

Switch Commitments by Coincidence: Confidential Transactions Were Quantum-Upgradable All Along

Yevhen Hrubiiian , Mykyta Redko , and Tim Ruffing 

Blockstream Research
{yhrubiian,mredko}@blockstream.com, me@real-or-random.org

September 23, 2026

Abstract. Upgrading cryptocurrencies to post-quantum security is inherently difficult, because their on-chain state associates users’ holdings with public keys: users who fail to transfer their funds to (yet to be added) post-quantum signature schemes remain exposed to theft. In networks with Confidential Transactions (CT) such as Liquid, Monero, and Grin, which hide monetary amounts inside homomorphic Pedersen commitments, the situation is worse still: an adversary who computes a single discrete logarithm can inflate the currency as a whole, rather than merely steal individual funds. The contentious remedy is to expire all funds not transferred by a deadline, destroying the funds of everyone who fails to act. Switch commitments [RM17] were proposed to avoid exactly this expiration, but only Grin has deployed them deliberately—and the security theorem later stated for its scheme does not hold, as we show.

In this paper, we show that Liquid and Monero already deploy a switch commitment scheme by coincidence: the ciphertext a sender stores on-chain to convey the hidden amount to the recipient can be re-interpreted, together with the Pedersen commitment, as a switch commitment. We conceptualize this construction as *coincidental switch commitments* and prove it secure in the quantum random oracle model. Additionally, we save the construction that Grin deploys: it is computationally switch binding, with practical bounds in the algebraic group model. The real-world implication of our results is that these networks can be upgraded to post-quantum CT schemes without expiring any user funds.

Table of Contents

1	Introduction	2
2	Background on Confidential Transactions	4
3	Preliminaries	6
3.1	One-Way to Hiding (O2H)	7
3.2	Commitment Schemes	8
4	Switch Commitments	9
5	Coincidental Switch Commitments	11
5.1	Decommitment Encryption	11
5.2	Coincidental Commitments	13
5.3	Application to Liquid	17
5.4	Application to Monero and Zano	24
6	Opt-In Switch Commitments, Revisited	26
6.1	Simplified Tweaking Switch Commitments	32
	References	34
A	Proof of Theorem 3	37
B	Extending the Coincidental Scheme to Assets	39

1 Introduction

The security of most modern cryptocurrency networks still relies on classical hardness assumptions, such as the discrete logarithm assumption underlying the unforgeability of ECDSA and Schnorr signatures. In the light of recent significant advances in the quantum cryptanalysis of discrete logarithms on the `secp256k1` elliptic curve as used in Bitcoin and many other networks [Bab+26; CFS26], pressure to upgrade to post-quantum security is intensifying.

However, a post-quantum upgrade is a particularly difficult endeavor in the presence of a blockchain keeping cryptographic state (e.g., public keys) that cannot be upgraded without active involvement of users. While, for example, a network upgrade can introduce support for post-quantum signatures, this will only ensure the security of future transactions but not retroactively fix transactions performed in the past: all user funds received before the upgrade will remain under legacy public keys and thus vulnerable to quantum attacks.

As soon as a post-quantum scheme is available, it is therefore the responsibility of users to transfer their funds to fresh post-quantum public keys. As a result, what such an upgrade can achieve is to keep the damage contained. Individual users may have their funds stolen if they fail to react in time, but if a sufficient number of users perform the transfer to post-quantum public keys, the security of the cryptocurrency as a whole will remain largely unaffected.

However, the situation will be much worse in a cryptocurrency network with Confidential Transactions (CT) [Bac13; Max15]. CT is a privacy-enhancing extension to cryptocurrencies which hides the monetary amounts in transactions by replacing plain amounts with commitments to the amounts. Existing deployments of CT rely on legacy Pedersen commitments, which are only *computationally* binding under the discrete logarithm assumption. Crucially, as opposed to the case of signatures, simply introducing a post-quantum CT scheme [Tor+18; Tor+19; Esg+19; LNS21; ESZ22; Gao+25] is not sufficient to contain the damage by a quantum attack. In fact, an adversary who breaks a *single* discrete logarithm can open a commitment corresponding to some legitimately owned funds to any chosen value, effectively claiming any chosen amount of money and thereby inflating the currency—while the perfect hiding property of the commitments ensures that the attack even goes entirely unnoticed. In other words, one broken discrete logarithm does not merely threaten the funds of users who fail to act; it destroys the integrity of the entire currency.

One drastic way out is to perform an upgrade that introduces a hard expiration deadline by which funds stored in legacy commitments become unspendable. However, such an upgrade would forcefully destroy funds held by honest users who fail to transfer their funds by the deadline—no matter if quantum attacks turn out to be a real threat by that time. It is therefore not clear at all whether an upgrade introducing an expiration deadline would find sufficient agreement among stakeholders. Worse, agreeing on a concrete deadline is difficult due to the inherently vague predictions on future advances in quantum computing. (In the context of Bitcoin, similar proposals that would destroy funds protected by legacy signature schemes have proven to be highly contentious [Lop25; Cru25].)

Switch commitments, originally proposed by Ruffing and Malavolta [RM17], provide a cryptographic solution to the problem of expiration. A network that uses switch commitments from the outset can be upgraded to a post-quantum CT scheme without requiring all users to actively transfer their funds, and thus without the need to expire funds forcefully.

A switch commitment scheme defines two verification modes, namely classically secure *partial verification* and post-quantum secure *full verification*. Initially, the network performs only partial verification, and since the partial verification algorithm (in all proposed schemes) matches the verification of Pedersen commitments, switch commitments are fully compatible with the original Pedersen-based CT scheme. The network never sees an opening in the clear: every output carries a *range proof*, a zero-knowledge proof of knowledge of a decommitment instead (which also proves that the amount is in the valid range), and that is what the network checks. As soon as the arrival of a cryptographically relevant quantum computer appears to be a realistic and immediate threat, the network can *switch* to the post-quantum secure full verification algorithm to retain the binding guarantee, thereby safeguarding the integrity of the currency. This switch to full verification is simply a soft-forking (i.e., backward-compatible) network upgrade that can be deployed without expiring funds, and it will also disallow the creation of further (now legacy) switch commitments.

What avoids the need to expire funds is that all the data necessary for performing full verification is published on chain in the form of switch commitments, even initially when only partial verification is used. In other words, the on-chain state always binds to the originally committed amounts, and partial (Pedersen-based) verification enforces this binding only against classical adversaries whereas full verification enforces it even against quantum adversaries. The security notion spanning the switch, *switch binding*, accordingly considers an adversary in two stages: a classical stage that creates a commitment and opens it before the switch, and a quantum stage that tries to open the same commitment to a different amount afterwards. Only the classical stage is subject to computational hardness assumptions; the quantum stage is bounded merely in the number of its random-oracle queries.

Assuming that a network upgrade has added support for a post-quantum CT scheme including a post-quantum commitment scheme, the owner of funds stored in a switch commitment can migrate this commitment as follows: The owner creates a post-quantum commitment to the same amount, and publishes a proof that the switch commitment and the post-quantum commitment indeed open to the same amount. Ideally, this proof is a post-quantum secure non-interactive zero-knowledge proof (*privacy-preserving* migration). Alternatively, the upgraded rules of the network may additionally allow the owner to simply publish openings to both commitments as a proof (*transparent* migration), which avoids the possibly large and expensive zero-knowledge proofs at the cost of forgoing privacy. Whether the proof statement uses partial or full verification for the switch commitment depends on whether the migration happens before or after the switch (and as a result, migrating after the switch may be significantly more expensive in terms of computation and proof size, because partial verification merely recomputes a Pedersen commitment whereas full verification does more, e.g., an additional computation of a hash function).

Unfortunately, deliberate adoption of switch commitments has been rare: to the best of our knowledge, Grin, an implementation of the Mimblewimble protocol [Jed16; FOS19], is the only deployed CT network that deliberately implemented switch commitments [Gri19a]. And while other existing networks could still introduce them in principle, doing so now would be much less effective because only commitments created from then on would enjoy the everlasting protection that switch commitments offer. In contrast, commitments created before the switch commitments were introduced would still put the entire currency at inflation risk, and the funds they hold would still need to be expired.

Contributions. As a first contribution, we show that the major CT networks Liquid and Monero have deployed switch commitments “by coincidence”, and thus have been *quantum-upgradable* all along (Sect. 5).

Behind this perhaps surprising insight is the fact that in every existing deployment of CT, the decommitment information of the Pedersen commitment (i.e., the amount inside the commitment and the randomness) is stored on the blockchain itself in a confidentiality-preserving way, so that the recipient can recover it. Depending on the concrete network, this is either done by attaching an explicit separate ciphertext to the on-chain transaction data, or by piggybacking the ciphertext data cleverly into other parts of the transaction data such as range proofs (the zero-knowledge proofs necessary for proving that the confidential amounts in a transaction do not cause arithmetic overflow). In Liquid, Monero, and Zano, the recipient obtains the secret to decrypt the decommitment information by completing an elliptic-curve Diffie–Hellman (ECDH) key exchange with the sender. In Grin, whose transactions are built interactively, the recipient creates its own commitment and derives the secret from its wallet keys, so that it can recover the decommitment information from the chain when restoring the wallet.

Since an honest recipient learns this secret, full verification can require it in addition to the decommitment information: full verification derives the decryption key from the secret, decrypts the ciphertext, and checks that the result is the claimed amount and randomness. This check does not rely on the hardness of discrete logarithms: the ciphertext is fixed on chain, and a quantum adversary who wants to open the commitment to a different amount can do no better than search for a secret whose key decrypts the ciphertext accordingly. As a result, the Pedersen commitment together with the ciphertext can be viewed as a switch commitment to the amount that is binding even against a quantum adversary.

In more detail, we make the following contributions:

- To capture existing methods of conveying the decommitment information to the recipient, we formally introduce the concept of *decommitment encryption* (Definition 8). This primitive is a variant of *committing encryption* [GLR17; BH22; CR22]: it comes with a robustness notion, *random-key robustness*: a ciphertext must not decrypt to a valid decommitment of an accompanying commitment under a uniformly random key (Sect. 5.1).
- We give a generic construction of *coincidental switch commitments* CoCom from a commitment scheme and a decommitment encryption scheme (Construction 3). We prove that it is switch binding (Theorem 1) and also *post-quantum hiding* (Theorem 2) in the quantum random oracle model (QROM).¹
- We then show that our formal framework fits existing CT deployments in popular networks such as Liquid [Poe+19], Monero [NMM16] and Zano [sow24] (Sects. 5.3 and 5.4). For each of these networks, we distill the underlying decommitment encryption scheme and prove that it is random-key robust (Lemma 4 for Liquid and Lemma 5 for Monero and Zano).

As a second contribution, we revisit *opt-in switch commitments*, a deliberate construction of switch commitments initially suggested by Wuille and deployed in Grin [Ruf18; Gri19a]. Their name refers to an emergency fallback: after the switch, a user can opt, for each of its commitments individually, to reveal an ElGamal commitment to the amount, which is perfectly binding and still hides the amount from classical adversaries.

- We give a simple counterexample showing that opt-in switch commitments are not *statistically* switch binding, as claimed [Ruf20].
- We show how to save the construction by proving that it is *computationally* switch binding under the square computational Diffie–Hellman (SqCDH) assumption (for the classical part of the adversary) in the (quantum) random oracle model (Theorem 4). In the algebraic group model (AGM) [FKL18], the loss of a factor of the size of the message space in this reduction can be traded for an *additive* increase in the running time of a reduction to the discrete logarithm assumption (Theorem 5). Both theorems cover OptCom and OptCom’ alike.
- Additionally, we provide a simplified variant of the construction, which we call the *simplified tweaking scheme* SimpCom. At the cost of giving up the emergency fallback, it is more efficient, and its computational switch binding property reduces to the discrete logarithm assumption directly in the QROM (Theorem 6), with a reduction that is tighter than both reductions for opt-in switch commitments: it loses no factor of the size of the message space and does not resort to the AGM. The scheme is post-quantum hiding in the quantum random oracle model (Theorem 3) by a standard “One-Way to Hiding” argument; the proof covers also both instances OptCom and OptCom’ of the opt-in scheme.

