossibility of [EFH⁺21] and Corollary 3.13.
- *Uniqueness on Alice’s side.* With $Q_A = 1$, the cardinality bound leaves room for Bob schemes with sufficiently large signature spaces. This is only a necessary condition. The concrete scheme proved here is BLS $\times$ pairing-Schnorr, for $\mathcal{R}_{\text{pair}} = \{(Y, W) : e(W, g_2) = Y\}$.
- *Comparable signature spaces.* Under the hypotheses of Corollary 3.11, two Q -regular schemes with comparable signature-space sizes, and in particular the same scheme, are excluded for the direct signature relation.
- *Larger Bob signature spaces.* For Schnorr or idealized ECDSA, Alice with $q \approx 2^{256}$, the bound does not exclude, for example, Okamoto with roughly q^2 valid signatures or Schnorr over a much larger group. This is a legitimate reading of a necessary condition. However, we do not construct adaptors for these pairs.

From the Bound to the Constructions. The table applies the bound to the direct relation, whose witnesses range over Alice’s full signature space. As outlined in the introduction, Sect. 4 instead derives statements for simpler relations: fixed-base discrete logarithm for Schnorr and ECDSA, and pairing preimages for BLS with empty preprocessing. The generic construction does not require unique witnesses.

4 Atomic Signature Swaps with Preprocessing

Last but not least, we look at how we can extend the range of signature schemes that our non-interactive atomic swap framework supports. To this end we introduce atomic signature swaps with preprocessing, as abstracted in Fig. 2. Alice publishes one-time use slots ρ_i . For a request (m_A, m_B, i) , Bob derives $Y = \text{Stmt}(pk_A, \rho_i, m_A)$ and sends a pre-signature. Alice computes her online response w , adapts Bob’s presignature (into his native signature), and publishes this native signature. Bob then extracts a witness for Y and Alice’s native signature. We formalize this exchange and construct it from a *statement-derivable* signature scheme for Alice and an adaptor for Bob over the same relation.

4.1 Syntax and Security of Atomic Signature Swaps

Fix two signature schemes Σ_A, Σ_B , we denote their public parameters by pp_A and pp_B , and the combination of the two is $\text{pp} = (\text{pp}_A, \text{pp}_B)$.⁴ Without loss of generality, a secret key also stores its corresponding public key. All algorithms reject malformed encodings and invalid public keys.

Definition 4.1 (Atomic signature swap with preprocessing). An atomic signature swap with preprocessing **ASwap** consists of the following algorithms. The bundle size $\ell \geq 1$ is polynomial in the security parameter; $\tilde{\sigma}$ denotes Bob’s pre-signature.

- $\text{pp} \leftarrow \text{Setup}(1^\lambda, \Sigma_A, \Sigma_B)$. Generate compatible parameters for the fixed schemes Σ_A and Σ_B , and return $\text{pp} = (\text{pp}_A, \text{pp}_B)$.

⁴Earlier we presented the syntax of (adaptor) signatures without public parameters. Since this section composes two signature schemes within a single atomic swap, we now write them explicitly, for example, the description of the cyclic group underlying Schnorr.

- $(\text{aux}, \text{st}) \leftarrow \text{Offline}(sk_A, \ell)$. Generate an advertisement aux of ℓ indexed slots and private mutable state st , independently of the messages and counterparty. We denote this public *context* and a subsequent *swap request* by

$$\text{ctx} = (pk_A, pk_B, \text{aux}), \quad \text{req} = (m_A, m_B, i), \quad i \in [\ell].$$

- $b \leftarrow \text{ReqAdm}(\text{ctx}, \text{req})$. Return a bit b that is 1 exactly when the request is admissible for the context.
- $\tilde{\sigma} \leftarrow \text{PreSign}(sk_B, \text{ctx}, \text{req})$. Generate Bob's pre-signature for the exchange, or reject. Bob retains $(\text{ctx}, \text{req}, \tilde{\sigma})$.
- $b \leftarrow \text{PreVerify}(\text{ctx}, \text{req}, \tilde{\sigma})$. Return a bit b that is 1 exactly when the offer passes public pre-verification.
- $(\text{st}', \sigma_B) \leftarrow \text{Adapt}(sk_A, \text{st}, \text{ctx}, \text{req}, \tilde{\sigma})$. Update Alice's state and return σ_B for publication, or $\perp$ on failure. An aborted completion may still consume its slot.
- $\sigma_A \leftarrow \text{Ext}(\text{ctx}, \text{req}, \tilde{\sigma}, \sigma_B)$. Recover Alice's signature from Bob's retained offer and σ_B , or reject.

For fixed schemes Σ_A and Σ_B , we abbreviate $\text{Setup}(1^\lambda, \Sigma_A, \Sigma_B)$ as $\text{Setup}(1^\lambda)$. Rejection by a signature-producing algorithm returns $\perp$, and adaptation always returns its updated state as well.

We call a swap *secure* if it satisfies correctness, atomicity, and unforgeability (Defs. 4.2 to 4.4).

Notation and state conventions. Fix a swap Π , we write Gen_X , Sign_X , and Ver_X for the native algorithms of Σ_X , use dot notation for tuple fields, and take pp implicitly except in $\text{Gen}_X(\text{pp}_X)$. Alice's slots are one-time use across all contexts. Bob reserves each offered native message for at most one pre-signature; ordinary signing and offers use disjoint messages. A fresh swap or session identifier in m_B naturally separates distinct obligations. Throughout, security requirements quantify over PPT adversaries and polynomial bundle sizes ℓ where present, and negligibility is in λ .

We set $\text{Ver}_X(pk, m, \perp) = 0$. The predicate $\text{Unused}(\text{st}, i)$ reads the unused flag of slot i , and $\mathcal{M}_B$ is Bob's native message space. A failed **require** returns $\perp$ from the current oracle call. State changes are atomic and persist on failure. In Bob's game, L records offers, M reserves their native messages, and S records ordinarily signed messages; $\parallel$ appends a list entry. The set $\mathcal{P}$ records Alice messages with successful completions.

The games use honestly generated pp and permit a preliminary adversary stage before key generation, with state z and shared public-oracle access. Probabilities are over parameter and key generation and all algorithm and adversary coins, including any public random oracles. We omit λ and ℓ from advantage notation when clear.

Definition 4.2 (Correctness). For a PPT message-selecting adversary $\mathcal{D}$, define

$$\text{Adv}_{\Pi}^{\text{corr}}(\mathcal{D}; \lambda, \ell) := \Pr[\text{SwapCorr}_{\Pi, \mathcal{D}}(\lambda, \ell) = 1],$$

where the game $\text{SwapCorr}_{\Pi, \mathcal{D}}(\lambda, \ell)$ is defined in Fig. 8. The swap is *correct* if this probability is negligible, and *perfectly correct* if it is 0. The game detects either party failing to obtain a native valid signature in an adaptive sequence of admitted honest exchanges.

Experiment $\text{SwapCorr}_{\Pi, \mathcal{D}}(\lambda, \ell)$:

1. $\text{pp} \leftarrow \text{Setup}(1^\lambda)$; $z \leftarrow \mathcal{D}_0(\text{pp})$
2. $(pk_A, sk_A) \leftarrow \text{KeyGen}_A(\text{pp}_A)$; $(pk_B, sk_B) \leftarrow \text{KeyGen}_B(\text{pp}_B)$
3. $(\text{aux}, \text{st}) \leftarrow \text{Offline}(sk_A, \ell)$
4. $\text{ctx} := (pk_A, pk_B, \text{aux})$
5. $\mathcal{T} := ()$; $U := [\ell]$; $M := \emptyset$
6. **for** $j \in [\ell]$ **do**
7. $(\text{req}, z) \leftarrow \mathcal{D}_1(z, \text{ctx}, \mathcal{T})$
8. **if** $\text{req} = \perp \vee \text{ReqAdm}(\text{ctx}, \text{req}) = 0$ **then return** 0
9. $(m_A, m_B, i) := \text{req}$
10. **if** $i \notin U \vee m_B \in M$ **then return** 0
11. $U := U \setminus \{i\}$; $M := M \cup \{m_B\}$
12. $\tilde{\sigma} \leftarrow \text{PreSign}(sk_B, \text{ctx}, \text{req})$
13. **if** $\tilde{\sigma} = \perp \vee \text{PreVerify}(\text{ctx}, \text{req}, \tilde{\sigma}) = 0$ **then return** 1
14. $(\text{st}, \sigma_B) \leftarrow \text{Adapt}(sk_A, \text{st}, \text{ctx}, \text{req}, \tilde{\sigma})$
15. **if** $\sigma_B = \perp$ **then return** 1
16. $\sigma_A \leftarrow \text{Ext}(\text{ctx}, \text{req}, \tilde{\sigma}, \sigma_B)$
17. $v_A := \text{Ver}_A(pk_A, m_A, \sigma_A)$; $v_B := \text{Ver}_B(pk_B, m_B, \sigma_B)$
18. **if** $v_A = 0 \vee v_B = 0$ **then return** 1
19. $\mathcal{T} := \mathcal{T} \parallel ((\text{ctx}, \text{req}, \tilde{\sigma}, \sigma_B))$
20. **return** 0

Figure 8: Correctness experiment for an atomic signature swap with preprocessing.

Definition 4.3 (Unforgeability). For an adversary $\mathcal{A}$ controlling Bob, define

$$\text{Adv}_{\Pi}^{\text{swapUF}}(\mathcal{A}; \lambda, \ell) := \Pr[\text{SwapUF}_{\Pi, \mathcal{A}}(\lambda, \ell) = 1],$$

where the game $\text{SwapUF}_{\Pi, \mathcal{A}}(\lambda, \ell)$ is defined in Fig. 9. The swap is *unforgeable* if this probability is negligible. The adversary wins with a native valid Alice signature on a message for which no successful completion occurred.