Table 1 shows the practical relevance of our results by exemplifying upper bounds on the success probabilities of a concrete attacker whose classical stage runs in time 2^{100} and makes 2^{100} random-oracle queries (about 25 years of the Bitcoin network’s current rate of SHA-256 evaluations [Blo26a]), and whose quantum stage makes 2^{64} quantum random-oracle query rounds (“the approximate number of gates that current classical computing architectures can perform serially in a decade” [Nat16]) where each query round makes 2^{36} queries in parallel, i.e., again 2^{100} queries altogether.

2 Background on Confidential Transactions

UTXO Model. The cryptocurrency networks we consider follow the *unspent transaction output* (UTXO) model introduced by Bitcoin [Nak08]. Funds are held in *outputs*, each of which carries an amount and a spending condition, typically a public key of its owner. A transaction *spends* some existing unspent outputs, which serve as the *inputs* of the transaction, and creates new unspent outputs in their place (usually including a *change* output that returns the remaining funds to the sender).

¹ Post-quantum hiding concerns the switch commitment alone: the deployed networks establish the recipient’s decryption secret via ECDH, which a quantum adversary can break retroactively in a harvest-now-decrypt-later attack, so preserving confidentiality additionally requires replacing ECDH with a post-quantum key encapsulation mechanism, which is orthogonal to our results.

Table 1. Upper bounds on the success probability of an attacker against switch binding and post-quantum (PQ) hiding of the switch commitment in the quantum random oracle model (QROM), as given by the indicated theorems. The attacker’s classical stage runs in time 2^{100} and makes 2^{100} random-oracle queries; its quantum stage makes 2^{100} quantum random-oracle queries in 2^{64} rounds of 2^{36} parallel queries each. The two OptCom rows hold for OptCom’ alike. We assume group orders $q \approx 2^{256}$ (secp256k1; Liquid, Grin) and $q \approx 2^{252}$ (ed25519; Monero, Zano), seeds of the same length as the group order, amounts of at most 64 bits, and that no better-than-generic algorithm for DL or SqCDH exists, i.e., that every algorithm running in time t' solves either of them with probability at most $t'^2/2q$ [cf. Sho97, Theorems 1 and 3]. The entry of the SqCDH row exceeds 1 and is thus vacuous. Each entry is the bound of the indicated theorem at these parameters; the reductions for switch binding run in time about 2^{108} to 2^{111} , depending on the scheme (Sects. 5.3 and 5.4).

Scheme	Deployed in	Switch binding		PQ hiding
		Assumption	Success prob.	Success prob.
CoCom (Sect. 5)	Liquid	DL	2^{-35} (Thm. 1)	2^{-44} (Thm. 2)
CoCom (Sect. 5)	Monero, Zano	DL	2^{-35} (Thm. 1)	2^{-42} (Thm. 2)
OptCom (Sect. 6)	Grin	SqCDH	vacuous (Thm. 4)	2^{-44} (Thm. 3)
OptCom (Sect. 6)	Grin	AGM+DL	2^{-41} (Thm. 5)	2^{-44} (Thm. 3)
SimpCom (Sect. 6) (new)		DL	2^{-39} (Thm. 6)	2^{-44} (Thm. 3)

Spending an input requires the owner’s authorization, e.g., a signature under the owner’s public key. Upgrading the authorization mechanism to post-quantum security is orthogonal to the problem of upgrading CT considered in this paper.

Confidential Amounts. In CT, the explicit amount of every output is replaced by a hiding, homomorphic commitment, as proposed by Back [Bac13] and Maxwell [Max15].² In the initial proposal, and in all deployed networks, this is a Pedersen commitment $C = mH + rG$ to the amount m under randomness r , which is perfectly hiding and whose binding reduces to the discrete logarithm problem (DL). (Even in Grin, which deploys a switch commitment scheme deliberately, switch commitments have exactly the shape of Pedersen commitments, and partial verification matches exactly the verification of Pedersen commitments.)

Balance Equation. A transaction is valid only if it is balanced, that is, if its inputs cover its outputs together with some transaction fee. Because Pedersen commitments are additively homomorphic, a validator can test this on the commitments alone, without the amount hidden inside the commitments. The *balance equation* states that the input commitments minus the output commitments and the fee commitment form a commitment to zero,

$$\sum_i C_{\text{in},i} - (\sum_i C_{\text{out},i} + C_{\text{fee}}) = 0 \cdot H + rG,$$

where the fee is published in the clear, so a validator can compute the commitment C_{fee} to the fee on the fly.

Different networks use different ways of convincing validators that the commitment on the right-hand side opens to 0. In Liquid, the sender chooses the randomness of the output commitments such that it sums to the randomness of the input commitments. In Monero and Zano, the randomness of an output is derived from a secret shared with the recipient and thus not free; there the sender replaces every input commitment in the balance equation by a fresh commitment to the same amount, a *pseudo output commitment*, chooses the randomness of these replacements such that the sums cancel, and proves in zero knowledge that each replacement commits to the same amount as the input it replaces [NMM16; sow24]. In Grin, whose transactions are built interactively, the recipient chooses the randomness of its own output, and the transaction publishes the group element

² Cryptocurrencies that hide amounts inside succinct zero-knowledge proofs instead of homomorphic commitments, e.g., Zcash [Ben+14], are outside the setting considered here.

on the right-hand side (called the *excess*). The sender and the recipient jointly sign under the excess as a public key with respect to G , which proves that they know its discrete logarithm and thus that the excess commits to the amount 0 [Jed16]. In each case, the randomness of the commitments in a transaction is not independent, but the only relation this induces is the balance equation itself, which is public by design. Since nobody knows $\log_G H$, these checks convince the network that no value was created or destroyed, while the individual amounts stay hidden.

Range Proofs. The balance equation on its own is not sufficient. Commitment arithmetic is carried out modulo the group order q , so a malicious sender could pick amounts that wrap around the modulus and behave as negative values, satisfying the balance equation while creating money from nothing. Each output therefore carries a *range proof*, a zero-knowledge proof of knowledge of a decommitment (m, r) of its commitment $C = mH + rG$ such that m lies in a fixed range of admissible amounts, e.g., $[0, 2^{52})$ in Liquid. Consequently, the network never sees a decommitment, and it checks the commitment of an output only through the range proof and the balance equation. Liquid uses Back-Maxwell range proofs [Poe+19] (Sect. 5.3), Grin uses Bulletproofs [Bün+18], and Monero and Zano use Bulletproofs+ [Chu+20].

3 Preliminaries

Notation. In pseudocode, we write $y := x$ for assignment of value x to variable y , and $y \leftarrow \$ S$ for sampling y uniformly at random from a set S . We use $\perp$ to denote the absence of a value, and algorithms return $\perp$ when they abort. For an (oracle) function H , we write $H[x' := y](x)$ for the function reprogrammed at x' , i.e., such that $H[x' := y](x) = y$ if $x' = x$ and $H[x' := y](x) = H(x)$ otherwise.

Groups. Throughout the paper, $\mathbb{G}$ denotes a group of prime order q with generator G , *written additively*, and $\mathbb{Z}_q$ denotes the integers modulo q .

The schemes discussed in this work will need a second generator H , and one of them a third generator J , such that nobody knows a discrete logarithm relation between them. Since our paper is inherently in the (quantum) random oracle model, we fix $H := \text{HashToGroup}(\text{"H"})$ and $J := \text{HashToGroup}(\text{"J"})$, where $\text{HashToGroup}: \{0, 1\}^* \rightarrow \mathbb{G}$ is a hash function into $\mathbb{G}$ modeled as a (quantum) random oracle. This avoids notational clutter from setup algorithms and public parameters, and it matches what deployments do in practice.

The advantage of an adversary $\mathcal{A}$ against the *Discrete Logarithm Problem (DL)* in $\mathbb{G}$ is

$$\text{Adv}_{\mathbb{G}}^{\text{dl}}(\mathcal{A}) := \Pr[\mathcal{A}(\mathbb{G}, G, xG) = x \mid x \leftarrow \$ \mathbb{Z}_q].$$

The advantage of an adversary $\mathcal{A}$ against the *Square Computational Diffie–Hellman Problem (SqCDH)* in $\mathbb{G}$ is

$$\text{Adv}_{\mathbb{G}}^{\text{sqcdh}}(\mathcal{A}) := \Pr[\mathcal{A}(\mathbb{G}, G, xG) = x^2G \mid x \leftarrow \$ \mathbb{Z}_q].$$

(Quantum) Algorithms and Oracles. Throughout, *running time* of algorithms is measured in a standard model of computation in which each oracle query and each group operation takes one unit of time; multiplying a group element by a scalar thus costs $t_{\text{mul}} = O(\log q)$.

We work in the (quantum) random oracle model [Bon+11], i.e., classical (or quantum) adversaries are modeled as oracle algorithms $\mathcal{A}^H$ that have classical (or quantum) oracle access to a random function H . In particular, quantum adversaries can query H “in superposition”.

We say that an oracle algorithm $\mathcal{A}^H$ has *query depth* d and *query width* w if it performs at most d query rounds, each consisting of at most w parallel queries that may not depend on the results of the queries in the same round. The distinction between the query depth and query width is motivated by the fact that better bounds can be derived against quantum attacks that exploit parallelism. Classical adversaries gain nothing from parallelism, because parallel classical queries can be serialized; we therefore let them query at width one, so that their query depth is simply their number of queries.

Zero Knowledge Proofs. We fix notation for the non-interactive proof systems that the range proof of Confidential Assets [Poe+19] instantiates. Let R be an efficiently computable relation consisting of tuples $(\mathbf{x}, \mathbf{w})$, where $\mathbf{x}$ is a statement and $\mathbf{w}$ a witness for it, and let $\text{NIP}_R = (\text{Prove}, \text{Verify})$ be a non-interactive proof system for R , i.e., one satisfying completeness and soundness.

Our security proofs use only *witness indistinguishability* [FS90], a notion considerably weaker than zero knowledge: an adversary holding two valid witnesses for a statement, together with a proof produced from one of them, cannot tell which was used. This weaker property suffices to show that the Back-Maxwell range proof constitutes a decommitment encryption (Lemma 4), where the proof itself forms part of the commitment.

Definition 1 (Witness Indistinguishability). Let $\text{NIP}_R = (\text{Prove}, \text{Verify})$ be a non-interactive proof system for relation R . We say it is perfectly witness-indistinguishable if for all adversaries $\mathcal{A}$, for every statement $\mathbf{x}$, and for every pair of witnesses $\mathbf{w}_0, \mathbf{w}_1$ such that $(\mathbf{x}, \mathbf{w}_0), (\mathbf{x}, \mathbf{w}_1) \in R$ we have that,

$$\Pr[\pi := \text{Prove}(\mathbf{x}, \mathbf{w}_0); \mathcal{A}(\mathbf{x}, \pi) = 1] = \Pr[\pi := \text{Prove}(\mathbf{x}, \mathbf{w}_1); \mathcal{A}(\mathbf{x}, \pi) = 1].$$

3.1 One-Way to Hiding (O2H)

To bound the distinguishing advantage when a quantum random oracle response is reprogrammed, we will rely on a strong One-Way to Hiding (O2H) lemma by Ambainis, Hamburg, and Unruh [AHU18, Theorem 3]; the following simplified variant states only the existence of an extractor $\mathcal{E}$, but does not give its construction.