Experiment $\text{SwapUF}_{\Pi, \mathcal{A}}(\lambda, \ell)$:

1. $\text{pp} \leftarrow \text{Setup}(1^\lambda)$
2. $z \leftarrow \mathcal{A}_0(\text{pp})$
3. $(pk_A, sk_A) \leftarrow \text{KeyGen}_A(\text{pp}_A)$
4. $(\text{aux}, \text{st}) \leftarrow \text{Offline}(sk_A, \ell)$
5. $\mathcal{P} := \emptyset$
6. $(m^*, \sigma^*) \leftarrow \mathcal{A}_1^{\text{COMPLETE}}(z, pk_A, \text{aux})$
7. $v := \text{Ver}_A(pk_A, m^*, \sigma^*)$
8. **return** $\llbracket v = 1 \wedge m^* \notin \mathcal{P} \rrbracket$

Oracle $\text{COMPLETE}(\text{ctx}, \text{req}, \tilde{\sigma})$:

9. **require** $\text{ReqAdm}(\text{ctx}, \text{req}) = 1$
10. **require** $\text{ctx}.pk_A = pk_A$
11. **require** $\text{ctx}.aux = \text{aux}$
12. **require** $\text{Unused}(\text{st}, \text{req}.i) = 1$
13. **require** $\text{PreVerify}(\text{ctx}, \text{req}, \tilde{\sigma}) = 1$
14. $(\text{st}, \sigma_B) \leftarrow \text{Adapt}(sk_A, \text{st}, \text{ctx}, \text{req}, \tilde{\sigma})$
15. $v_B := \text{Ver}_B(\text{ctx}.pk_B, \text{req}.m_B, \sigma_B)$
16. **require** $v_B = 1$
17. $\mathcal{P} := \mathcal{P} \cup \{\text{req}.m_A\}$
18. **return** σ_B

Figure 9: Unforgeability experiment (Alice's security) and completion oracle.

Definition 4.4 (Atomicity). For an adversary $\mathcal{A}$ controlling Alice, define

$$\text{Adv}_{\Pi}^{\text{swapAtom}}(\mathcal{A}; \lambda) := \Pr[\text{SwapAtom}_{\Pi, \mathcal{A}}(\lambda) = 1],$$

where the game $\text{SwapAtom}_{\Pi, \mathcal{A}}(\lambda)$ is defined in Fig. 10. The swap is *atomic* if this probability is negligible. The adversary wins with a native valid Bob signature for a recorded offer from which extraction fails to return a native valid Alice signature.

Experiment $\text{SwapAtom}_{\Pi, \mathcal{A}}(\lambda)$:	Oracle $\text{SWAPOFFER}(\text{ctx}, \text{req})$:
1. $\text{pp} \leftarrow \text{Setup}(1^\lambda)$	12. require $\text{ReqAdm}(\text{ctx}, \text{req}) = 1$
2. $z \leftarrow \mathcal{A}_0(\text{pp})$	13. require $\text{ctx}.pk_B = pk_B$
3. $(pk_B, sk_B) \leftarrow \text{KeyGen}_B(\text{pp}_B)$	14. require $\text{req}.m_B \notin M \cup S$
4. $L := (); M := \emptyset; S := \emptyset$	15. $M := M \cup \{\text{req}.m_B\}$
5. $(j, \sigma_B) \leftarrow \mathcal{A}_1^{\text{SWAPOFFER}, \text{SIGNOTHER}}(z, pk_B)$	16. $\tilde{\sigma} \leftarrow \text{PreSign}(sk_B, \text{ctx}, \text{req})$
6. if $j \notin [L]$ then return 0	17. require $\tilde{\sigma} \neq \perp$
7. $(\text{ctx}, \text{req}, \tilde{\sigma}) := L[j]$	18. require $\text{PreVerify}(\text{ctx}, \text{req}, \tilde{\sigma}) = 1$
8. $\sigma_A \leftarrow \text{Ext}(\text{ctx}, \text{req}, \tilde{\sigma}, \sigma_B)$	19. $L := L \parallel ((\text{ctx}, \text{req}, \tilde{\sigma}))$
9. $v_B := \text{Ver}_B(pk_B, \text{req}.m_B, \sigma_B)$	20. return $(L , \tilde{\sigma})$
10. $v_A := \text{Ver}_A(\text{ctx}.pk_A, \text{req}.m_A, \sigma_A)$	
11. return $\llbracket v_B = 1 \wedge v_A = 0 \rrbracket$	
	Oracle $\text{SIGNOTHER}(m)$:
	21. require $m \in \mathcal{M}_B \setminus M$
	22. $S := S \cup \{m\}$
	23. $\sigma \leftarrow \text{Sign}_B(sk_B, m)$
	24. return σ

Figure 10: Atomicity experiment (Bob’s security), offer oracle, and ordinary signing oracle.

Security scope. These are fresh-message signature-exchange guarantees for one polynomial-size preprocessing bundle. They do not cover an additional ordinary signing interface under Alice’s same key, bundle replenishment, state exposure or reset, corruption, delivery, or liveness. They also do not by themselves establish asset-level ledger atomicity. We discuss deployment requirements in Sect. 4.10.

4.2 Decomposable and Statement-Derivable Signatures

We now introduce *decomposability* and *statement-derivability*, the two key ingredients for Alice’s signatures.

Definition 4.5 (Decomposable signatures). Let $\Sigma = (\text{Gen}, \text{Sign}, \text{Ver})$ be a signature scheme. A decomposition equips it with algorithms

$$\begin{aligned} (\rho, \tau) &\leftarrow \text{OffSign}(sk), \\ b &\leftarrow \text{Adm}(pk, \rho, m), \\ w &\leftarrow \text{OnSign}(sk, \tau, m), \\ \sigma &\leftarrow \text{Assemble}(pk, \rho, m, w). \end{aligned}$$

Here $b \in \{0, 1\}$, and w and σ may be $\perp$. The public output ρ is message-independent. Each call of OffSign uses independent randomness and produces one private slot τ , which is one-time used.

The private slot stores or determines its associated public output ρ . Admission and assembly are public and deterministic.

The bit $\text{Adm}(pk, \rho, m)$ specifies whether the public slot ρ may be used to sign m under pk . The test includes native key and message validation, the format of ρ , and any construction-specific consistency or nondegeneracy checks. For a private slot associated with ρ , OnSign returns $\perp$ if $\text{Adm}(pk, \rho, m) = 0$. Assemble also rejects such triples.

We say that the scheme is *perfectly correct* if for every honest key pair, offline output (ρ, τ) , and admitted message m , every output $w \leftarrow \text{OnSign}(sk, \tau, m)$ satisfies $w \neq \perp$ and

$$\text{Ver}(pk, m, \text{Assemble}(pk, \rho, m, w)) = 1.$$

The following experiments use the public parameters and oracle conventions of Sect. 4.3.

Definition 4.6 ((Strong) unforgeability with preprocessing). Let Σ be a decomposable signature scheme. Let $t \in \{\text{euf}, \text{seuf}\}$ denote the ordinary and strong unforgeability variants, which we call PP-EUF and PP-sEUF, respectively. In Fig. 11, OFFSIGN exposes independent indexed public slots, and $\text{ONSIGN}(i, m)$ consumes an unused admitted slot and returns the raw online response w . The adversary wins with a native valid signature on a message never answered by ONSIGN (PP-EUF), or a message–signature pair distinct from all assembled oracle responses (PP-sEUF). Define

$$\text{Adv}_{\Sigma}^{\text{ppeuf}}(\mathcal{A}) = \Pr[\text{PPUF}_{\Sigma, \mathcal{A}}^t(\lambda) = 1],$$

and analogously define the strong advantage using $t = \text{seuf}$. We say that Σ is *PP-EUF* if $\text{Adv}_{\Sigma}^{\text{ppeuf}}(\mathcal{A})$ is negligible for every PPT $\mathcal{A}$, and *PP-sEUF* if the $\text{Adv}_{\Sigma}^{\text{ppseuf}}(\mathcal{A})$ advantage is negligible.

<u>Experiment $\text{PPUF}_{\Sigma, \mathcal{A}}^t(\lambda)$:</u>	<u>Oracle $\text{OFFSIGN}()$:</u>
1. $\text{pp} \leftarrow \text{Setup}(1^\lambda)$	11. $\text{ctr} := \text{ctr} + 1$
2. $z \leftarrow \mathcal{A}_0(\text{pp})$	12. $(\rho, \tau) \leftarrow \text{OffSign}(sk)$
3. $(pk, sk) \leftarrow \text{KeyGen}(\text{pp})$	13. $L_{\text{off}}[\text{ctr}] := (\rho, \tau)$
4. $\text{ctr} := 0; S := \emptyset; L_{\text{off}}[\cdot] := \perp$	14. return ρ
5. $(m^*, \sigma^*) \leftarrow \mathcal{A}_1^{\text{OFFSIGN}, \text{ONSIGN}}(z, pk)$	<u>Oracle $\text{ONSIGN}(i, m)$:</u>
6. if $\text{Ver}(pk, m^*, \sigma^*) = 0$ then	15. require $L_{\text{off}}[i] \neq \perp$
7. return 0	16. $(\rho, \tau) := L_{\text{off}}[i]$
8. if $t = \text{euf}$ then	17. require $\text{Adm}(pk, \rho, m) = 1$
9. return $\llbracket (m^*, \cdot) \notin S \rrbracket$	18. $L_{\text{off}}[i] := \perp$
10. return $\llbracket (m^*, \sigma^*) \notin S \rrbracket$	19. $w \leftarrow \text{OnSign}(sk, \tau, m); \tau := \perp$
	20. $\sigma := \text{Assemble}(pk, \rho, m, w)$
	21. $S := S \cup \{(m, \sigma)\}$
	22. return w

Figure 11: Unforgeability with preprocessing: ordinary ($t = \text{euf}$) and strong ($t = \text{seuf}$) variants.

Batch Equivalence. For independently generated slots, adaptive offline queries are equivalent to initial exposure of any polynomial-size batch: one direction reveals precomputed slots on demand, and the other requests the whole batch initially. We use either formulation interchangeably.

Definition 4.7 (Statement-derivable signatures). Let $\mathcal{R}$ be an efficiently decidable relation. A decomposable signature scheme is also *statement-derivable for $\mathcal{R}$* if it has a public, deterministic algorithm

$$Y \leftarrow \text{Stmt}(pk, \rho, m).$$

The output may be a failure symbol $\perp$, which is neither a statement nor a witness, and we say $\mathcal{R}(Y, w) = 0$ whenever $Y = \perp$ or $w = \perp$. We require the following two properties.