Lemma 1 (One-Way to Hiding (O2H)). Let $S \subseteq \mathcal{X}$ be a random set, and let z be a random bitstring. Let $\mathbf{G}, \mathbf{H} : \mathcal{X} \rightarrow \mathcal{Y}$ be random functions satisfying $\forall x \notin S : \mathbf{G}(x) = \mathbf{H}(x)$. ($S, \mathbf{G}, \mathbf{H}, z$ may have arbitrary joint distribution). Let $\mathcal{A}$ be a quantum oracle algorithm with query depth d and query width w . Then there is a quantum oracle algorithm $\mathcal{E}$ outputting a set of at most w elements of $\mathcal{X}$ such that

$$\begin{aligned} & \left| \Pr[\mathcal{A}^{\mathbf{H}}(z) = 1] - \Pr[\mathcal{A}^{\mathbf{G}}(z) = 1] \right| \leq 2d\sqrt{\Pr[\mathcal{E}^{\mathbf{G}}(z) \cap S \neq \emptyset]} \\ \text{and} \quad & \left| \sqrt{\Pr[\mathcal{A}^{\mathbf{H}}(z) = 1]} - \sqrt{\Pr[\mathcal{A}^{\mathbf{G}}(z) = 1]} \right| \leq 2d\sqrt{\Pr[\mathcal{E}^{\mathbf{G}}(z) \cap S \neq \emptyset]}. \end{aligned}$$

We will also need the following lemma (a variant of [AHU18, Lemma 2]), which bounds the probability that a quantum oracle algorithm $\mathcal{A}^{\mathbf{H}}$ finds an oracle input x such that $x, \mathbf{H}(x)$ satisfies some boolean predicate θ .

Lemma 2 (Predicate-Based Search). Let $\mathbf{H} : \mathcal{X} \rightarrow \mathcal{Y}$ be a random function whose values $\mathbf{H}(x)$ for $x \in \mathcal{X}$ are mutually independent (e.g., uniformly distributed over $\mathcal{Y}$ as in a random oracle). Let $\theta : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\}$ be a boolean predicate such that $\Pr[\theta(x, \mathbf{H}(x)) = 1] \leq \varepsilon$ for all $x \in \mathcal{X}$. Let z be random and independent of $\mathbf{H}$. Then for every quantum oracle algorithm $\mathcal{A}$ with query depth d and query width w ,

$$\Pr[\theta(x, \mathbf{H}(x)) = 1 \mid x := \mathcal{A}^{\mathbf{H}}(z)] \leq 4(d+1)^2 w \varepsilon.$$

Proof. Let $\hat{\mathcal{A}}^{\mathbf{H}}$ be the quantum oracle algorithm that runs $x := \mathcal{A}(z)$ and returns $\theta(x, \mathbf{H}(x))$. Since evaluating $\theta(x, \mathbf{H}(x))$ requires one additional oracle query, $\hat{\mathcal{A}}$ is a quantum oracle algorithm with query depth $d+1$ and query width w .

Let $S = \{x \in \mathcal{X} : \theta(x, \mathbf{H}(x)) = 1\}$ and define an oracle $\mathbf{G}$ as follows: For every $x \in \mathcal{X} \setminus S$, let $\mathbf{G}(x) := \mathbf{H}(x)$. For every $x \in S$, let $\mathbf{G}(x) := y$, where y is drawn from the distribution that is the same as the conditional distribution of $\mathbf{H}(x)$ given $\theta(x, \mathbf{H}(x)) = 0$. (This is equivalent to rejection sampling y from the distribution of $\mathbf{H}(x)$ until a value is found for which $\theta(x, y) = 0$.) For every $x \in \mathcal{X}$, conditioned on $x \notin S$, the value $\mathbf{G}(x) = \mathbf{H}(x)$ is distributed as $\mathbf{H}(x)$ conditioned on $\theta(x, \mathbf{H}(x)) = 0$; conditioned on $x \in S$, the same holds by construction. So $\mathbf{G}(x)$ is independent of whether $x \in S$, and since the values of $\mathbf{H}$ are mutually independent, $\mathbf{G}$ is independent of S ; so is $(\mathbf{G}, z)$, because z is independent of $\mathbf{H}$.

By [Lemma 1](#) (the variant with square roots on the left-hand side) applied to $\hat{\mathcal{A}}$, there exists a quantum oracle algorithm $\mathcal{E}$ outputting a set of at most w elements of $\mathcal{X}$ such that

$$\left| \sqrt{\Pr[\hat{\mathcal{A}}^H(z) = 1]} - \sqrt{\Pr[\hat{\mathcal{A}}^G(z) = 1]} \right| \leq 2(d+1)\sqrt{\Pr[\mathcal{E}^G(z) \cap S \neq \emptyset]}. \quad (1)$$

By definition of G , we have for every $x \in \mathcal{X}$ that $\theta(x, G(x)) = 0$. Thus

$$\Pr[\hat{\mathcal{A}}^G(z) = 1] = 0. \quad (2)$$

Since (G, z) is independent of S , the algorithm $\mathcal{E}^G(z)$ has no better chance of finding $x \in S$ than outputting w guesses, i.e.,

$$\Pr[\mathcal{E}^G(z) \cap S \neq \emptyset] \leq w\varepsilon. \quad (3)$$

Plugging (2) and (3) into (1) and squaring yields

$$\Pr[\theta(x, H(x)) = 1 \mid x := \mathcal{A}^H(z)] = \Pr[\hat{\mathcal{A}}^H(z) = 1] \leq 4(d+1)^2 w\varepsilon. \quad \square$$

3.2 Commitment Schemes

A non-interactive commitment scheme allows committing to a hidden message m with the ability to reveal it later.

Definition 2 (Commitment Scheme). A commitment scheme is a pair of algorithms $\text{CS} = (\text{Com}, \text{Verify})$ with an associated message space $\mathcal{M}$ and randomness space $\mathcal{R}$, where $\text{Com}(m, r)$ maps a message $m \in \mathcal{M}$ and randomness $r \in \mathcal{R}$ to a commitment C and a decommitment D , and $\text{Verify}(C, m, D) \in \{0, 1\}$ verifies that C has decommitment (m, D) ; when m is clear from context, we also call D alone a decommitment. We require the scheme to be correct, i.e., $\text{Verify}(C, m, D) = 1$ whenever $(C, D) := \text{Com}(m, r)$ for some $m \in \mathcal{M}$ and $r \in \mathcal{R}$.

We call a commitment scheme *canonical* if the decommitment is the randomness itself and verification consists of recomputing the commitment, i.e., $\text{Com}(m, r)$ outputs (C, r) and $\text{Verify}(C, m, r)$ accepts iff $(m, r) \in \mathcal{M} \times \mathcal{R}$ and C is the commitment output by $\text{Com}(m, r)$. Since the decommitment output of a canonical scheme is just the input r , we omit it and write $\text{Com}(m, r)$ for the commitment C alone. The Pedersen and ElGamal commitment schemes below are canonical, but the partial commitment schemes of the switch commitment schemes we consider ([Sect. 4](#)) are not: their partial verification does recompute a Pedersen commitment, but the partial decommitment is not the input randomness.

Security is captured by two properties: *binding*, which ensures that a commitment can be opened to only one message, and *hiding*, which ensures that a commitment reveals nothing about the message it holds.

Definition 3 (Binding). The advantage of an adversary $\mathcal{A}$ against the binding property of (the verification algorithm Verify of) a commitment scheme CS is

$$\text{Adv}_{\text{CS}}^{\text{bind}}(\mathcal{A}) := \Pr[\text{Verify}(C, m_0, D_0) = 1 \wedge \text{Verify}(C, m_1, D_1) = 1 \wedge m_0 \neq m_1],$$

where $(C, m_0, m_1, D_0, D_1) := \mathcal{A}()$. If $\text{Adv}_{\text{CS}}^{\text{bind}}(\mathcal{A}) = 0$ for every unbounded adversary $\mathcal{A}$, the scheme is perfectly binding.

Definition 4 (Hiding). The advantage of an adversary $\mathcal{A}$ against the hiding property of (the commitment algorithm Com of) a commitment scheme CS is

$$\text{Adv}_{\text{CS}}^{\text{hide}}(\mathcal{A}) := \max_{m_0, m_1 \in \mathcal{M}} \left| \Pr[\text{Hide}_{\text{CS}}^{\mathcal{A}}(m_0) = 1] - \Pr[\text{Hide}_{\text{CS}}^{\mathcal{A}}(m_1) = 1] \right|,$$

where the game $\text{Hide}_{\text{CS}}^{\mathcal{A}}(m_b)$ samples $r \leftarrow \mathcal{R}$, computes $(C, D) := \text{Com}(m_b, r)$, and returns $\mathcal{A}(C)$ (i.e., the adversary sees the commitment but not the decommitment). If $\text{Adv}_{\text{CS}}^{\text{hide}}(\mathcal{A}) = 0$ for every unbounded adversary $\mathcal{A}$, the scheme is perfectly hiding.

CT relies on Pedersen commitments [Ped92], whose homomorphic property enables checking the balance equation (Sect. 2) on commitments alone. In CT, the message space is the set of admissible amounts, e.g., $\mathcal{M} = [0, 2^{52})$ in Liquid, and the range proof accompanying a commitment is what enforces membership in $\mathcal{M}$ in practice.

Construction 1 (Pedersen Commitments). *The Pedersen commitment scheme PedCom over a message space $\mathcal{M} \subseteq \mathbb{Z}_q$ is the canonical commitment scheme with randomness space $\mathcal{R} = \mathbb{Z}_q$ and*

$$\text{PedCom.Com}(m, r) := mH + rG.$$

PedCom is perfectly hiding because, for every m , the commitment $mH + rG$ is uniformly distributed in $\mathbb{G}$. Binding holds computationally and reduces to DL: an adversary that finds decommitments (m_0, r_0) and (m_1, r_1) of one commitment with $m_0 \neq m_1$ obtains $\log_G H = (r_1 - r_0)/(m_0 - m_1)$ (a formal reduction programs HashToGroup("H") with its DL challenge).

PedCom is *additively homomorphic*: for all $m_0, r_0, m_1, r_1 \in \mathbb{Z}_q$, we have $\text{PedCom.Com}(m_0, r_0) + \text{PedCom.Com}(m_1, r_1) = \text{PedCom.Com}(m_0 + m_1, r_0 + r_1)$, since both sides equal $(m_0 + m_1)H + (r_0 + r_1)G$.

ElGamal commitments extend Pedersen commitments by a second group element rK , formed either with $K = H$ or with $K = J$, and are dual to them: they are also homomorphic, but binding is perfect, since rK determines r and then the first component determines m , whereas hiding holds only computationally, under the decisional Diffie–Hellman assumption.

Construction 2 (ElGamal Commitments). *For $K \in \{H, J\}$, the ElGamal commitment scheme EGCom_K over a message space $\mathcal{M} \subseteq \mathbb{Z}_q$ is the canonical commitment scheme with randomness space $\mathcal{R} = \mathbb{Z}_q$ and*

$$\text{EGCom}_K.\text{Com}(m, r) := (mH + rG, rK).$$

4 Switch Commitments

In this section, we define *switch commitment schemes*; our treatment deviates from [RM17] in a few aspects as mentioned below. As compared to an ordinary commitment scheme, a switch commitment scheme has two verification modes called *partial verification* and *full verification*.

Definition 5 (Switch Commitment Scheme). *A switch commitment scheme is a tuple of algorithms $\text{SwCS} = (\text{SwCom}, \text{Verify}_{\text{part}}, \text{Verify}_{\text{full}})$ with an associated message space $\mathcal{M}$ and randomness space $\mathcal{R}$:*

- $\text{SwCom}(m, r) \rightarrow (c, p, f)$: Takes a message $m \in \mathcal{M}$ and randomness $r \in \mathcal{R}$, and outputs a commitment c , a partial decommitment p , and a full decommitment f .
- $\text{Verify}_{\text{part}}(c, m, p) \rightarrow b \in \{0, 1\}$ verifies that the commitment c has a partial decommitment (m, p) .
- $\text{Verify}_{\text{full}}(c, m, f) \rightarrow b \in \{0, 1\}$ verifies that the commitment c has a full decommitment (m, f) .

Correctness. Partial verification accepts every honestly generated commitment by definition. Correctness requires that full verification does so too, except with some small error probability δ . In the confidential transaction setting, δ bounds the probability that the switch renders honest funds unspendable, because an honest recipient’s full decommitment would fail full verification.

Definition 6 (Correctness). *A switch commitment scheme is correct with error δ if for all messages $m \in \mathcal{M}$*

$$\Pr \left[\text{Verify}_{\text{full}}(c, m, f) = 1 \mid (c, p, f) := \text{SwCom}(m, r) \right] \geq 1 - \delta.$$

If $\delta = 0$, we call the scheme perfectly correct.

Security. A switch commitment scheme has to satisfy three security properties. Two of them are properties of its *partial commitment scheme* $(\text{SwCom}, \text{Verify}_{\text{part}})$, i.e., the commitment scheme in the sense of Definition 2 obtained by dropping the full decommitment f from the output of SwCom . Before the switch, the network checks commitments only with partial verification, so the partial commitment scheme must be binding (Definition 3) against classical adversaries; we call this *partial binding*. As soon as quantum attacks are considered a realistic concern, the network can perform an upgrade to switch to full verification. Across the switch, the scheme must be *switch binding* (Definition 7 below), which guarantees that commitments created and opened to one message by a classical adversary (before the switch) cannot be opened to a different message even by a quantum adversary (after the switch). Finally, the scheme must be *post-quantum hiding*, i.e., the partial commitment scheme must be hiding (Definition 4) even against quantum adversaries, which guarantees confidentiality even after the switch. (Since a hiding adversary sees only the commitment c , which both verification modes share, this is a property of SwCom alone, and we write $\text{Adv}_{\text{SwCS}}^{\text{hide}}(\mathcal{A})$ for its advantage.) Unlike [RM17], we do not require $(\text{SwCom}, \text{Verify}_{\text{full}})$ to form a post-quantum binding commitment scheme. That requirement would matter only if fresh switch commitments could be created after the switch. They cannot in our setting: full verification is applied only to legacy switch commitments created before the switch, for which switch binding provides the required guarantee. New outputs instead use a different commitment scheme—either that of the post-quantum CT scheme or, in the emergency opt-in fallback (Sect. 6), the perfectly binding EGCom_K . Moreover, the switch binding and hiding definitions in [RM17] model the quantum adversary after the switch simply as computationally unbounded, effectively yielding a purely statistical security notion after the switch. (Switch binding was accordingly called *everlasting binding*.) In contrast, our definitions model it as a quantum oracle algorithm, which allows us to quantify the advantage of a quantum adversary that does a bounded number of queries to a quantum random oracle.

Definition 7 (Switch Binding). A switch binding adversary *against a switch commitment scheme* SwCS is a pair $\mathcal{A} = (\mathcal{A}_c, \mathcal{A}_q)$ of a classical algorithm $\mathcal{A}_c$, which acts before the switch, and a quantum oracle algorithm $\mathcal{A}_q$, which acts after it and receives the state of $\mathcal{A}_c$. Its advantage is

$$\text{Adv}_{\text{SwCS}}^{\text{swbind}}(\mathcal{A}) := \Pr \left[\text{SwBind}_{\text{SwCS}}^{\mathcal{A}} = 1 \right],$$

where the game $\text{SwBind}_{\text{SwCS}}^{\mathcal{A}}$ is defined as follows:

$\text{SwBind}_{\text{SwCS}}^{\mathcal{A}}$
$(c, m, p, \text{state}) := \mathcal{A}_c^{\text{H}}()$
if $c = \perp$ then return 0
$(m', f) := \mathcal{A}_q^{\text{H}}(\text{state})$
return $m \neq m' \wedge \text{Verify}_{\text{part}}(c, m, p) = 1 \wedge \text{Verify}_{\text{full}}(c, m', f) = 1$

Intuitively, switch binding guarantees that a commitment fixed by a classical adversary before the switch cannot later be opened to a different message, even by a quantum adversary. The game splits the adversary into two stages along this timeline. The classical algorithm $\mathcal{A}_c$ first outputs a commitment c with a partial decommitment (m, p) , and the quantum algorithm $\mathcal{A}_q$ then receives its state and wins if it produces a full decommitment (m', f) for a different message $m' \neq m$.

In a CT network, the partial decommitment is never handed to the network. Every output carries a range proof instead, which is a proof of knowledge of a partial decommitment (m, p) of the commitment with m in the admissible range (Sect. 2). The partial decommitment that $\mathcal{A}_c$ outputs in the game is thus the one that a knowledge extractor for the range proof extracts from an adversary that creates a valid output before the switch, and m is the amount that the network has accepted for that output.

Note that the adversary creates the commitment it later reopens, which is exactly the attack a currency has to fear: an attacker who wants to inflate the supply creates an output of its own before the switch and reopens it to a larger amount afterwards.

We remark that, in general, asking that an adversary cannot produce two decommitments of one commitment can be a delicate requirement because a quantum adversary may be able to open a commitment to a message of its choice but, due to the quantum state collapse when doing so, not to two different messages simultaneously [ARU14]. However, switch binding is not affected by the issue because the first decommitment (m, p) is output by the classical stage $\mathcal{A}_c$, so it is a classical value fixed before the quantum stage $\mathcal{A}_q$ starts.

5 Coincidental Switch Commitments

This section shows that a switch commitment scheme is *already deployed* in Liquid, Monero, and Zano. Because it was not designed to serve this purpose, we call it the *coincidental switch commitment scheme*.

The key observation is that every CT network needs a mechanism for conveying the decommitment of a Pedersen commitment, i.e., the amount m and the randomness r , confidentially to the recipient of an output in a transaction. In every network, an honest recipient will learn a *seed* η from which it can derive a symmetric key k via a key derivation function K . The recipient can then use this key k to decrypt a ciphertext on chain and recover (m, r) . We formally capture the underlying encryption scheme, which differs from network to network, as *decommitment encryption* (Sect. 5.1).

A coincidental switch commitment is then the pair (C, ct) of the Pedersen commitment C and the decommitment ciphertext ct . Partial verification $\text{Verify}_{\text{part}}$ verifies the decommitment (m, p) of C as before, while full verification $\text{Verify}_{\text{full}}$ additionally derives the key k from the seed η in the full decommitment, decrypts ct under it, and checks that the result matches (m, p) . We prove that the scheme is switch binding and post-quantum hiding in the quantum random oracle model (Sect. 5.2).

How the Recipient Learns the Seed. In Liquid, Monero, and Zano, the sender conveys the seed η by encapsulating it to a public key in the recipient’s payment address using ECDH as a key encapsulation mechanism (KEM). The KEM ciphertext is stored in the output on the blockchain.

Instantiations of Decommitment Encryption. Deployed networks differ in how encryption works and what the ciphertext is. In Liquid, the range proof itself serves as the ciphertext: the prover derives all of its randomness from k , and the recipient recovers the witness (m, r) from the proof by rederiving randomness from k ; we call range proofs with this functionality *witness-recoverable*.³ Liquid’s Confidential Assets protocol [Poe+19] uses a witness-recoverable variant of Back-Maxwell range proofs, and we show that these range proofs constitute a decommitment encryption, which yields CoCom for Liquid (Sect. 5.3). In the CryptoNote-based networks Monero [KAN20] and Zano [sow24], the output carries the ciphertext in a dedicated field, where the key k consists of a one-time pad that masks the amount and of the commitment randomness, and we show that this masked-amount field constitutes a decommitment encryption (Sect. 5.4).

Implications. As a result, every output of Liquid, Monero, and Zano contains a coincidental switch commitment. None of these networks has deployed switch commitments deliberately, yet all of them have been quantum-upgradable all along: they can activate full verification and migrate every existing output to a post-quantum CT scheme without expiring any funds.

We note that Liquid’s Confidential Assets [Poe+19], an extension to CT, enables users to use multiple assets (e.g., multiple currencies) in one transaction and additionally hides the asset type of every output in a second Pedersen-based commitment. We focus on amounts in this paper but we sketch how CoCom can be extended to Confidential Assets in App. B.

5.1 Decommitment Encryption

We formally define *decommitment encryption* relative to an arbitrary canonical commitment scheme CS; in the deployed networks, CS is PedCom.

³ Grin uses witness-recoverable Bulletproofs [Bün+18] for the same purpose [Gri20; Gri19b]. We do not work this case out, because Grin has deliberately deployed an opt-in switch commitment scheme and is thus quantum-upgradable in any case (Sect. 6) but we describe it informally at the end of Sect. 5.3.

Definition 8 (Decommitment Encryption). Let CS be a canonical commitment scheme with algorithms $(\text{Com}, \text{Verify})$, message space $\mathcal{M}$, and randomness space $\text{CS}.\mathcal{R}$ (Sect. 3.2). A decommitment encryption $\text{DE}[\text{CS}]$ for CS is a pair of algorithms that encrypts messages from the message space $\mathcal{M}$ of CS , with a key space $\text{DE}.\mathcal{K}$ and a randomness space $\text{DE}.\mathcal{R}$:

- $\text{Enc}(k, m, \omega) \rightarrow (ct, r)$: Encrypts a message $m \in \mathcal{M}$ using the key $k \in \text{DE}.\mathcal{K}$ with randomness $\omega \in \text{DE}.\mathcal{R}$, returning a ciphertext ct and randomness $r \in \text{CS}.\mathcal{R}$, which is then used to form the commitment $C = \text{CS}.\text{Com}(m, r)$.
- $\text{Dec}(k, C, ct) \rightarrow (m, r)$: Decrypts a ciphertext ct using the key $k \in \text{DE}.\mathcal{K}$ and commitment C , returning the decommitment $(m, r) \in \mathcal{M} \times \text{CS}.\mathcal{R}$ if successful, or $\perp$ otherwise.

The randomness space $\text{DE}.\mathcal{R}$ may be a singleton; then Enc is deterministic.

The syntax deviates from ordinary encryption wherever the deployed schemes require it. The randomness r of the commitment is an output of Enc rather than an input, because Monero derives it from the key (Sect. 5.4), whereas Liquid chooses it freely and passes it in as ω . Moreover, Dec receives the commitment C , because recovering the decommitment from a range proof needs it (Sect. 5.3), whereas the masked-amount scheme ignores it.

Correctness. Decryption has to recover the decommitment that the sender committed to.

Definition 9 (Correctness). A decommitment encryption $\text{DE}[\text{CS}]$ for a commitment scheme CS is correct with error δ if for every $m \in \mathcal{M}$ and every $\omega \in \text{DE}.\mathcal{R}$, the following holds for $k \leftarrow \$ \text{DE}.\mathcal{K}$, $(ct, r) := \text{Enc}(k, m, \omega)$, and $C := \text{CS}.\text{Com}(m, r)$:

$$\Pr[\text{Dec}(k, C, ct) \neq (m, r)] \leq \delta.$$

Security. A decommitment encryption has to satisfy two security properties: the first is what switch binding of CoCom will reduce to (Theorem 1), the second what its post-quantum hiding will reduce to (Theorem 2).