1. *Admissible message consistency.* For every honest key pk , offline output ρ , and message m , if $\text{Adm}(pk, \rho, m) = 0$, then $\text{Stmt}(pk, \rho, m) = \perp$.
2. *Honest response membership.* For every honest keypair sk, pk , offline output (ρ, τ) , admitted message m , and $w \leftarrow \text{OnSign}(sk, \tau, m)$,

$$\mathcal{R}(\text{Stmt}(pk, \rho, m), w) = 1.$$

Definition 4.8 (Sound verification collapse). Let $\mathcal{R}$ be an efficiently decidable relation. Define

$$\text{Adv}_{\Sigma}^{\text{collapse}}(\mathcal{D}) := \Pr[\text{Collapse}_{\Sigma, \mathcal{D}}(\lambda) = 1],$$

where the game $\text{Collapse}_{\Sigma, \mathcal{D}}(\lambda)$ is defined in Fig. 12. The adversary chooses arbitrary (pk, ρ, m, w) and wins if w is a valid witness for $\text{Stmt}(pk, \rho, m)$ but $\text{Assemble}(pk, \rho, m, w)$ fails native verification. A statement-derivable signature scheme has *sound verification collapse for $\mathcal{R}$* if this probability is negligible in λ for every PPT $\mathcal{D}$. Collapse is *perfect* if this probability is always 0.

Experiment $\text{Collapse}_{\Sigma, \mathcal{D}}(\lambda)$:

1. $\text{pp} \leftarrow \text{Setup}(1^\lambda)$
2. $(pk, \rho, m, w) \leftarrow \mathcal{D}(\text{pp})$
3. $Y := \text{Stmt}(pk, \rho, m)$
4. **if** $Y = \perp$ **then return** 0
5. **if** $\mathcal{R}(Y, w) = 0$ **then return** 0
6. $\sigma := \text{Assemble}(pk, \rho, m, w)$
7. **return** $\llbracket \text{Ver}(pk, m, \sigma) = 0 \rrbracket$

Figure 12: Sound verification collapse for a statement-derivable signature scheme.

Notice that, for honest slots, PP-EUF implies hardness of finding such a witness for a fresh message, since public assembly would turn it into a forgery.

4.3 Generic Construction and Security

The construction composes Alice’s statement derivation with Bob’s adaptor over the same relation.

Construction 4.9 (Generic swap). Let Σ_A be a statement-derivable signature scheme for an efficiently decidable relation $\mathcal{R}$, and let Σ_B be a native signature scheme that admits an adaptor scheme ASig_B for $\mathcal{R}$. Write

$$\text{ctx} = (pk_A, pk_B, \text{aux}), \quad \text{req} = (m_A, m_B, i).$$

On admitted inputs, each algorithm derives $Y = \text{Stmt}_A(pk_A, \rho_i, m_A)$ from its context and request.

- $\text{pp} \leftarrow \text{Setup}(1^\lambda)$. Generate compatible common parameters $\text{pp} = (\text{pp}_A, \text{pp}_B)$.
- $(\text{aux}, \text{st}) \leftarrow \text{Offline}(sk_A, \ell)$. Independently run $(\rho_i, \tau_i) \leftarrow \text{OffSign}_A(sk_A)$ for $i \in [\ell]$. Return $\text{aux} = (\ell, \rho_1, \dots, \rho_\ell)$ and state st storing aux , the private slots, and their unused flags.
- $b \leftarrow \text{ReqAdm}(\text{ctx}, \text{req})$. Parse both tuples and check that pk_B and m_B are valid native inputs, aux encodes a bundle $(\ell, \rho_1, \dots, \rho_\ell)$ with $\ell \geq 1$, and $i \in [\ell]$. Also check $\text{Adm}_A(pk_A, \rho_i, m_A) = 1$ and $\text{Stmt}_A(pk_A, \rho_i, m_A) \neq \perp$. Return 1 iff all checks pass.
- $\tilde{\sigma} \leftarrow \text{PreSign}(sk_B, \text{ctx}, \text{req})$. Reject if $\text{ReqAdm}(\text{ctx}, \text{req}) = 0$ or pk_B differs from the public key stored in sk_B . Return $\tilde{\sigma} \leftarrow \text{pSign}_B(sk_B, m_B, Y)$.
- $b \leftarrow \text{PreVerify}(\text{ctx}, \text{req}, \tilde{\sigma})$. Return 0 if $\text{ReqAdm}(\text{ctx}, \text{req}) = 0$; otherwise return

$$\text{pVer}_B(pk_B, m_B, Y, \tilde{\sigma}).$$

- $(\text{st}', \sigma_B) \leftarrow \text{Adapt}(sk_A, \text{st}, \text{ctx}, \text{req}, \tilde{\sigma})$. Require $\text{PreVerify}(\text{ctx}, \text{req}, \tilde{\sigma}) = 1$, pk_A equal to the key stored in sk_A , aux equal to the stored advertisement, and slot i unused; otherwise return $(\text{st}, \perp)$. Atomically consume i *before* computing

$$\begin{aligned} w &\leftarrow \text{OnSign}_A(sk_A, \tau_i, m_A), \\ \sigma_B &\leftarrow \text{Adapt}_B(pk_B, m_B, \tilde{\sigma}, w). \end{aligned}$$

Erase τ_i , obtaining st' . Return (st', σ_B) if $\text{Ver}_B(pk_B, m_B, \sigma_B) = 1$, and $(\text{st}', \perp)$ otherwise.

- $\sigma_A \leftarrow \text{Ext}(\text{ctx}, \text{req}, \tilde{\sigma}, \sigma_B)$. Reject unless all three checks hold:

$$\text{ReqAdm}(\text{ctx}, \text{req}) = 1, \text{PreVerify}(\text{ctx}, \text{req}, \tilde{\sigma}) = 1, \text{Ver}_B(pk_B, m_B, \sigma_B) = 1.$$

Compute $w \leftarrow \text{Extract}_B(pk_B, m_B, Y, \tilde{\sigma}, \sigma_B)$ and reject unless $\mathcal{R}(Y, w) = 1$. Set $\sigma_A := \text{Assemble}_A(pk_A, \rho_i, m_A, w)$. Return σ_A if $\text{Ver}_A(pk_A, m_A, \sigma_A) = 1$, or $\perp$ otherwise.

Write $\Pi = \text{ASwap}(\Sigma_A, \Sigma_B, \mathcal{R})$ for the resulting atomic signature swap with preprocessing, with the above choice of ASig_B understood.

Remark 4.10 (Index selection). Multiple parties may offer for one slot, but Alice can complete at most one; an additional reservation mechanism is needed.

Theorem 4.11 (Generic composition). *Let $\mathcal{R}$ be an efficiently decidable relation. Let Σ_A be a statement-derivable signature scheme for $\mathcal{R}$, and let ASig_B be an adaptor signature for $\mathcal{R}$ with native signature scheme Σ_B . Suppose that Σ_A is PP-EUF and has sound verification collapse, and that ASig_B has honest correctness, deterministic pre-verification, full extractability, and exact pre-signature adaptability. Model conventions and the reductions are given directly below. Then the generic construction $\Pi = \text{ASwap}(\Sigma_A, \Sigma_B, \mathcal{R})$ is a secure atomic signature swap with preprocessing, i.e. it is correct, unforgeable, and atomic. More precisely, for every PPT $\mathcal{D}, \mathcal{A}$ and polynomial bundle size ℓ , there are PPT reductions satisfying*

$$\begin{aligned} \text{Adv}_{\Pi}^{\text{corr}}(\mathcal{D}; \lambda, \ell) &\leq \text{Adv}_{\Sigma_A}^{\text{collapse}}(\mathcal{B}_{\text{corr}}), \\ \text{Adv}_{\Pi}^{\text{swapUF}}(\mathcal{A}; \lambda, \ell) &\leq \text{Adv}_{\Sigma_A}^{\text{ppeuf}}(\mathcal{B}_{\text{pp}}), \\ \text{Adv}_{\Pi}^{\text{swapAtom}}(\mathcal{A}) &\leq \text{Adv}_{\text{ASig}_B}^{\text{fext}}(\mathcal{B}_{\text{ext}}) + \text{Adv}_{\Sigma_A}^{\text{collapse}}(\mathcal{B}_{\text{collapse}}). \end{aligned}$$

If collapse is perfect, Π is perfectly correct. The reduction $\mathcal{B}_{\text{pp}}$ makes ℓ offline queries and at most ℓ online queries.

Proof. The theorem follows from Propositions 4.12 to 4.14, which establish correctness, unforgeability, and atomicity of the generic construction under the stated assumptions. $\square$

Let Π be Construction 4.9, with the chosen adaptor ASig_B and the common model specified below. We prove the three security propositions separately.

Common Model. All component assumptions use the same joint pp , encodings, and public or random oracles as the swap. They allow arbitrary polynomial preprocessing before the challenge key is generated, and every stage counts toward the stated query bounds. Each reduction preserves the challenger’s parameters, key, and oracle state; Bob’s full-extractability game is understood with this same preliminary stage.

4.3.1 Correctness

Proposition 4.12 (Correctness). *Suppose Alice’s Σ_A has honest response membership and sound verification collapse, as in Defs. 4.7 and 4.8. Suppose Bob’s adaptor ASig_B has deterministic pre-verification and honest correctness. Then Π is correct; in particular, for every PPT $\mathcal{D}$ and polynomial bundle size ℓ , there is a PPT reduction $\mathcal{B}_{\text{corr}}$ such that*

$$\text{Adv}_{\Pi}^{\text{corr}}(\mathcal{D}; \lambda, \ell) \leq \text{Adv}_{\Sigma_A}^{\text{collapse}}(\mathcal{B}_{\text{corr}}).$$

If collapse is perfect, Π is perfectly correct.

of Proposition 4.12. The collapse reduction $\mathcal{B}_{\text{corr}}$ runs $\mathcal{D}_0(\text{pp})$, generates both honest keys and Alice’s bundle, and simulates Fig. 8. On each admitted unused slot, honest response membership gives $\mathcal{R}(Y, w) = 1$, hence $w \neq \perp$. Honest adaptor correctness gives a native valid σ_B and an extracted witness w' with $\mathcal{R}(Y, w') = 1$. The only remaining failure is

$$\text{Ver}_A(pk_A, m_A, \text{Assemble}_A(pk_A, \rho_i, m_A, w')) = 0.$$

At the first such failure the reduction outputs (pk_A, ρ_i, m_A, w') ; deterministic statement derivation and assembly make it a winning collapse tuple. This detects every correctness failure without guessing a slot, proving the bound and the perfect-correctness claim. $\square$

4.3.2 Unforgeability

Proposition 4.13 (Unforgeability). *Suppose Alice’s Σ_A is statement-derivable (Def. 4.7) and PP-EUF (Def. 4.6). Suppose Bob’s adaptor ASig_B has deterministic pre-verification and exact pre-signature adaptability (Def. 2.8). Then Π is unforgeable; in particular, for every PPT $\mathcal{A}$ and polynomial bundle size ℓ , there is a PPT reduction $\mathcal{B}_{\text{pp}}$ such that*

$$\text{Adv}_{\Pi}^{\text{swapUF}}(\mathcal{A}; \lambda, \ell) \leq \text{Adv}_{\Sigma_A}^{\text{ppeuf}}(\mathcal{B}_{\text{pp}}).$$

The reduction makes ℓ offline queries and at most ℓ online queries.

of Proposition 4.13. The reduction $\mathcal{B}_{\text{pp}}$ in Fig. 13 forwards the PP-EUF challenge key and obtains ℓ independent public slots from OFFSIGN. Its preliminary stage runs $\mathcal{A}_0$. For each completion passing the public and unused-slot checks, it consumes the same slot as the challenger, obtains w from $\text{ONSIGN}(i, m_A)$, and runs Bob’s public adaptation. This exactly simulates Construction 4.9, including failed calls.