The first property is that no ciphertext can be made to decrypt to a decommitment of the commitment it accompanies, except under the key it was created with. This is close to the binding notions studied for *committing* authenticated encryption [GLR17; BH22; CR22] and to robustness of public-key encryption [ABN10], where a ciphertext must not decrypt to meaningful plaintexts under two different keys. For our application, we need a notion that differs from these in three respects, which we call *random-key robustness*. First, the key is not chosen by the adversary but is uniform and independent of the ciphertext, which the game models by drawing it after the ciphertext is fixed; CoCom derives the key from a seed through the random oracle K and thus provides this for every seed that the adversary has not queried. Second, the bound is statistical in the sense that it must hold for every commitment and ciphertext (equivalently, for a commitment and ciphertext produced by an unbounded adversary). Third, what counts as a meaningful decryption is fixed by a separate commitment C : the result must be a decommitment of C and not merely a plaintext, which is what ties the ciphertext to the commitment the network already stores.

The second property is that the ciphertext must not leak the message, which we capture as indistinguishability as for ordinary encryption. CoCom never reuses a key, since every commitment samples a fresh seed η and with it a fresh key $k = \mathsf{K}(\eta)$, so a notion in which the adversary sees a single ciphertext and is given no encryption oracle is all we need.

Definition 10 (Random-Key Robustness). Write

$$\mathcal{D}(C) := \{(m, r) \in \mathcal{M} \times \text{CS}.\mathcal{R} : \text{CS}.\text{Verify}(C, m, r) = 1\}$$

for the set of decommitments of a commitment C under a commitment scheme CS . A decommitment encryption $\text{DE}[\text{CS}]$ for CS is random-key robust with error ε if for every (possibly maliciously generated) commitment C and ciphertext ct ,

$$\Pr[\text{RKRob}_{\text{DE}}(C, ct) = 1] \leq \varepsilon,$$

where the game RKRob_{DE} is defined in Fig. 1.

$\text{RKRob}_{\text{DE}}(C, ct)$	$\text{Ind}_{\text{DE}}^{\mathcal{A}}(m_b)$
$k \leftarrow \$ \text{DE}.\mathcal{K}$	$k \leftarrow \$ \text{DE}.\mathcal{K}; \omega \leftarrow \$ \text{DE}.\mathcal{R}$
return $\text{Dec}(k, C, ct) \in \mathcal{D}(C)$	$(ct, r) := \text{Enc}(k, m_b, \omega)$
	$C := \text{CS}.\text{Com}(m_b, r)$
	return $\mathcal{A}(C, ct)$

Fig. 1. The games of random-key robustness (Definition 10) and indistinguishability (Definition 11) of a decommitment encryption $\text{DE}[\text{CS}]$.

Definition 11 (Indistinguishability). A decommitment encryption $\text{DE}[\text{CS}]$ for a commitment scheme CS is indistinguishable with error ε if for all adversaries $\mathcal{A}$ and all messages $m_0, m_1 \in \mathcal{M}$,

$$|\Pr[\text{Ind}_{\text{DE}}^{\mathcal{A}}(m_0) = 1] - \Pr[\text{Ind}_{\text{DE}}^{\mathcal{A}}(m_1) = 1]| \leq \varepsilon,$$

where the game $\text{Ind}_{\text{DE}}^{\mathcal{A}}$ is defined in Fig. 1. If $\varepsilon = 0$, it is perfectly indistinguishable.

5.2 Coincidental Commitments

The following construction turns a decommitment encryption for CS into a switch commitment scheme.

Construction 3 (Coincidental Switch Commitments). Let CS be a canonical commitment scheme with algorithms $(\text{Com}, \text{Verify})$, message space $\mathcal{M}$, and randomness space $\text{CS}.\mathcal{R}$, let $\text{DE}[\text{CS}] = (\text{Enc}, \text{Dec})$ be a decommitment encryption for CS , and let $\text{K}: \{0, 1\}^\ell \rightarrow \text{DE}.\mathcal{K}$ be a hash function. The coincidental switch commitment scheme $\text{CoCom}[\text{CS}, \text{DE}[\text{CS}]]$ has message space $\mathcal{M}$ and randomness space $\text{DE}.\mathcal{R} \times \{0, 1\}^\ell$, and is defined as follows, with Enc 's randomness as the partial decommitment:

$\text{SwCom}(m, r)$	$\text{Verify}_{\text{part}}(c, m, p)$	$\text{Verify}_{\text{full}}(c, m, f)$
$(\omega, \eta) := r$	$(C, ct) := c$	$(p, \eta) := f$
$k := \text{K}(\eta)$	if $C = \perp \vee ct = \perp$ then	if $\text{Verify}_{\text{part}}(c, m, p) = 0$ then
$(ct, p) := \text{DE}.\text{Enc}(k, m, \omega)$	return 0	return 0
$C := \text{CS}.\text{Com}(m, p)$	return $\text{CS}.\text{Verify}(C, m, p)$	$(C, ct) := c$
$c := (C, ct)$		$k := \text{K}(\eta)$
$f := (p, \eta)$		$(m', p') := \text{DE}.\text{Dec}(k, C, ct)$
return (c, p, f)		return $m = m' \wedge p = p'$

If $\text{DE}[\text{CS}]$ has correctness error δ , then CoCom is correct with error δ in the random oracle model for K : full verification rejects an honest commitment only if $\text{DE}.\text{Dec}$ fails to recover (m, p) under the key $k = \text{K}(\eta)$, which is uniform for a uniform seed η . CoCom is also partially binding: partial verification checks only the component C under CS , so two partial decommitments of $c = (C, ct)$ to different messages are two decommitments of C to different messages; hence partial binding reduces directly to binding of CS .

For switch binding, recall that the seed η is part of the full decommitment. A winning adversary $(\mathcal{A}_c, \mathcal{A}_q)$ thus has to exhibit a seed η whose key $k = \text{K}(\eta)$ decrypts ct to a decommitment of C to a message $m' \neq m$, where $c = (C, ct)$ and m are fixed by $\mathcal{A}_c$ before the switch. Either $\mathcal{A}_c$ has queried K at such a conflicting seed or not. If it has, a classical reduction that runs only $\mathcal{A}_c$ finds the seed among its queries by trial decryption and obtains two decommitments of C to different messages, breaking binding of CS . If it has not, the key at every unqueried seed is uniform and independent of (C, ct, m) , so random-key robustness of $\text{DE}[\text{CS}]$ (Definition 10) bounds the probability that a given seed conflicts by ε , and Lemma 2 applies to the search of $\mathcal{A}_q$ for a conflicting seed.

Theorem 1 (Switch Binding). Let $\text{DE}[\text{CS}]$ be a decommitment encryption with random-key robustness error ε for a commitment scheme CS , where DE.Dec and CS.Verify run in time t_{Dec} and t_{Verify} , and let $\mathcal{A} = (\mathcal{A}_c, \mathcal{A}_q)$ be a switch binding adversary against $\text{CoCom}[\text{CS}, \text{DE}[\text{CS}]]$ in the quantum random oracle model for K , where $\mathcal{A}_c$ runs in time t and makes d_c classical queries to K , and $\mathcal{A}_q$ has query depth d_q and query width w_q . Then there is a classical adversary $\mathcal{B}_c$ against the binding property of CS , running in time $t' \approx t + d_c \cdot (t_{\text{Dec}} + t_{\text{Verify}})$, such that

$$\text{Adv}_{\text{CoCom}}^{\text{swbind}}(\mathcal{A}) \leq \text{Adv}_{\text{CS}}^{\text{bind}}(\mathcal{B}_c) + 4(d_q + 1)^2 w_q \varepsilon.$$

Proof. Consider the game $\text{SwBind}_{\text{CoCom}}^{\mathcal{A}}$ after the classical algorithm $\mathcal{A}_c^{\text{K}}()$ has finished with some output (c, m, p, state) , where $c = (C, ct)$, and let Q_c be the set of K queries performed by $\mathcal{A}_c$. Let E be the event that there is $\eta \in Q_c$ such that $(m', p') := \text{DE.Dec}(\text{K}(\eta), C, ct)$ satisfies $m \neq m' \wedge \text{Verify}_{\text{part}}(c, m, p) \wedge \text{Verify}_{\text{full}}(c, m', (p', \eta))$.

Case E. If E happens, then the following reduction $\mathcal{B}_c^{\mathcal{A}_c}()$ to the binding property of CS will be successful.

```

 $\mathcal{B}_c^{\mathcal{A}_c}()$ 


---


 $(c, m, p, \text{state}) := \mathcal{A}_c^{\text{K}}()$ 
if  $c = \perp \vee \text{Verify}_{\text{part}}(c, m, p) = 0$  then return  $\perp$ 
 $(C, ct) := c$ 
for  $\eta \in Q_c$  do // the  $\text{K}$  queries of  $\mathcal{A}_c$ 
     $k := \text{K}(\eta)$ 
     $(m', p') := \text{DE.Dec}(k, C, ct)$ 
    if  $(m', p') = \perp \vee m' = m$  then continue
    if  $\text{CS.Verify}(C, m', p') = 1$  then
        return  $(C, m, m', p, p')$  // two decommitments of  $C$  with  $m \neq m'$ 
return  $\perp$ 

```

The reduction $\mathcal{B}_c$ runs $\mathcal{A}_c$ once, verifies $\mathcal{A}_c$'s own decommitment, and performs one decryption and one verification for each of the at most d_c queries in Q_c , so it runs in time $t' \leq t + d_c \cdot t_{\text{Dec}} + (d_c + 1) t_{\text{Verify}}$ (note that it never calls $\mathcal{A}_q$). It samples the random oracle K itself (lazily, as $\mathcal{A}_c$ queries it), so $\mathcal{A}_c$'s view is exactly as in $\text{SwBind}_{\text{CoCom}}^{\mathcal{A}}$. Hence E implies that $\mathcal{B}_c$ returns two decommitments (m, p) and (m', p') of the same commitment C for different messages, i.e., that $\mathcal{B}_c$ wins the binding game:

$$\Pr[E] = \Pr[\mathcal{B}_c^{\mathcal{A}_c}() \neq \perp] \leq \text{Adv}_{\text{CS}}^{\text{bind}}(\mathcal{B}_c). \quad (4)$$

Case $\neg E$. Since the game rejects $c = \perp$ and its winning condition includes $\text{Verify}_{\text{part}}(c, m, p) = 1$, a run in which $\mathcal{A}_c$ does not output a well-formed commitment that passes partial verification cannot be won. It therefore suffices to bound the winning probability over the *admissible* executions of $\mathcal{A}_c$, i.e., those that satisfy $\neg E$ and output a commitment $c = (C, ct) \neq \perp$ together with a partial decommitment (m, p) such that $\text{Verify}_{\text{part}}(c, m, p) = 1$; in particular $(m, p) \in \mathcal{D}(C)$. We fix such an execution for the remainder of this case; this fixes Q_c , the values $\text{K}(\eta)$ for $\eta \in Q_c$, and state , and leaves the values $\text{K}(\eta)$ for $\eta \notin Q_c$ independent and uniform, so [Lemma 2](#) applies to $\mathcal{A}_q$ with auxiliary input $z := \text{state}$.

Under $\neg E$, no key $\text{K}(\eta)$ with $\eta \in Q_c$ decrypts ct to an alternative valid decommitment, so $\mathcal{A}_q$ has to find a fresh seed $\eta \notin Q_c$ for which $\text{DE.Dec}(\text{K}(\eta), C, ct)$ returns a decommitment $(m', p') \in \mathcal{D}(C)$ with $m' \neq m$. Note that $\mathcal{A}_q$ may be computationally unbounded, so it can in particular find such a decommitment (m', p') of C on its own: binding of CS is assumed only against the classical $\mathcal{A}_c$, and the guarantee against $\mathcal{A}_q$ rests instead on the requirement that this decommitment also be the output of DE.Dec .