The crucial invariant is equality between the swap's paid-message set and the message projection of the PP-EUF signature history S . Every legal response query has

$$\mathcal{R}(Y, w) = 1, \quad \text{pVer}_B(pk_B, m_B, Y, \tilde{\sigma}) = 1.$$

Pre-signature adaptability, including for malicious Bob keys and offers, therefore gives

$$\text{Ver}_B(pk_B, m_B, \text{Adapt}_B(pk_B, m_B, \tilde{\sigma}, w)) = 1.$$

Thus every response records and pays the same m_A , while a call rejected before signing changes neither set:

$$\mathcal{P} = \{m : \exists \sigma (m, \sigma) \in S\}.$$

A winning swap output is consequently a fresh-message PP-EUF forgery. The simulation makes ℓ offline and at most ℓ online queries, with no loss in success probability.

$\mathcal{B}_{\text{pp},0}(\text{pp})$:	Oracle COMPLETE(ctx, req, $\tilde{\sigma}$):
1. $z \leftarrow \mathcal{A}_0(\text{pp})$	9. require ReqAdm(ctx, req) = 1
2. return (pp, z)	10. require ctx.pk _A = pk _A
$\mathcal{B}_{\text{pp},1}((\text{pp}, z), pk_A)$:	11. require ctx.aux = aux
3. for $i \in [\ell]$ do	12. $(m_A, m_B, i) := \text{req}; pk_B := \text{ctx.pk}_B$
4. $\rho_i \leftarrow \text{OFFSIGN}()$	13. require $i \in U$
5. $\text{aux} := (\ell, \rho_1, \dots, \rho_\ell)$	14. require PreVerify(ctx, req, $\tilde{\sigma}$) = 1
6. $U := [\ell]; \mathcal{P} := \emptyset$	15. $U := U \setminus \{i\}$
7. $(m^*, \sigma^*) \leftarrow \mathcal{A}_1^{\text{COMPLETE}}(z, pk_A, \text{aux})$	16. $w \leftarrow \text{ONSIGN}(i, m_A)$
8. return (m^*, σ^*)	17. $\sigma_B \leftarrow \text{Adapt}_B(pk_B, m_B, \tilde{\sigma}, w)$
	18. $v_B := \text{Ver}_B(pk_B, m_B, \sigma_B)$
	19. require $v_B = 1$
	20. $\mathcal{P} := \mathcal{P} \cup \{m_A\}$
	21. return σ_B

Figure 13: Reduction from Alice's swap security to PP-EUF. The second stage has access to the challenger's OFFSIGN and ONSIGN oracles.

Adaptability is needed for freshness: a failed adaptation after an online query would leave m_A recorded by PP-EUF but unpaid in the swap. $\square$

4.3.3 Atomicity

Proposition 4.14 (Atomicity). *Suppose Alice's statement-derivable signature scheme Σ_A has sound verification collapse (Def. 4.8), and Bob's adaptor ASig_B has deterministic pre-verification and full extractability (Def. 2.7). Then the protocol Π is atomic; in particular, for every PPT $\mathcal{A}$, there are PPT reductions $\mathcal{B}_{\text{ext}}$ and $\mathcal{B}_{\text{collapse}}$ such that*

$$\text{Adv}_{\Pi}^{\text{swapAtom}}(\mathcal{A}) \leq \text{Adv}_{\text{ASig}_B}^{\text{fext}}(\mathcal{B}_{\text{ext}}) + \text{Adv}_{\Sigma_A}^{\text{collapse}}(\mathcal{B}_{\text{collapse}}).$$

of Proposition 4.14. A valid target signature can cause exactly two failures: extraction returns no $\mathcal{R}$ -witness, or it returns a witness whose assembly fails native verification. For the recorded target offer (ctx, req, $\tilde{\sigma}$), write

$$Y = \text{Stmt}_A(pk_A, \rho_i, m_A), \quad w = \text{Extract}_B(pk_B, m_B, Y, \tilde{\sigma}, \sigma_B).$$

Admission and pre-verification are deterministic and passed when the offer was recorded, so $Y \neq \perp$ and these checks cannot newly fail. Let Bad_{ext} denote a valid target with $\mathcal{R}(Y, w) = 0$, and $\text{Bad}_{\text{collapse}}$ a valid target with $\mathcal{R}(Y, w) = 1$ but invalid assembly. The winning event is their disjoint union.

For Bad_{ext} , the reduction $\mathcal{B}_{\text{ext}}$ uses the component challenger's Bob key and the preliminary stage $(\text{pp}, \mathcal{A}_0(\text{pp}))$. It simulates Fig. 10, replacing each legal PreSign call by $\text{PSIGN}(m_B, Y)$ and each legal $\text{SignOther}(m)$ call by $\text{SIGN}(m)$. All admission checks, reservations, failed-call behavior, and offer-list updates are preserved. For the reserved target message, the component histories satisfy

$$m_B \notin \mathcal{S}, \quad \mathcal{T}[m_B] = \{(Y, \tilde{\sigma})\}.$$

The unique recorded offer is therefore the sole entry in the full-extractability test, and Bad_{ext} wins that game. Indeed, full extractability only promises that extraction succeeds for some recorded pre-signature on the target message; the reservation rule makes that entry the offer under attack. Each swap-oracle call uses at most one corresponding component query.

For $\text{Bad}_{\text{collapse}}$, the reduction $\mathcal{B}_{\text{collapse}}$ generates Bob's key locally and simulates both swap oracles. On that event it returns (pk_A, ρ_i, m_A, w) , a winning collapse tuple even if Alice's key and advertisement were malicious. The two reductions in Fig. 14 therefore give

$$\begin{aligned} \Pr[\text{Bad}_{\text{ext}}] &\leq \text{Adv}_{\text{ASig}_B}^{\text{fext}}(\mathcal{B}_{\text{ext}}), \\ \Pr[\text{Bad}_{\text{collapse}}] &\leq \text{Adv}_{\Sigma_A}^{\text{collapse}}(\mathcal{B}_{\text{collapse}}). \end{aligned}$$

Their sum is the claimed bound, with no guessed-offer loss.

$\mathcal{B}_{\text{ext},1}((\text{pp}, z), pk_B):$	$\mathcal{B}_{\text{collapse}}(\text{pp}):$
1. $L := (); M := \emptyset; S := \emptyset$	6. $z \leftarrow \mathcal{A}_0(\text{pp})$
2. $(j, \sigma_B) \leftarrow \mathcal{A}_1^{\text{SWAPOFFER}, \text{SIGNOTHER}}(z, pk_B)$	7. $(pk_B, sk_B) \leftarrow \text{KeyGen}_B(\text{pp}_B)$
3. require $j \in [L]$	8. $L := (); M := \emptyset; S := \emptyset$
4. $(\text{ctx}, \text{req}, \tilde{\sigma}) := L[j]$	9. $(j, \sigma_B) \leftarrow \mathcal{A}_1^{\text{SWAPOFFER}, \text{SIGNOTHER}}(z, pk_B)$
5. return $(\text{req}.m_B, \sigma_B)$	10. require $j \in [L]$
	11. $(\text{ctx}, \text{req}, \tilde{\sigma}) := L[j]$
	12. $(pk_A, pk_B, \text{aux}) := \text{ctx}$
	13. $(\ell, \rho_1, \dots, \rho_\ell) := \text{aux}$
	14. $(m_A, m_B, i) := \text{req}$
	15. require $\text{Ver}_B(pk_B, m_B, \sigma_B) = 1$
	16. $Y := \text{Stmt}_A(pk_A, \rho_i, m_A)$
	17. $w := \text{Extract}_B(pk_B, m_B, Y, \tilde{\sigma}, \sigma_B)$
	18. require $Y \neq \perp \wedge \mathcal{R}(Y, w) = 1$
	19. $\sigma_A := \text{Assemble}_A(pk_A, \rho_i, m_A, w)$
	20. require $\text{Ver}_A(pk_A, m_A, \sigma_A) = 0$
	21. return (pk_A, ρ_i, m_A, w)

Figure 14: Reductions for Bob's security. The left reduction uses the component signing and pre-signing oracles; the right reduction generates Bob's key and answers locally. Oracle substitutions are specified in the proof.

□

PP-EUF protects Alice against misuse of public preprocessing. Exact adaptability makes each consumed online response a successful Bob completion, preserving message freshness in the PP-EUF reduction. Full extractability gives Bob a valid witness; sound collapse turns it into a native Alice signature even for malicious advertisements. Neither unlinkability nor pre-verify soundness is needed for these guarantees.

4.4 Instantiations Overview

Our instantiations use two relations: $\mathcal{R}_{\text{DL}}$, given by $Y = g^w$ in a prime-order group, and $\mathcal{R}_{\text{pair}}$, given by $Y = e(W, g_2)$ for a bilinear pairing. The table records Alice’s preprocessing security and Bob’s adaptor; Schnorr and conditional ECDSA use the first relation, and BLS uses the second with empty preprocessing.

Relation	Alice	Alice’s security	Bob	Bob’s adaptor
$\mathcal{R}_{\text{DL}}$	Schnorr	OMDL in ROM (Sect. 4.7)	Schnorr	Standard [AEE+21]
$\mathcal{R}_{\text{DL}}$	ECDSA	Conditional PP-EUF (Sect. 4.8)	Schnorr	Standard [AEE+21]
$\mathcal{R}_{\text{pair}}$	BLS	Native BLS EUF-CMA (Sect. 4.9)	Pairing-Schnorr	Sect. 4.6, [GSST24]

4.5 Bob’s Adaptors for Canonical Relations

For a public, efficiently computable function $F : D \rightarrow T$, define

$$\mathcal{R}_F = \{(Y, w) \in T \times D : F(w) = Y\}.$$

We focus on efficiently represented finite abelian groups D, T and homomorphisms F , so $F(tw) = F(t)F(w)$. We use multiplicative notation, except for scalar domains. The function is one-way if finding a preimage of $F(w)$ for uniformly sampled $w \in D$ is computationally hard. Both swap components must use the same function, parameters, and encodings.

Schnorr and the Discrete Logarithm Relation. For a prime-order group $G = \langle g \rangle$, take $D = (\mathbb{Z}_q, +)$, $T = G$, and $F_{\text{DL}}(w) = g^w$. Then

$$\mathcal{R}_{\text{DL}} = \mathcal{R}_{F_{\text{DL}}} = \{(Y, w) \in G \times \mathbb{Z}_q : g^w = Y\}.$$

Bob’s native Schnorr key is $B = g^b$, for $b \leftarrow \mathbb{Z}_q^*$, and verification checks $g^{s_B} = C_B B^{H_B(C_B, B, m_B)}$. The standard Schnorr adaptor [AEE+21, GSST24] pre-signs by sampling $r_B \leftarrow \mathbb{Z}_q$ and returning

$$C_B = g^{r_B} Y, \quad \tilde{\sigma} = (C_B, \hat{s}_B), \quad \hat{s}_B = r_B + H_B(C_B, B, m_B) b.$$

Pre-verification checks $g^{\hat{s}_B} = (C_B/Y) B^{H_B(C_B, B, m_B)}$; adaptation adds w , and extraction subtracts $\hat{s}_B$ from a valid full response with matching commitment. We use the known security of this adaptor under DL in the ROM, with canonical encodings and validated inputs.

Pairing-Schnorr. For a bijective homomorphism $F : D \rightarrow T$ between prime-order cyclic groups, the same pattern gives a Schnorr-type signature. Choose $K \leftarrow D \setminus \{1\}$ and publish $P = F(K)$. With $T_0 \leftarrow D$, set

$$C = F(T_0), \quad c = H_B(C, P, m), \quad S = T_0 K^c, \quad F(S) = CP^c.$$

For $F_{\text{pair}}(W) = e(W, g_2) : G_1 \rightarrow G_T$, this gives *pairing-Schnorr*. Its adaptor for $\mathcal{R}_{\text{pair}} = \{(Y, W) : F_{\text{pair}}(W) = Y\}$ uses

$$\tilde{C} = F_{\text{pair}}(T_0)Y, \quad c = H_B(\tilde{C}, P, m), \quad \tilde{S} = T_0K^c,$$

then adapts as $S = \tilde{S}W$ and extracts $W = S/\tilde{S}$ from a valid signature with matching commitment.

Relation to Prior Work. These adaptors follow the witness-shift pattern of Gerhart et al. [GSST24]: multiply the commitment image by Y , translate the response by a witness, and extract by division or subtraction. Their full-version Lemma 17 identifies suitable commitment-recoverable Fiat–Shamir schemes as dichotomic, and Theorem 5 supplies a simulatable transparent reduction. The compiler applies once its hypotheses are matched, including a quasi-injective homomorphic one-way function and native strong unforgeability. Related pairing-based Schnorr signatures appear in [GV13, WHL24]; the latter’s SIG_{DDH} uses a source-group commitment, whereas our $C = F_{\text{pair}}(T_0) \in G_T$ lets Bob fold Y directly into it. Sect. 4.6 proves native strong unforgeability and full extractability directly for our format and games.

4.6 Pairing-Schnorr Signatures and Adaptors

The identification-based dichotomic framework of Gerhart et al. [GSST24] also yields this adaptor (full version, Lemma 17 and Theorem 5), together with their compiler, after matching its hypotheses to our common model. We give a direct proof because it is short and establishes exactly the native strong unforgeability and full extractability used here.

Let $e : G_1 \times G_2 \rightarrow G_T$ be the nondegenerate prime-order pairing of Sect. 4.9, and write $F = F_{\text{pair}}$, where $F_{\text{pair}}(W) = e(W, g_2)$. This is a bijective homomorphism. We work in the common model of Sect. 4.3, with $H_B(C, P, m) \in \mathbb{Z}_q$ a random oracle. All algorithms validate canonical subgroup encodings; public keys must be nonidentity, while identity commitments and statements are allowed.

Native Signatures. $(P, K) \leftarrow \text{KeyGen}_B(\text{pp}_B)$. Sample $K \leftarrow G_1 \setminus \{1\}$ and return (P, K) with $P = F(K)$. For $\sigma \leftarrow \text{Sign}_B(K, m)$, sample $T_0 \leftarrow G_1$ and return $\sigma = (C, S)$, where

$$C = F(T_0), \quad c = H_B(C, P, m), \quad S = T_0K^c.$$

For $v \leftarrow \text{Ver}_B(P, m, \sigma)$, return the bit testing $F(S) = CP^c$. This is dichotomic for F , with $\Sigma_1(K, m; Z) = Z$ and $\Sigma_2(K, m, C; t) = tK^{H_B(C, P, m)}$.

Adaptor algorithms. For $\tilde{\sigma} \leftarrow \text{pSign}_B(K, m, Y)$ with $Y \in G_T$, sample $T_0 \leftarrow G_1$ and return $\tilde{\sigma} = (C, \hat{S})$, where

$$C = F(T_0)Y, \quad c = H_B(C, P, m), \quad \hat{S} = T_0K^c.$$

For $v \leftarrow \text{pVer}_B(P, m, Y, \tilde{\sigma})$, return the bit testing $F(\hat{S}) = (C/Y)P^c$. For $\sigma \leftarrow \text{Adapt}_B(P, m, \tilde{\sigma}, W)$, return $(C, \hat{S}W)$. For $W \leftarrow \text{Extract}_B(P, m, Y, \tilde{\sigma}, \sigma)$, first verify the full signature $\sigma = (C', S)$ and the pre-signature. If $C' = C$, return $W = S/\hat{S}$; otherwise return $\perp$.

Lemma 4.15 (Pairing-Schnorr security). *The native signature and adaptor are perfectly correct on honest inputs. The adaptor’s pre-verification is deterministic, and its pre-signature adaptability is exact. If inverting F on a uniformly random nonidentity image is hard, the native signature is strongly unforgeable and the adaptor has full extractability in the ROM, including with multiple pre-signatures per message.*

Proof. Correctness and adaptability. Homomorphism gives native correctness and, for any admitted key, accepted pre-signature, and witness W ,

$$F(\widehat{S}W) = (C/Y)P^cY = CP^c, \quad F(S/\widehat{S}) = Y$$

whenever the full signature is valid with matching commitment. These identities also prove exact adaptability and honest extraction.

Simulation. Given an inversion challenge P , sample independent $c \leftarrow \mathbb{Z}_q$ and $\widehat{S} \leftarrow G_1$, set $C = F(\widehat{S})P^{-c}Y$, and program $H_B(C, P, m) = c$. This simulates a pre-signature for any Y ; native signing uses $Y = 1$. The uniform commitment hits a previously assigned input with probability at most Q_H/q per call, where Q_H counts all hash assignments, including preliminary queries. Outside these negligible collisions, the simulation is exact.

The target query. For native strong unforgeability, a winning signature cannot have the same commitment and message as an oracle-issued signature: injectivity of F would force the same response and hence the identical signature. For full extractability, the winning message was not ordinarily signed, and its commitment differs from every pre-signature issued for that message; a matching entry would extract a witness by the first part. Thus, in either game, the target verification input was not programmed by the simulator. Except with probability $1/q$, the adversary queried it before outputting the signature.

Forking. The general forking lemma [BN06], rewinding at that query, gives two accepting signatures $(C^*, S), (C^*, S')$ with distinct challenges c, c' and polynomial loss. Their verification equations give

$$K = (S/S')^{(c-c')^{-1}}, \quad F(K) = P,$$

contradicting pairing-preimage hardness. The simulation-collision bound applies to both executions. This proves both security claims with the same simulation and extraction argument. $\square$

4.7 Schnorr Alice for the Discrete-Logarithm Relation

Over $G = \langle g \rangle$ of prime order q , we make Schnorr statement-derivable for $\mathcal{R}_{DL}$ using two independent nonces per slot. This technique follows MuSig2 [NRS21] and is necessary to avoid a preprocessing attack on single public nonces.

Recall that Alice generates $(A, a) \leftarrow \text{KeyGen}_A(\text{pp}_A)$ by sampling $a \leftarrow \mathbb{Z}_q^*$ and setting $A = g^a$. She uses the native Schnorr verifier $g^s = RA^{H_{\text{sig}}(R, A, m)}$ for signatures (R, s) . The independent random oracles H_{non} and H_{sig} output scalars in $\mathbb{Z}_q$.

An Attack on Single Public Nonces. For public nonces $R_i = g^{r_i}$, set $R^* = \prod_i R_i$. If a fresh m^* and signing messages m_i satisfy $c^* = H_{\text{sig}}(R^*, A, m^*) = \sum_i c_i$ with $c_i = H_{\text{sig}}(R_i, A, m_i)$, one response $s_i = r_i + c_i a$ per slot gives $s^* = \sum_i s_i$ and the forgery (R^*, s^*) :

$$g^{s^*} = \left(\prod_i R_i \right) A^{\sum_i c_i} = R^* A^{c^*}.$$

Wagner's generalized-birthday search [Wag02] finds such additive relations; Sect. 4.7 gives the disjoint message lists and search conditions. This is a concrete-security concern, not by itself a polynomial-time PP-EUF break, and motivates our message-dependent two-nonce aggregation.

Generalized-Birthday Attack on Single Public Nonces. Publishing one nonce per slot creates linear relations across responses and admits a generalized-birthday attack. For public nonces $R_i = g^{r_i}$, fix $R^* = \prod_{i=1}^{\ell} R_i$ and choose pairwise disjoint sets of candidate messages $\mathcal{M}_0, \dots, \mathcal{M}_{\ell}$. Before requesting any response, populate the lists

$$\begin{aligned} L_0 &= \{-H_{\text{sig}}(R^*, A, m) : m \in \mathcal{M}_0\}, \\ L_i &= \{H_{\text{sig}}(R_i, A, m) : m \in \mathcal{M}_i\}, \quad i \in [\ell]. \end{aligned}$$

A solution to this modular $(\ell + 1)$ -sum problem selects messages with $c^* = \sum_i c_i \pmod{q}$. Request the corresponding response $s_i = r_i + c_i a$ once for each slot and output (R^*, s^*) with $s^* = \sum_i s_i$. Then

$$g^{s^*} = \left(\prod_i R_i \right) A^{\sum_i c_i} = R^* A^{c^*}.$$

The target message lies in $\mathcal{M}_0$, so it is fresh. For $\ell + 1$ a power of two, Wagner's generalized-birthday technique [Wag02] applies to this list problem; a larger bundle can supply a subset of that size. The attack demonstrates the concrete-security risk of fixed public nonces; it does not by itself establish a polynomial-time violation of PP-EUF.