Define the boolean predicate

$$\theta(\eta, \text{K}(\eta)) = 1 \iff (m', p') := \text{DE.Dec}(\text{K}(\eta), C, ct) : (m', p') \in \mathcal{D}(C) \wedge m' \neq m.$$

For every $\eta \in Q_c$ the value $K(\eta)$ is fixed, and $\neg E$ gives $\theta(\eta, K(\eta)) = 0$. For every $\eta \notin Q_c$ the key $K(\eta)$ is uniformly distributed over $\text{DE}.\mathcal{K}$, and $\theta(\eta, K(\eta)) = 1$ implies that $\text{RKRob}_{\text{DE}}(C, ct)$ returns 1 for the key $k = K(\eta)$, so random-key robustness of $\text{DE}[\text{CS}]$ (Definition 10) gives $\Pr[\theta(\eta, K(\eta)) = 1] \leq \varepsilon$. The predicate therefore satisfies the hypothesis of Lemma 2 with bound ε at every $\eta \in \{0, 1\}^\ell$. Moreover, a winning run of the game satisfies $m \neq m'$ and $\text{Verify}_{\text{full}}(c, m', (p', \eta))$, so the seed η in the full decommitment (p', η) returned by $\mathcal{A}_q$ satisfies $\theta(\eta, K(\eta)) = 1$. By Lemma 2, we have

$$\begin{aligned} \Pr[\text{SwBind}_{\text{CoCom}}^{\mathcal{A}} = 1 \mid \neg E] &\leq \Pr[\theta(\eta, K(\eta)) = 1 \mid (m', (p', \eta)) := \mathcal{A}_q^{\mathcal{K}}(\text{state})] \\ &\leq 4(d_q + 1)^2 w_q \varepsilon. \end{aligned} \quad (5)$$

Summing. Combining (4) and (5), we obtain

$$\begin{aligned} \text{Adv}_{\text{CoCom}}^{\text{swbind}}(\mathcal{A}) &\leq \Pr[E] + \Pr[\text{SwBind}_{\text{CoCom}}^{\mathcal{A}} = 1 \mid \neg E] \\ &\leq \text{Adv}_{\text{CS}}^{\text{bind}}(\mathcal{B}_c) + 4(d_q + 1)^2 w_q \varepsilon. \end{aligned} \quad \square$$

Theorem 2 (Post-Quantum Hiding). *Let $\text{DE}[\text{CS}]$ be a decommitment encryption with indistinguishability error ε for a commitment scheme CS , and let $\mathcal{A} = \mathcal{A}_q$ be a quantum oracle algorithm with query depth d_q and query width w_q . Then, in the quantum random oracle model for K , $\text{CoCom}[\text{CS}, \text{DE}[\text{CS}]]$ satisfies*

$$\text{Adv}_{\text{CoCom}}^{\text{hide}}(\mathcal{A}) \leq 4d_q \sqrt{\frac{w_q}{2^\ell}} + \varepsilon.$$

The proof is a standard game hopping proof that applies the O2H lemma (Lemma 1) once.

Proof of Theorem 2. The proof proceeds by a sequence of hops as defined in Fig. 2, using the O2H lemma (Lemma 1): we show that

$$\text{Hide}_{\text{CoCom}}^{\mathcal{A}}(m_b) = \text{Game}_1^{\mathcal{A}}(m_b) \approx \text{Game}_2^{\mathcal{A}}(m_b) = \text{Game}_3^{\mathcal{A}}(m_b)$$

for each $b \in \{0, 1\}$, and that $\text{Game}_3^{\mathcal{A}}(m_0) \approx \text{Game}_3^{\mathcal{A}}(m_1)$. Fix a quantum oracle algorithm $\mathcal{A}$ with query depth d_q and query width w_q , a message pair (m_0, m_1) , and a bit $b \in \{0, 1\}$.

Hop $\text{Hide}_{\text{CoCom}}^{\mathcal{A}} \rightarrow \text{Game}_1$. $\text{Hide}_{\text{CoCom}}^{\mathcal{A}}(m_b)$ is the hiding game of Definition 4 for CoCom with the commitment algorithm SwCom of Construction 3 inlined and the random oracle K explicit. The code of $\text{Game}_1^{\mathcal{A}}(m_b)$ only adds the assignments of the variables k' and K' . Since these additional assignments are dead stores, this change does not influence the return value of the game, and thus

$$\left| \Pr[\text{Hide}_{\text{CoCom}}^{\mathcal{A}}(m_b) = 1] - \Pr[\text{Game}_1^{\mathcal{A}}(m_b) = 1] \right| = 0. \quad (6)$$

Hop $\text{Game}_1 \rightarrow \text{Game}_2$. The code of $\text{Game}_2^{\mathcal{A}}(m_b)$ is identical to that of $\text{Game}_1^{\mathcal{A}}(m_b)$ with the difference that $\mathcal{A}$ is given oracle access to the function $K' := K[\eta := k']$ instead of K , where $k' \leftarrow \$ \text{DE}.\mathcal{K}$ is freshly sampled. In other words, K' is a random function that is identical to K except at the seed η , at which its value has been resampled.

In order to bound the probability that $\mathcal{A}$ distinguishes K and K' , we apply Lemma 1 instantiated with $S = \{\eta\}$ and $z = c$ to $\mathcal{A}$. Here, the values η, k, K, K', c have the same distribution as the values of the corresponding variables in $\text{Game}_1^{\mathcal{A}}(m_b)$ and $\text{Game}_2^{\mathcal{A}}(m_b)$; note that the code assigning these variables is identical in the two games. This yields

$$\begin{aligned} &\left| \Pr[\text{Game}_1^{\mathcal{A}}(m_b) = 1] - \Pr[\text{Game}_2^{\mathcal{A}}(m_b) = 1] \right| \\ &= \left| \Pr[\mathcal{A}^K(c) = 1 \text{ in } \text{Game}_1^{\mathcal{A}}] - \Pr[\mathcal{A}^{K'}(c) = 1 \text{ in } \text{Game}_2^{\mathcal{A}}] \right| \\ &\leq 2d_q \sqrt{\Pr[\mathcal{E}^{K'}(c) \cap S \neq \emptyset \text{ in } \text{Game}_2^{\mathcal{E}}(m_b)]}. \end{aligned} \quad (7)$$

As will be shown below, $\text{Game}_2(m_b)$ and $\text{Game}_3(m_b)$ induce the same joint distribution of all their variables, whichever algorithm they run, in particular of the seed η and that algorithm's output. Applying this to $\mathcal{E}$, we have

$$\Pr[\mathcal{E}^{K'}(c) \cap S \neq \emptyset \text{ in } \text{Game}_2^{\mathcal{E}}(m_b)] = \Pr[\mathcal{E}^{K'}(c) \cap S \neq \emptyset \text{ in } \text{Game}_3^{\mathcal{E}}(m_b)]. \quad (8)$$

$\text{Hide}_{\text{CoCom}}^A(m_b)$	$\text{Game}_1^A(m_b)$
$K \leftarrow \$ \text{DE}.\mathcal{K}^{\{0,1\}^\ell}$	$K \leftarrow \$ \text{DE}.\mathcal{K}^{\{0,1\}^\ell}$
$\omega \leftarrow \$ \text{DE}.\mathcal{R}$	$\omega \leftarrow \$ \text{DE}.\mathcal{R}$
$\eta \leftarrow \$ \{0,1\}^\ell$	$\eta \leftarrow \$ \{0,1\}^\ell$
$k := K(\eta)$	$k := K(\eta)$
$(ct, p) := \text{DE}.\text{Enc}(k, m_b, \omega)$	$(ct, p) := \text{DE}.\text{Enc}(k, m_b, \omega)$
$C := \text{CS}.\text{Com}(m_b, p)$	$C := \text{CS}.\text{Com}(m_b, p)$
$c := (C, ct)$	$c := (C, ct)$
 	$k' \leftarrow \$ \text{DE}.\mathcal{K}$
 	$K' := K[\eta := k']$
return $\mathcal{A}^K(c)$	return $\mathcal{A}^K(c)$
$\text{Game}_2^A(m_b)$	$\text{Game}_3^A(m_b)$
$K \leftarrow \$ \text{DE}.\mathcal{K}^{\{0,1\}^\ell}$	$K' \leftarrow \$ \text{DE}.\mathcal{K}^{\{0,1\}^\ell}$
$\omega \leftarrow \$ \text{DE}.\mathcal{R}$	$\omega \leftarrow \$ \text{DE}.\mathcal{R}$
$\eta \leftarrow \$ \{0,1\}^\ell$	$\eta \leftarrow \$ \{0,1\}^\ell$
$k := K(\eta)$	$k \leftarrow \$ \text{DE}.\mathcal{K}$
$(ct, p) := \text{DE}.\text{Enc}(k, m_b, \omega)$	$(ct, p) := \text{DE}.\text{Enc}(k, m_b, \omega)$
$C := \text{CS}.\text{Com}(m_b, p)$	$C := \text{CS}.\text{Com}(m_b, p)$
$c := (C, ct)$	$c := (C, ct)$
$k' \leftarrow \$ \text{DE}.\mathcal{K}$	
$K' := K[\eta := k']$	
return $\mathcal{A}^{K'}(c)$	return $\mathcal{A}^{K'}(c)$

Fig. 2. Sequence of games for the proof of [Theorem 2](#). Boxed lines indicate changes from the preceding game. The commitment algorithm SwCom of [Construction 3](#) is inlined in the games.

Hop $\text{Game}_2 \rightarrow \text{Game}_3$. In Game_2^A , the random value $K(\eta)$ is used only in the assignment to k . (Recall that $\mathcal{A}$ is not given K but only K' , which is identical to K except at the seed η where the value $K'(\eta)$ has been resampled.) Thus $K(\eta)$ is distributed independently of η , of ω and of K' , i.e., of everything in the game except the values computed from it, and can be replaced by a freshly sampled key $k \leftarrow \$ \text{DE.K}$ without changing the joint distribution. Similarly, K' can be replaced by a freshly sampled random oracle without changing its distribution. As a result, the two games induce the same joint distribution, and since these are the only differences between $\text{Game}_2^A(m_b)$ and $\text{Game}_3^A(m_b)$, we have

$$\left| \Pr[\text{Game}_2^A(m_b) = 1] - \Pr[\text{Game}_3^A(m_b) = 1] \right| = 0. \quad (9)$$

In $\text{Game}_3^\mathcal{E}(m_b)$, the inputs to $\mathcal{E}^{K'}(c)$, namely the fresh random oracle $K' \leftarrow \$ \text{DE.K}^{\{0,1\}^\ell}$ and $c = (C, ct)$ computed from the freshly sampled key $k \leftarrow \$ \text{DE.K}$, are distributed independently of $S = \{\eta\}$. Thus $\mathcal{E}^{K'}(c)$ has no better chance of outputting a set of size at most w_q containing the only element η of S than guessing w_q different values of $\eta \in \{0, 1\}^\ell$, i.e.,

$$\Pr[\mathcal{E}^{K'}(c) \cap S \neq \emptyset \text{ in } \text{Game}_3^\mathcal{E}(m_b)] \leq \frac{w_q}{2^\ell}. \quad (10)$$

By combining (7), (8) and (10), we get

$$\left| \Pr[\text{Game}_1^A(m_b) = 1] - \Pr[\text{Game}_2^A(m_b) = 1] \right| \leq 2d_q \sqrt{\frac{w_q}{2^\ell}}. \quad (11)$$

Hop $\text{Game}_3(m_0) \rightarrow \text{Game}_3(m_1)$. In $\text{Game}_3^A(m_b)$, the key $k \leftarrow \$ \text{DE.K}$ is uniform and independent of the oracle K' , so the pair $c = (C, ct)$ is distributed exactly as in $\text{Ind}_{\text{DE}}(m_b)$ (Definition 11), and K' is sampled independently of c . The algorithm $\mathcal{B}$ that, on input (C, ct) , samples $K' \leftarrow \$ \text{DE.K}^{\{0,1\}^\ell}$ itself and returns $\mathcal{A}^{K'}((C, ct))$ is therefore an adversary against indistinguishability of $\text{DE}[\text{CS}]$ with $\Pr[\text{Ind}_{\text{DE}}^\mathcal{B}(m_b) = 1] = \Pr[\text{Game}_3^A(m_b) = 1]$ for each b , so indistinguishability with error ε gives

$$\left| \Pr[\text{Game}_3^A(m_0) = 1] - \Pr[\text{Game}_3^A(m_1) = 1] \right| \leq \varepsilon. \quad (12)$$

Summing. Combining (6), (9), (11) and (12) along the sequence of hops yields

$$\begin{aligned} & \left| \Pr[\text{Hide}_{\text{CoCom}}^A(m_0) = 1] - \Pr[\text{Hide}_{\text{CoCom}}^A(m_1) = 1] \right| \\ & \leq \left| \Pr[\text{Hide}_{\text{CoCom}}^A(m_0) = 1] - \Pr[\text{Game}_3^A(m_0) = 1] \right| \\ & \quad + \left| \Pr[\text{Game}_3^A(m_0) = 1] - \Pr[\text{Game}_3^A(m_1) = 1] \right| \\ & \quad + \left| \Pr[\text{Game}_3^A(m_1) = 1] - \Pr[\text{Hide}_{\text{CoCom}}^A(m_1) = 1] \right| \\ & \leq 4d_q \sqrt{\frac{w_q}{2^\ell}} + \varepsilon. \end{aligned}$$

The message pair was arbitrary, so the same bound holds for $\text{Adv}_{\text{CoCom}}^{\text{hide}}(\mathcal{A})$. $\square$

5.3 Application to Liquid

The Liquid sidechain runs the Confidential Assets protocol [Poe+19], which uses a witness-recoverable range proof; we recall that proof system here and show that it constitutes a de-commitment encryption which instantiates coincidental switch commitments for Liquid.