Decomposition and Statement Derivation. For the relation $\mathcal{R}_{\text{DL}}$ and a well-formed slot $\rho = (R_1, R_2)$, use the public computations

$$\begin{aligned} b &= H_{\text{non}}(A, R_1, R_2, m), & R &= R_1 R_2^b, \\ c &= H_{\text{sig}}(R, A, m), & Y &= R A^c. \end{aligned} \tag{3}$$

- $(\rho, \tau) \leftarrow \text{OffSign}(a)$. Sample independent $r_1, r_2 \leftarrow \mathbb{Z}_q$; return $\rho = (g^{r_1}, g^{r_2})$ and $\tau = (r_1, r_2)$.
- $d \leftarrow \text{Adm}_{\text{Schnorr}}(A, \rho, m)$. Return 1 exactly when A passes native key validation ($A \in G \setminus \{1\}$), ρ is a pair of canonically encoded elements of G , and m is a native message.
- $s \leftarrow \text{OnSign}(a, \tau, m)$. Recover $A = g^a$ and ρ from τ , and reject if admission fails. Compute b, c from Equation (3) and return $s = r_1 + br_2 + ca \pmod{q}$.
- $\sigma \leftarrow \text{Assemble}(A, \rho, m, s)$. Reject if admission fails or s is not a canonical scalar in $\mathbb{Z}_q$. Otherwise recompute R from Equation (3) and return (R, s) .
- $Y \leftarrow \text{Stmt}(A, \rho, m)$. Reject if admission fails; otherwise return Y from Equation (3).

Identity nonce points, effective commitments R , and statements Y are allowed. Each call to OffSign produces an independent one-use slot. Relative to the single-nonce design, each slot adds one group element; nonce derivation adds one hash and one exponentiation.

Lemma 4.16 (Schnorr statement derivation). *The preceding decomposition is perfectly correct and statement-derivable for $\mathcal{R}_{\text{DL}}$, with perfect sound verification collapse.*

Proof. Inadmissible inputs are rejected. Honest membership and correctness follow from

$$g^s = g^{r_1 + br_2 + ca} = R_1 R_2^b A^c = Y.$$

For any admitted key and nonce pair, $g^s = Y = R A^{H_{\text{sig}}(R, A, m)}$ is exactly native verification of the assembled (R, s) , proving perfect collapse even for malicious inputs. $\square$

Assumption. The one-more discrete logarithm (OMDL) assumption says that, given random challenge points in G and access to an oracle returning discrete logarithms of chosen points, a PPT algorithm cannot output the logarithms of all its challenge points using strictly fewer logarithm-oracle queries than challenge points. We need it for $2\ell + 1$ challenges and at most 2ℓ queries, for every polynomial ℓ .

Lemma 4.17 (Two-nonce preprocessing security). *Under OMDL, the two-nonce implementation is PP-EUF in the random-oracle model, with one online response per slot under a single honest signing key.*

Proof. Embedding and simulation. By batch equivalence, fix a polynomial bound ℓ on offline queries and expose all slots initially. The nonce-combination technique is from MuSig2 [NRS21]. Run the preliminary adversary stage and retain its state and oracle assignments, then obtain the OMDL challenges

$$A, R_{1,1}, R_{1,2}, \dots, R_{\ell,1}, R_{\ell,2}.$$

Abort if $A = 1$ or a preliminary hash query contains A in its key field; otherwise publish A and the nonce pairs. Use lazy oracle tables. Whenever a fresh $H_{\text{non}}(A, R_{i,1}, R_{i,2}, m)$ is assigned b , immediately look up or assign

$$H_{\text{sig}}(R_{i,1}R_{i,2}^b, A, m).$$

This eager lookup only samples otherwise unobserved independent oracle values. To answer an online signing query (i, m) , query the logarithm oracle on

$$R_{i,1}R_{i,2}^bA^c,$$

obtaining the exact online response. Each slot is answered once per execution.

Forking the final forgery. Consider a successful fresh-message forgery (R^*, s^*) whose verification input was assigned before the output. Treat assignments to both hash domains, including internal H_{sig} assignments, as one sequence. Report the assignment index for (R^*, A, m^*) . The general forking lemma [BN06] yields, with polynomial loss, two successful executions sharing the prefix before this assignment and having distinct answers c^*, c'^* . Rewind adversary state and slot flags, preserve prefix oracle values, and independently sample subsequent assignments in both domains, using the same adversary coins. The two forgeries have the same R^*, m^* and give

$$a = (s^* - s'^*)(c^* - c'^*)^{-1} \pmod{q}.$$

Freshness excludes response queries on m^* in both successful runs.

Per-slot equation accounting. Cache identical logarithm-query points across both executions. A slot supplies at most two response equations:

$$s = r_{i,1} + br_{i,2} + ca, \quad s' = r_{i,1} + b'r_{i,2} + c'a.$$

Exclude duplicate published nonce pairs and collisions between distinct fresh H_{non} assignments, including across the forks. Then $b = b'$ implies a common-prefix assignment. Its companion signature challenge is also fixed in the prefix by eager lookup, unless it is the forked assignment itself. In that case its message is m^* and neither run requests a response for it. Hence equal b implies the same response equation: two distinct equations have distinct b . This is the accounting property supplied by the weighted second nonce.

OMDL contradiction. Abort on the excluded collisions before making completion queries. Given a , a slot with no response equations costs two nonce-logarithm queries; a slot with one costs a query for $r_{i,2}$, after which $r_{i,1} = s - br_{i,2} - ca$. With two distinct equations, compute

$$r_{i,2} = \frac{s - s' - (c - c')a}{b - b'}, \quad r_{i,1} = s - br_{i,2} - ca$$

without another query. Response plus completion queries are respectively $0 + 2$, $1 + 1$, or $2 + 0$ per slot; cached coincidences only reduce the total. Thus at most 2ℓ queries recover all $2\ell + 1$ challenge logarithms. The excluded events— $A = 1$, a preliminary-key collision, duplicate nonce pairs, fresh-hash collisions, an unqueried verification hash, and a repeated fork challenge—have total probability polynomial in the query bounds divided by q ; the forking loss is polynomial in the number of assignments. Nonnegligible PP-EUF success therefore contradicts OMDL. $\square$

Remark 4.18 (One-time slots and concrete variants). Reusing a two-nonce slot can give three independent linear equations in $r_{i,1}, r_{i,2}, a$ and reveal the key. The reduction is asymptotic and does not give a tight concrete bound. The construction uses full group encodings and scalar responses; compatibility with x-only encodings, parity conventions, or deterministic nonce generation requires separate analysis.

Theorem 4.19 (Schnorr swap). *Under OMDL in the stated ROM and common group, the two-nonce decomposition and scalar Schnorr adaptor give a secure atomic signature swap with preprocessing for every polynomial ℓ . Both outputs satisfy the native Schnorr verifiers.*

Proof. Lemmas 4.16 and 4.17 give Alice’s properties. OMDL implies DL hardness, so the known Schnorr adaptor [AEE⁺21, GSST24] gives Bob’s required properties. Applying Theorem 4.11 then yields Theorem 4.19. $\square$

4.8 Alice: ECDSA for the Discrete Logarithm Relation

Let the ECDSA group be $G = \langle g \rangle$ of prime order q , written multiplicatively for consistency, with native coordinate reduction $f(R) = x(R) \bmod q$. Alice generates $(A, a) \leftarrow \text{KeyGen}_A(\text{pp}_A)$ by sampling $a \leftarrow \mathbb{Z}_q^*$ and setting $A = g^a$. Let $h = H_E(m) \in \mathbb{Z}_q$ denote the native message-to-scalar rule. The native verifier on (r, s) requires $r, s \in \mathbb{Z}_q^*$ and checks that $g^{h/s} A^{r/s}$ is a nonidentity point whose reduced coordinate is r .

Fixed-base Statement Derivation. For a fixed nonce $R = g^k$, an honest ECDSA response satisfies

$$s = k^{-1}(h + ra), \quad R^s = g^h A^r, \quad r = f(R).$$

The equation $R^s = g^h A^r$ gives a discrete-logarithm statement with base R , whereas Bob’s scalar Schnorr adaptor requires the fixed base g . The message-independent points $U = g^{1/k}$ and $V = A^{1/k}$ give

$$g^s = U^h V^r.$$

This supplies the fixed-base relation $\mathcal{R}_{\text{DL}}$. The inverse-nonce data require consistency validation and a separate preprocessing-unforgeability assumption.

Slot data and consistency proofs. In each slot, sample $k \leftarrow \mathbb{Z}_q^*$, set $R = g^k$, and resample if $r = f(R) = 0$. Publish

$$\rho = (R, U, V, \pi_k, \pi_a), \quad U = g^{1/k}, \quad V = A^{1/k}.$$

Retain $\tau = (k, \rho)$. The proofs certify respectively

$$\exists k : R = g^k \wedge g = U^k, \quad \exists a : A = g^a \wedge V = U^a.$$

Use Fiat–Shamir Chaum–Pedersen proofs [CP93] with independent coins and a random oracle H_{cp} to $\mathbb{Z}_q$, domain-separated by proof type and from all other hashes, binding the full tuple and commitments through an unambiguous canonical encoding. To prove $X = B^x$ and $Z = D^x$, sample $v \leftarrow \mathbb{Z}_q$ and output $\pi = (T_1, T_2, z)$, where

$$T_1 = B^v, \quad T_2 = D^v, \quad d = H_{\text{cp}}(\text{tag}, A, R, U, V, B, D, X, Z, T_1, T_2), \quad z = v + dx.$$

Scalar arithmetic is modulo q . The verifier checks canonical subgroup encodings of T_1, T_2 and a canonical scalar $z \in \mathbb{Z}_q$, recomputes d , and checks $B^z = T_1 X^d$ and $D^z = T_2 Z^d$. The first proof uses the bases (g, U) and images (R, g) ; the second uses the same bases and images (A, V) . All points must have canonical subgroup encodings; require $A, R, U, V \neq 1$, $r \neq 0$, and verify both proofs.

Lemma 4.20 (Consistency-proof soundness). *The two proofs have perfect completeness. For an adversary making at most Q_{cp} queries to H_{cp} before outputting a tuple and its two proofs, the probability that validation accepts a false consistency claim is at most $\epsilon_{\text{cp}} = (Q_{\text{cp}} + 2)/q$.*

Proof. Substitution proves completeness. Special soundness implies that, for a false claim and fixed commitments, at most one challenge can accept. Each fresh hash query fixes these before sampling a uniform challenge; a union bound over Q_{cp} adversarial queries and at most two final verification queries gives $(Q_{\text{cp}} + 2)/q$. $\square$

Decomposition and statement derivation. For $\rho = (R, U, V, \pi_k, \pi_a)$ and $h = H_E(m)$, write

$$r = f(R), \quad Y = U^h V^r, \quad s = (h + ra)/k \pmod{q}, \quad (4)$$

where only the response s requires the private nonce and key.

- $(\rho, \tau) \leftarrow \text{OffSign}(a)$. Generate R, U, V, π_k, π_a as above with $A = g^a$ and $f(R) \neq 0$; return $\rho = (R, U, V, \pi_k, \pi_a)$ and $\tau = (k, \rho)$.
- $d \leftarrow \text{Adm}_{\text{ECDSA}}(A, \rho, m)$. Check native key and message validity, the canonical nonidentity subgroup encodings of R, U, V , $r = f(R) \neq 0$, and both consistency proofs. Return 1 exactly when these checks pass and $U^{H_E(m)} V^r \neq 1$.
- $s \leftarrow \text{OnSign}(a, \tau, m)$. Recover $A = g^a$ and the stored ρ ; reject if admission fails. Return $s = (H_E(m) + ra)/k \pmod{q}$.
- $\sigma \leftarrow \text{Assemble}(A, \rho, m, s)$. Reject if admission fails or s is not a canonical nonzero scalar. Otherwise return the native ECDSA signature $(f(R), s)$.
- $Y \leftarrow \text{Stmt}(A, \rho, m)$. Reject inadmissible inputs. Otherwise compute and return $Y = U^{H_E(m)} V^{f(R)}$.

Admission must exclude $Y = 1$: Alice can choose k, m with $r = f(g^k) \neq 0$, $h \neq 0$, and $a = -h/r$, giving true consistency claims but witness $s = 0$, which yields no native signature. Correctness holds on admitted inputs; otherwise another unused slot may be tried.

Lemma 4.21 (ECDSA derivation and assembly). *The ECDSA decomposition is perfectly correct on admitted inputs and statement-derivable for $\mathcal{R}_{\text{DL}}$. It has sound verification collapse in the ROM: for every PPT $\mathcal{D}$,*

$$\text{Adv}_{\Sigma_{\text{ECDSA}}}^{\text{collapse}}(\mathcal{D}) \leq (Q_{\text{cp}} + 2)/q,$$

where Q_{cp} bounds its consistency-proof hash queries.

Proof. True consistency claims give $R = g^k$, $U = g^{1/k}$, $A = g^a$, and $V = U^a$ for nonzero k, a . Admission ensures $s \neq 0$, so

$$g^s = U^h V^r \implies ks = h + ra \implies g^{h/s} A^{r/s} = g^k = R.$$

Since $f(R) = r \neq 0$, assembly verifies. Honest responses satisfy the relation, and inadmissible inputs are rejected, proving correctness and statement derivation. Every collapse violation thus supplies a false accepted consistency proof; Lemma 4.20 gives the bound. $\square$

Definition 4.22 (ECDSA nonce-preprocessing assumption). For every polynomial ℓ , given an honest ECDSA public key and ℓ independent public slots $(R_i, U_i, V_i, \pi_{k,i}, \pi_{a,i})$ distributed exactly as above, no PPT adversary with one admissible response query $s_i = (H_E(m) + r_i a)/k_i$ per slot can output a native ECDSA signature on a message not queried to the response oracle. The assumption includes the full proof transcripts, the public inverse-nonce points, and adaptive message and random-oracle queries, including preprocessing on pp and the public oracles before the key is revealed. It is precisely PP-EUF for this implementation.

Remark 4.23 (Preprocessing security and ECDSA variants). Def. 4.22 is an additional assumption, not a consequence of ordinary ECDSA EUF-CMA or consistency-proof soundness. Slot reuse on distinct hash values reveals $(s - s')/(h - h') = 1/k$ and then $a = (ks - h)/r$. Raw ECDSA also fails PP-sEUF: $(r, q - s)$ is a different valid signature after one response. Low- s assembly may return $(r, \min(s, q - s))$, but the response interface and PP-EUF assumption still expose the raw s .

Theorem 4.24 (Conditional ECDSA swap). *Assume Def. 4.22 in the joint model, with independent random-oracle domains for consistency proofs and Bob's challenges. The ECDSA decomposition and scalar Schnorr adaptor then give a secure atomic signature swap with preprocessing for every polynomial ℓ , with native ECDSA for Alice and native Schnorr for Bob.*

Proof. Lemma 4.21 gives statement derivation and sound collapse; Def. 4.22 gives PP-EUF by batch equivalence. That assumption also implies DL hardness: recovering a from an honest $A = g^a$ permits a fresh forgery without response queries. The known Schnorr adaptor [AEE⁺21, GSST24] supplies Bob's properties, and Theorem 4.11 applies. $\square$

4.9 BLS Alice and Pairing-Schnorr Bob for the Pairing Relation

BLS supplies Alice's empty-preprocessing decomposition; Bob uses the pairing-Schnorr adaptor of Section 4.5. Alice's native BLS signature is the witness, and pairing verification determines the statement. Let $e : G_1 \times G_2 \rightarrow G_T$ be a nondegenerate, efficiently computable type-3 pairing on prime-order groups of order q , with generators g_1, g_2 in pp . Fix $F_{\text{pair}}(W) = e(W, g_2)$. Nondegeneracy and prime order make this a bijective homomorphism from G_1 to G_T . Its canonical relation is

$$\mathcal{R}_{\text{pair}} = \mathcal{R}_{F_{\text{pair}}} = \{(Y, W) \in G_T \times G_1 : e(W, g_2) = Y\}.$$

All public keys, signatures, statements and commitments are subgroup-checked and canonically encoded. Hash H_A maps messages to $G_1 \setminus \{1\}$. The hash for Bob's challenges has a separate domain and outputs $\mathbb{Z}_q$.

Alice's Statement-Derivable BLS Signatures. Sample $a \leftarrow \mathbb{Z}_q^*$ and set $A = g_2^a$ to generate $(A, a) \leftarrow \text{KeyGen}_A(\text{pp}_A)$; her native signature is $W = H_A(m)^a$. Verification is $e(W, g_2) = e(H_A(m), A)$, with nonidentity key and signature checks. Let $\text{Adm}_{\text{BLS}}(A, \rho, m)$ accept exactly when $A \in G_2 \setminus \{1\}$ has a canonical encoding, $\rho = \varepsilon$, and m is a native message. The algorithms are

$$\begin{aligned} (\rho, \tau) &\leftarrow \text{OffSign}(a), & (\rho, \tau) &= (\varepsilon, \varepsilon), \\ d &\leftarrow \text{Adm}_{\text{BLS}}(A, \rho, m), & d &\in \{0, 1\}, \\ W &\leftarrow \text{OnSign}(a, \varepsilon, m), & W &= H_A(m)^a, \\ \sigma &\leftarrow \text{Assemble}(A, \varepsilon, m, W), & \sigma &= W, \\ Y &\leftarrow \text{Stmt}(A, \varepsilon, m), & Y &= e(H_A(m), A). \end{aligned}$$

The right-hand identities specify the outputs on accepting inputs. Online signing, assembly, and statement derivation return $\perp$ on inadmissible inputs; online signing uses $A = g_2^a$ and requires $\tau = \varepsilon$, and assembly requires $W \in G_1 \setminus \{1\}$ with canonical encoding. The advertisement records the bundle capacity, and the state tracks slot consumption.

Lemma 4.25 (BLS statement derivation and preprocessing security). *The BLS decomposition is perfectly correct and statement-derivable for $\mathcal{R}_{\text{pair}}$, with perfect sound verification collapse. For every PPT PP-EUF adversary $\mathcal{A}$ there is a native BLS EUF-CMA adversary $\mathcal{B}$ in the same common model such that*

$$\text{Adv}_{\Sigma_{\text{BLS}}}^{\text{ppeuf}}(\mathcal{A}) = \text{Adv}_{\Sigma_{\text{BLS}}, \mathcal{B}}^{\text{uf}}(\lambda).$$

The reduction makes one signing query per legal online query and none for offline queries.

Proof. Bilinearity gives correctness and membership; $Y \neq 1$ makes every admitted witness a native valid nonidentity signature, proving perfect collapse. For PP-EUF, preserve the preliminary stage and challenge key, create empty slots locally, and forward legal online queries to the native signing oracle. Views and queried message sets coincide. $\square$

Lemma 4.26 (BLS security implies pairing-preimage hardness). *Native BLS EUF-CMA security implies hardness of inverting $F(W) = e(W, g_2)$ on a uniformly random nonidentity image, in the same common-parameter and public-oracle model.*

Proof. Given an honest BLS key A , choose a fresh m and set $Y = e(H_A(m), A)$. For uniform $U \leftarrow G_1$, mask this image as $P = YF(U)$ and abort if $P = 1$, an event of probability $1/q$. Conditioned on no abort, P is uniform nonidentity and independent of the prior public-oracle view. An inverter returning K with $F(K) = P$ therefore gives the forgery $W = K/U$, since

$$e(W, g_2) = F(K)/F(U) = Y = e(H_A(m), A).$$

Here $Y \neq 1$ ensures native nonidentity validity. No signing query is needed, so nonnegligible inversion success contradicts BLS EUF-CMA. $\square$

The resulting proof chain is

$$\begin{aligned} \text{BLS EUF-CMA} &\implies F_{\text{pair}}\text{-inversion hardness} \\ &\implies \text{pairing-Schnorr full extractability} \implies \text{swap security}. \end{aligned}$$

Theorem 4.27 (BLS–pairing-Schnorr swap). *Assume native BLS EUF-CMA in the joint model, with Bob’s challenge hash an independent random oracle and Alice’s hash in the assumed BLS model. For every polynomial ℓ , the empty BLS decomposition and pairing-Schnorr adaptor give a secure atomic signature swap with preprocessing: the exchanged signatures are native BLS under Alice’s key and native pairing-Schnorr under Bob’s key. The adaptor extends Bob’s scheme, not BLS.*

Proof. Combine Alice’s properties (Lemma 4.25) with Bob’s (Lemmas 4.15 and 4.26) and apply Theorem 4.11. $\square$

4.10 From Signature Swaps to Ledger Atomic Swaps

Ledger deployment must bind the recipient, amount, chain/domain, and transaction or session identifier into native messages; persist slot consumption and Bob-message reservations; and ensure Bob observes σ_B in time to use σ_A , with valid transactions, appropriate confirmations, and compatible refund deadlines. It must also prevent Alice from publishing σ_B and authorizing a conflicting spend before Bob settles, for example through joint control or locked funds. Our theorem establishes signature-level atomicity; ledger-level atomicity and liveness require these additional mechanisms.