Back-Maxwell Range Proofs. A Back-Maxwell range proof allows a prover to prove that a Pedersen-committed value m lies within the range $\mathcal{M} = [0, B^n)$ where $B \geq 2$ is the base, i.e., the

size of each ring, and $n \geq 2$ is the number of digits, i.e., the number of rings; thus it is the proof system for the following relation:

$$R_{\text{RP}} = \{(C; m, r) : C = mH + rG, m \in \mathcal{M}\}$$

The deployed version [Ele26] differs from the published one [Poe+19] in the one point that matters here: it generates the proof deterministically, with all of the prover’s randomness derived from key material that the key derivation function K expands from a seed shared by sender and recipient. In the implementation, this expansion takes the commitment, the generator and a short proof header as further inputs besides the seed [Blo26b]. Since all of them are fixed before the switch, the resulting map from the seed to the key material is, for every fixed commitment, generator and header, again a random oracle, and [Construction 3](#) applies with K instantiated by that map. A recipient who holds the seed can therefore recover the witness from the proof, which we abstract as the *witness recovery* algorithm RPWitRec ; it corresponds to `secp256k1_rangeproof_rewind` in `secp256k1-zkp` [Blo26b]. At a high level, the proof decomposes the committed value into digits and proves with a sequence of ring signatures that each digit is valid.

Definition 12 (Back-Maxwell Range Proof). *Let $\mathbb{G}$ be a cyclic group of prime order q and generator $G \in \mathbb{G}$, let $\mathcal{M} = [0, B^n)$ with $n \geq 2$ and $B \geq 2$ be the message space, let $\mathcal{R} = \mathbb{Z}_q$ be the randomness space, and let $\mathcal{K} = \mathbb{Z}_q^{n(B+1)-1}$ be the space of key material. Let $\phi : \mathcal{M} \rightarrow \{0, 1\}^L$ be an injective message encoding function whose image does not contain the all-zero string 0^L , and let $H : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$ be a cryptographic hash function. The Back-Maxwell range proof scheme BMRP consists of the following three algorithms, spelled out in [Figs. 3–5](#):*

- $\text{RPProve}(C, m, r, k) \rightarrow \pi$: Creates a range proof π for commitment $C = mH + rG$ using the key material k . Here $m \in \mathcal{M}$ and $r \in \mathcal{R}$.
- $\text{RPVerify}(C, \pi) \rightarrow b \in \{0, 1\}$: Verifies range proof π for commitment C .
- $\text{RPWitRec}(C, \pi, k) \rightarrow (m, r)$: Recovers the witness (m, r) from proof π using the key material k , or returns $\perp$.

The variable M in the figures is a Borromean context string that commits to the commitment point C , the alternative generator H , the proof header, and the sub-commitments $C_0 \dots C_{n-2}$, i.e., $M = H(C \parallel H \parallel \text{header} \parallel C_0 \parallel \dots \parallel C_{n-2})$.

Remark 1. The actual implementation of BMRP also embeds the asset tag A and asset blinder s in the forged signatures. Since A and s are not referenced in any of our proofs or scheme descriptions, we omit them from the algorithm inputs and outputs. However, we retain the embedding logic (shown on a gray background) to illustrate how switch commitments are extended to confidential assets in [App. B](#).

Remark 2. Note that while the signature values $s_{i,j}$ are formally elements of the scalar field $\mathbb{Z}_q$, we simplify our notation by applying the bitwise XOR operator $(\oplus)$ directly to them. It is implicitly understood that an appropriate encoding function converts these field elements to bitstrings prior to the operation. The corresponding inverse function is applied when assigning the result back to $s_{i,j}$.

Remark 3. We intentionally omit several implementation details such as the implementation of the hash to scalar H to keep the notation clear and focused on the core cryptographic operations.

The construction is a slightly changed version of Borromean ring signatures [MP15], with one addition that matters for us: the committed amount is piggybacked to the last forged signature in the last ring. This lets the receiver of a transaction recover the amount inside the commitment, while the commitment stays hiding against everyone else. Note that the embedding slot is never the true slot m_{n-1} of the last ring, so the final overwrite $s_{i,m_i} := u_i - e_{i,m_i-1}x_i$ in RPProve cannot clobber the embedded amount.

It is immediate that the proof system $(\text{RPProve}, \text{RPVerify})$ is perfectly complete. In this work we additionally show that BMRP has perfect witness indistinguishability and constitutes a decommitment encryption scheme.

```

RPProve( $C, m, r, A, s, k$ )
// Write  $m$  in base  $B$  as  $m = \sum_{i=0}^{n-1} m_i B^i$ 
// Parse key material to obtain ring secrets and random scalars for ring signature
 $\{x_i\}_{i=0}^{n-2}, \{otp_{i,j}\} := k$ 
// Initialize all  $s_{i,j}$  from the key material
 $\{s_{i,j}\} := \{otp_{i,j}\}$ 
// Embed  $m$  into the highest-indexed forged signature of the last ring
if  $m_{n-1} = B - 1$  then  $s_{n-1,B-2} := s_{n-1,B-2} \oplus \phi(m)$ 
else  $s_{n-1,B-1} := s_{n-1,B-1} \oplus \phi(m)$ 
// Embed the asset ID  $A$  and the asset blinder  $s$  into the first ring's first two signatures
 $s_{0,0} := s_{0,0} \oplus A$ 
 $s_{0,1} := s_{0,1} \oplus s$ 
 $x_{n-1} := r - \sum_{i=0}^{n-2} x_i$  // Set last ring blinder so  $\sum x_i = r$ 
// Compute sub-commitments  $C_i$  upfront for the context message
for  $i \in \{0, \dots, n-1\}$  do
   $C_i := m_i B^i H + x_i G$ 
 $M := H(C \parallel H \parallel \text{header} \parallel C_0 \parallel \dots \parallel C_{n-2})$ 
for  $i \in \{0, \dots, n-1\}$  do
   $u_i := s_{i,m_i}$ 
   $R_{i,m_i} := u_i G$ 
   $e_{i,m_i} := H(R_{i,m_i} \parallel M \parallel i \parallel m_i)$ 
  for  $j \in \{m_i + 1, \dots, B-1\}$  do
     $R_{i,j} := e_{i,j-1}(C_i - j B^i H) + s_{i,j} G$ 
     $e_{i,j} := H(R_{i,j} \parallel M \parallel i \parallel j)$ 
 $e_0 := H(R_{0,B-1} \parallel \dots \parallel R_{n-1,B-1} \parallel M)$ 
for  $i \in \{0, \dots, n-1\}$  do
   $e_{i,-1} := e_0$ 
  for  $j \in \{0, \dots, m_i - 1\}$  do
     $R_{i,j} := e_{i,j-1}(C_i - j B^i H) + s_{i,j} G$ 
     $e_{i,j} := H(R_{i,j} \parallel M \parallel i \parallel j)$ 
   $s_{i,m_i} := u_i - e_{i,m_i-1} x_i$  // Overwrites pre-generated value for true signatures
return  $\pi = \{e_0, (C_0, s_{0,0}, \dots, s_{0,B-1}), \dots, (C_{n-1}, s_{n-1,0}, \dots, s_{n-1,B-1})\}$ 

```

Fig. 3. Proof generation of the Back-Maxwell range proof (Definition 12).

```

RPVerify( $C, \pi$ )
// Parse  $\pi$  as  $\{e_0, (C_0, s_{0,0}, \dots, s_{0,B-1}), \dots, (C_{n-1}, s_{n-1,0}, \dots, s_{n-1,B-1})\}$ 
 $M := H(C \parallel H \parallel \text{header} \parallel C_0 \parallel \dots \parallel C_{n-2})$ 
for  $i \in \{0, \dots, n-1\}$  do
     $e_{i,-1} := e_0$ 
    for  $j \in \{0, \dots, B-1\}$  do
         $R_{i,j} := e_{i,j-1}(C_i - jB^i H) + s_{i,j}G$ 
         $e_{i,j} := H(R_{i,j} \parallel M \parallel i \parallel j)$ 
return  $(H(R_{0,B-1} \parallel \dots \parallel R_{n-1,B-1} \parallel M) = e_0) \wedge (C = \sum_{i=0}^{n-1} C_i)$ 

```

Fig. 4. Verification of the Back-Maxwell range proof (Definition 12).

```

RPWitRec( $C, \pi, k$ )
// Parse  $\pi$  as  $\{e_0, (C_0, s_{0,0}, \dots, s_{0,B-1}), \dots, (C_{n-1}, s_{n-1,0}, \dots, s_{n-1,B-1})\}$ 
 $M := H(C \parallel H \parallel \text{header} \parallel C_0 \parallel \dots \parallel C_{n-2})$ 
 $\{x_i\}_{i=0}^{n-2}, \{otp_{i,j}\} := k$ 
// Attempt to extract the embedded message
if  $\phi^{-1}(s_{n-1,B-1} \oplus otp_{n-1,B-1}) \neq \perp$  then  $m := \phi^{-1}(s_{n-1,B-1} \oplus otp_{n-1,B-1})$ 
elseif  $\phi^{-1}(s_{n-1,B-2} \oplus otp_{n-1,B-2}) \neq \perp$  then  $m := \phi^{-1}(s_{n-1,B-2} \oplus otp_{n-1,B-2})$ 
else return  $\perp$ 

// Write  $m$  in base  $B$  as  $m = \sum_{i=0}^{n-1} m_i B^i$ 
// Extract Asset ID  $A$  and Asset blinder  $s$  from the first ring
if  $m_0 = 0$  then  $A := (s_{0,0} + e_0 x_0) \oplus otp_{0,0}$ 
else  $A := s_{0,0} \oplus otp_{0,0}$ 
if  $m_0 = 1$  then
     $R_{0,0} := e_0 C_0 + s_{0,0} G$ 
     $e_{0,0} := H(R_{0,0} \parallel M \parallel 0 \parallel 0)$ 
     $s := (s_{0,1} + e_{0,0} x_0) \oplus otp_{0,1}$ 
else  $s := s_{0,1} \oplus otp_{0,1}$ 
// Recompute hash chain for the last ring up to  $m_{n-1} - 1$ 
 $e_{n-1,-1} := e_0$ 
for  $j \in \{0, \dots, m_{n-1} - 1\}$  do
     $R_{n-1,j} := e_{n-1,j-1}(C_{n-1} - jB^{n-1} H) + s_{n-1,j} G$ 
     $e_{n-1,j} := H(R_{n-1,j} \parallel M \parallel n-1 \parallel j)$ 
// Recover effective blinding factor and extract  $r$ 
 $u_{n-1} := otp_{n-1,m_{n-1}}$  // Retrieve the recycled true nonce
 $x_{n-1}^{\text{eff}} := (u_{n-1} - s_{n-1,m_{n-1}}) \cdot e_{n-1,m_{n-1}-1}^{-1}$ 
 $r := x_{n-1}^{\text{eff}} + \sum_{i=0}^{n-2} x_i$ 
return  $(m, r, A, s)$ 

```

Fig. 5. Witness recovery of the Back-Maxwell range proof (Definition 12).