Acknowledgments

Vassilis Zikas was supported in part by JPMorgan Chase, Halcyon Futures via the AI Security Institute, the Stellar Development Foundation, and Sunday Group, Inc. Riccardo Zanotto is supported by the European Research Council (ERC) grant (LACONIC, 101041207). Funded by the European Union.

Use of AI tools

Generative AI assistants were used to improve the language of this manuscript (grammar, phrasing, and typographical corrections) and to adjust the formatting of figures and tables. All definitions, theorems, proofs, and technical content are the authors’ own, and the authors take full responsibility for the correctness of all material in this work.

References

- [ABC⁺26] Masayuki Abe, Dung Bui, Kelong Cong, Miyako Ohkubo, Zehua Shang, Akira Takahashi, and Mehdi Tibouchi. Practical adaptor signatures for NP from online/offline NIZK. In *ASIACRYPT*, 2026.
- [AEE⁺21] Lukas Aumayr, Oguzhan Ersoy, Andreas Erwig, Sebastian Faust, Kristina Hostáková, Matteo Maffei, Pedro Moreno-Sanchez, and Siavash Riahi. Generalized channels from limited blockchain scripts and adaptor signatures. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part II*, volume 13091 of *LNCS*, pages 635–664. Springer, Cham, December 2021.
- [BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Berlin, Heidelberg, December 2001.

- [BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM Press, October / November 2006.
- [CDG⁺26] Ryann Cartor, Nathan Daly, Giulia Gaggero, Jason T LeGrow, Andrea Sanguineti, and Silvia Sconza. Post-quantum adaptor signatures with strong security from cryptographic group actions. In *International Conference on Post-Quantum Cryptography*, pages 370–405. Springer, 2026.
- [CLTZ25] Michele Ciampi, Xiangyu Liu, Ioannis Tzannetos, and Vassilis Zikas. Universal adaptor signatures from blackbox multi-party computation. In Arpita Patra, editor, *CT-RSA 2025*, volume 15598 of *LNCS*, pages 375–398. Springer, Cham, April / May 2025.
- [CP93] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *CRYPTO’92*, volume 740 of *LNCS*, pages 89–105. Springer, Berlin, Heidelberg, August 1993.
- [Dis24] Discreet Log Contracts. DLC specification, 2024. <https://github.com/discreetlogcontracts/dlcspecs>.
- [DMZ26] Nico Döttling, Manar Mohamed, and Riccardo Zanotto. GATOR: Group action AdapTOR signatures via MPC-in-the-head. Cryptology ePrint Archive, Paper 2026/1067, 2026.
- [DOY22] Wei Dai, Tatsuaki Okamoto, and Go Yamamoto. Stronger security and generic constructions for adaptor signatures. In Takanori Isobe and Santanu Sarkar, editors, *INDOCRYPT 2022*, volume 13774 of *LNCS*, pages 52–77. Springer, Cham, December 2022.
- [Dry17] Thaddeus Dryja. Discreet log contracts, 2017. MIT Digital Currency Initiative. <https://adiabat.github.io/dlc.pdf>.
- [DTS⁺26] Nico Döttling, Sri Aravinda Krishnan Thyagarajan, Pratik Soni, Jay Taylor, Hendrik Waldner, and Riccardo Zanotto. A general approach to adaptor signatures. Cryptology ePrint Archive, Paper 2026/1889, 2026.
- [EEE20] Muhammed F. Esgin, Oguzhan Ersoy, and Zekeriya Erkin. Post-quantum adaptor signatures and payment channel networks. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *ESORICS 2020, Part II*, volume 12309 of *LNCS*, pages 378–397. Springer, Cham, September 2020.
- [EFH⁺21] Andreas Erwig, Sebastian Faust, Kristina Hostáková, Monosij Maitra, and Siavash Riahi. Two-party adaptor signatures from identification schemes. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 451–480. Springer, Cham, May 2021.
- [GH26] Emanuele Giunta and Kristina Hostáková. Almost scriptless adaptor signatures from any signature scheme. Cryptology ePrint Archive, Paper 2026/1346, 2026.
- [GMM⁺22] Noemi Glaeser, Matteo Maffei, Giulio Malavolta, Pedro Moreno-Sanchez, Erkan Tairi, and Sri Aravinda Krishnan Thyagarajan. Foundations of coin mixing services. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 1259–1273. ACM Press, November 2022.
- [GMTMSS26] Javier Gomez-Martinez, Erkan Tairi, Pedro Moreno-Sanchez, and Clara Schneidewind. Adaptor signatures meet BLS: Enabling efficient blockchain applications with unique adaptor signatures. Cryptology ePrint Archive, Paper 2026/1394, 2026.
- [GSST24] Paul Gerhart, Dominique Schröder, Pratik Soni, and Sri Aravinda Krishnan Thyagarajan. Foundations of adaptor signatures. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 161–189. Springer, Cham, May 2024.

- [GV13] David Galindo and Srinivas Vivek. A leakage-resilient pairing-based variant of the Schnorr signature scheme. In Martijn Stam, editor, *14th IMA International Conference on Cryptography and Coding*, volume 8308 of *LNCS*, pages 173–192. Springer, Berlin, Heidelberg, December 2013.
- [HLTW24] Lucjan Hanzlik, Julian Loss, Sri Aravinda Krishnan Thyagarajan, and Benedikt Wagner. Sweep-UC: Swapping coins privately. In *2024 IEEE Symposium on Security and Privacy*, pages 3822–3839. IEEE Computer Society Press, May 2024.
- [HMR26] Pavel Hubáček, Kristýna Mašková, and Berenika Richterová. Practical adaptor signatures: Security and applications. *Cryptology ePrint Archive*, Paper 2026/1482, 2026.
- [JMV01] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ECDSA). *International Journal of Information Security*, 1(1):36–63, 2001.
- [KH22] Jean Belo Klamti and M Anwar Hasan. Post-quantum two-party adaptor signature based on coding theory. *Cryptography*, 6(1):6, 2022.
- [LTZ24] Xiangyu Liu, Ioannis Tzannetos, and Vassilis Zikas. Adaptor signatures: New security definition and a generic construction for NP relations. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part II*, volume 15485 of *LNCS*, pages 168–193. Springer, Singapore, December 2024.
- [MMS⁺19] Giulio Malavolta, Pedro Moreno-Sanchez, Clara Schneidewind, Aniket Kate, and Matteo Maffei. Anonymous multi-hop locks for blockchain scalability and interoperability. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019.
- [Nat23] National Institute of Standards and Technology. FIPS 186-5: Digital signature standard (DSS), 2023.
- [NRS21] Jonas Nick, Tim Ruffing, and Yannick Seurin. MuSig2: Simple two-round Schnorr multi-signatures. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 189–221, Virtual Event, August 2021. Springer, Cham.
- [Oka93] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *CRYPTO’92*, volume 740 of *LNCS*, pages 31–53. Springer, Berlin, Heidelberg, August 1993.
- [Poe17] Andrew Poelstra. Scriptless scripts. Presentation slides, March 2017. Accessed on May 2026.
- [Poe18] Andrew Poelstra. Mumblewimble and scriptless scripts. Presentation slides, January 2018. Accessed on May 2026.
- [Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO’89*, volume 435 of *LNCS*, pages 239–252. Springer, New York, August 1990.
- [TMM21] Erkan Tairi, Pedro Moreno-Sanchez, and Matteo Maffei. Post-quantum adaptor signature for privacy-preserving off-chain payments. In Nikita Borisov and Claudia Díaz, editors, *FC 2021, Part II*, volume 12675 of *LNCS*, pages 131–150. Springer, Berlin, Heidelberg, March 2021.
- [TMM22] Sri Aravinda Krishnan Thyagarajan, Giulio Malavolta, and Pedro Moreno-Sanchez. Universal atomic swaps: Secure exchange of coins across all blockchains. In *2022 IEEE Symposium on Security and Privacy*, pages 1299–1316. IEEE Computer Society Press, May 2022.
- [TZY22] Binbin Tu, Min Zhang, and Chen Yu. Efficient ECDSA-based adaptor signature for batched atomic swaps. In Willy Susilo, Xiaofeng Chen, Fuchun Guo, Yudi Zhang, and Rolly Intan, editors, *ISC 2022*, volume 13640 of *LNCS*, pages 175–193. Springer, Cham, December 2022.

- [Wag02] David Wagner. A generalized birthday problem. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 288–303. Springer, Berlin, Heidelberg, August 2002.
- [WHL24] Weihao Wang, Shuai Han, and Shengli Liu. Anamorphic authenticated key exchange: Double key distribution under surveillance. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part V*, volume 15488 of *LNCS*, pages 168–200. Springer, Singapore, December 2024.

Contents

1	Introduction	1
1.1	Our Contributions	3
1.2	Related Work	5
2	Preliminaries	6
2.1	Signatures	6
2.2	Schnorr, ECDSA, Okamoto, and BLS Signatures	7
2.3	Hardness Assumptions	9
2.4	NP Languages	9
2.5	Adaptor Signatures	9
3	Lower Bound for Non-Interactive Atomic Swaps	12
3.1	Definitions and Useful Lemmas	12
3.2	Lower Bound	13
3.3	Corollaries and Implications	15
3.4	Which Signature Pairs Are Feasible	16
4	Atomic Signature Swaps with Preprocessing	17
4.1	Syntax and Security of Atomic Signature Swaps	17
4.2	Decomposable and Statement-Derivable Signatures	20
4.3	Generic Construction and Security	22
4.3.1	Correctness	24
4.3.2	Unforgeability	24
4.3.3	Atomicity	25
4.4	Instantiations Overview	27
4.5	Bob's Adaptors for Canonical Relations	27
4.6	Pairing-Schnorr Signatures and Adaptors	28
4.7	Schnorr Alice for the Discrete-Logarithm Relation	29
4.8	Alice: ECDSA for the Discrete Logarithm Relation	32
4.9	BLS Alice and Pairing-Schnorr Bob for the Pairing Relation	34
4.10	From Signature Swaps to Ledger Atomic Swaps	36