Lemma 3 (Witness Indistinguishability). *Over uniformly random key material k , BMRP is perfectly witness-indistinguishable.*

Proof. We show that for any commitment C and any proof π accepted by RPVerify , i.e., any $\pi = (e_0, \{C_i, s_{i,0}, \dots, s_{i,B-1}\}_{i=0}^{n-1})$, the probability that the prover RPProve outputs π is exactly the same for any valid witness $(m, r) \in \mathcal{D}(C)$ (recall from [Definition 10](#) that, for PedCom , $(m, r) \in \mathcal{D}(C) \Leftrightarrow mH + rG = C \wedge m \in \mathcal{M}$):

$$\Pr[\pi = \text{RPProve}(C, m, r, k) \mid k \leftarrow \mathcal{K}] = 1/q^{n(B+1)-1}$$

Let $(m, r) \in \mathcal{D}(C)$ be a valid witness, where $m = \sum_{i=0}^{n-1} m_i B^i$. During proof generation, all internal randomness is parsed from the key material k . Specifically, for each digit index $i \in \{0, \dots, n-2\}$, k provides a blinding factor $x_i \in \mathbb{Z}_q$, a random nonce $u_i \in \mathbb{Z}_q$, and $B-1$ forged responses $s_{i,j} \in \mathbb{Z}_q$ for all $j \neq m_i$. Because k is uniformly distributed, these components are uniform and independently distributed over the space $\Omega_i = \mathbb{Z}_q \times \mathbb{Z}_q \times \mathbb{Z}_q^{B-1}$. Note that all challenges $e_{i,j}$ are derived deterministically using the hash function H from the proof π so they do not contribute to the prover's randomness distribution.

Below we identify, for each digit i , the components of k whose values are necessary and sufficient for the prover to output the fixed proof π : necessary because π pins each value down uniquely, and sufficient because running RPProve on the resulting k replays the same commitments, challenges, and responses and reproduces π .

- To output the specific commitment $C_i \in \pi$, $i \neq n-1$, the key material k must contain x_i such that $C_i = m_i B^i H + x_i G$. Since C_i, m_i are fixed, there is exactly one such $x_i \in \mathbb{Z}_q$, namely $x_i := \log_G(C_i - m_i B^i H)$. The last ring secret x_{n-1} is fully determined by the previous x_i 's and the fixed value of r as $x_{n-1} := r - \sum_{i=0}^{n-2} x_i$.
- To output the specific forged signatures in π , k must contain the exact scalars $otp_{i,j} = s_{i,j}$ for all $j \neq m_i, i \neq n-1$. For the last ring it must contain $otp_{n-1,j} = s_{n-1,j} \oplus \phi(m)$ for position $j = B-2$ if $m_{n-1} = B-1$, or for position $j = B-1$ otherwise; the values at the other forged positions j must be equal to $s_{n-1,j}$ as in the previous case. There is exactly one choice for these $B-1$ values in both cases.
- To output the true response $s_{i,m_i} \in \pi$, k must provide u_i such that $s_{i,m_i} = u_i - e_{i,m_i-1} x_i$. Because s_{i,m_i} is fixed by π , e_{i,m_i-1} is deterministically fixed by the proof and hash function H , and x_i is uniquely determined as shown above, there is exactly one required $u_i \in \mathbb{Z}_q$ that satisfies this equation.

Therefore, for any valid witness digit $m_i, i \neq n-1$, there is exactly one unique required component tuple in k out of the $|\mathbb{Z}_q|^{B+1} = q^{B+1}$ possibilities that produces the components of π associated with digit i . Since k is uniformly distributed, the probability that it yields this exact tuple is $1/q^{B+1}$, which is strictly independent of the value of m_i . For the last digit m_{n-1} , the required components of k are also uniquely determined by the same logic with the exception that x_{n-1} is already determined by previous x_i , and the probability that k yields these components is $1/q^B$.

As all key-material components $(s_{i,j}, u_i, x_i)$ are independent of each other we can apply the product law. The overall probability that k generates the complete proof π is exactly $1/q^{n(B+1)-1}$. Since this exact probability is identical for any valid witness (m, r) , the protocol is perfectly witness-indistinguishable. $\square$

Back-Maxwell Range Proofs as Decommitment Encryption. We show that Back-Maxwell range proofs can be used to construct a decommitment encryption scheme. This fact is crucial for building CoCom in Liquid .

Lemma 4 (BMRP as a Decommitment Encryption). *BMRP induces a decommitment encryption $\text{DE}[\text{PedCom}]$ for PedCom over $\mathbb{G}$ with message space $\mathcal{M}$ and randomness space $\text{PedCom}.\mathcal{R} = \mathbb{Z}_q$ ([Construction 1](#)), with key space $\text{DE}.\mathcal{K} = \mathcal{K}$ and randomness space $\text{DE}.\mathcal{R} = \text{PedCom}.\mathcal{R}$, via the following algorithms:*

$\text{Enc}(k, m, \omega)$	$\text{Dec}(k, C, ct)$
$r := \omega$	return RPWitRec(C, ct, k)
$C := mH + rG$	
$ct := \text{RPProve}(C, m, r, k)$	
return (ct, r)	

It has correctness error $\delta \leq (|\mathcal{M}| - 1)/(q - 1)$ in the random oracle model for H , random-key robustness error $\varepsilon \leq 1/q$ and perfect indistinguishability.

Proof. We show that (Enc, Dec) as defined above is correct (Definition 9), random-key robust (Definition 10), and indistinguishable (Definition 11).

Correctness. This is the one property for which we model H as a random oracle; the probability below is taken over the key material $k \leftarrow \mathcal{K}$ and over H . Fix a message $m \in \mathcal{M}$ with base- B digits $m = \sum_{i=0}^{n-1} m_i B^i$ and randomness $\omega = r \in \mathcal{R}$, and let $\pi = \text{RPProve}(C, m, r, k)$ be an honestly generated proof for $C = mH + rG$.

Run on the same key material k , RPWitRec reparses the identical blinders $\{x_i\}_{i=0}^{n-2}$ and pads $\{otp_{i,j}\}$ that RPProve used and replays the same hash chain, so it recomputes exactly the challenges RPProve produced; since all random oracle queries within one execution of RPProve have pairwise distinct inputs (each challenge query carries a distinct index suffix $i \parallel j$, and the queries defining M and e_0 have formats different from these and from each other), these challenges are uniform in $\mathbb{Z}_q^*$ and independent of k . Hence RPWitRec inverts precisely the equations RPProve evaluated and returns (m, r) , provided it decodes the message from the slot where RPProve embedded it: slot $B - 1$ if $m_{n-1} \neq B - 1$, and slot $B - 2$ if $m_{n-1} = B - 1$, in which case slot $B - 1$ instead carries a true signature.

Since the message embedding in the last slot is conditional on m_{n-1} it can occur that for $m_{n-1} = B - 1$ (and the message m is embedded in slot $B - 2$) the first trial decoding of the true signature spuriously succeeds on a wrong message, i.e., $\phi^{-1}(s_{n-1,B-1} \oplus otp_{n-1,B-1}) = m^*$ for some $m^* \in \mathcal{M} \setminus \{m\}$. We denote this bad event as F_{dec} . Its probability is precisely the correctness error δ as the recovery is correct whenever $\neg F_{\text{dec}}$ happens.

To bound F_{dec} , write $v := s_{n-1,B-1} \oplus otp_{n-1,B-1} = (u_{n-1} - e x_{n-1}) \oplus u_{n-1}$, where $e := e_{n-1,B-2}$, $u_{n-1} := otp_{n-1,B-1}$, and $x_{n-1} = r - \sum_{i=0}^{n-2} x_i$ is the last ring blinder. If $x_{n-1} = 0$, the challenge drops out and $v = u_{n-1} \oplus u_{n-1} = 0^L$ deterministically, which lies outside the image of ϕ (Definition 12), so the trial decoding fails and F_{dec} cannot occur. Otherwise e is uniform in $\mathbb{Z}_q^*$ and independent of k , so $u_{n-1} - e x_{n-1}$ is uniform over $\mathbb{Z}_q \setminus \{u_{n-1}\}$; since XOR with the fixed encoding of u_{n-1} is injective, v is uniform over a set of $q - 1$ bitstrings, which hits the target set $\phi(\mathcal{M} \setminus \{m\})$ of size $|\mathcal{M}| - 1$ with probability at most $(|\mathcal{M}| - 1)/(q - 1)$. Thus

$$\delta = \Pr[F_{\text{dec}}] = \Pr[v \in \phi(\mathcal{M} \setminus \{m\})] \leq \frac{|\mathcal{M}| - 1}{q - 1}.$$

Random-Key Robustness. Fix a commitment C and a proof $\pi = \{e_0, \{C_i, s_{i,j}\}_{i,j}\}$, and draw fresh uniform key material k , so that its components $\{x_i\}_{i=0}^{n-2}, \{otp_{i,j}\}$ are uniform and mutually independent in $\mathbb{Z}_q$. On (C, π, k) , RPWitRec first decodes a message

$$\begin{aligned} \mu &:= \phi^{-1}(s_{n-1,B-1} \oplus otp_{n-1,B-1}) \quad \text{or, if that fails,} \\ \mu &:= \phi^{-1}(s_{n-1,B-2} \oplus otp_{n-1,B-2}), \end{aligned}$$

returning $\perp$ if both decoding attempts fail, and then computes the blinder

$$\rho := (otp_{n-1,\mu_{n-1}} - s_{n-1,\mu_{n-1}}) e_{n-1,\mu_{n-1}-1}^{-1} + \sum_{i=0}^{n-2} x_i,$$

where μ_{n-1} denotes the top base- B digit of μ . The recovered pair (μ, ρ) lies in $\mathcal{D}(C)$ exactly when $\mu = m'$ for some $m' \in \mathcal{M}$ and additionally $\rho = \log_G(C - m'H)$. The events $\{\mu = m'\}_{m' \in \mathcal{M}}$ are

Game₀^A = Ind_{DE}^A(m₀)	Game₁^A
$r \leftarrow \mathcal{R}$	$r \leftarrow \mathcal{R}$
$k \leftarrow \mathcal{K}$	$k \leftarrow \mathcal{K}$
$C := m_0 H + rG$	$C := m_1 H + \sigma(r)G$
$ct := \text{RPProve}(C, m_0, r, k)$	$ct := \text{RPProve}(C, m_0, r, k)$
return $\mathcal{A}^H(C, ct)$	return $\mathcal{A}^H(C, ct)$
Game₂^A	Game₃^A = Ind_{DE}^A(m₁)
$r \leftarrow \mathcal{R}$	$r \leftarrow \mathcal{R}$
$k \leftarrow \mathcal{K}$	$k \leftarrow \mathcal{K}$
$C := m_1 H + \sigma(r)G$	$C := m_1 H + rG$
$ct := \text{RPProve}(C, m_1, \sigma(r), k)$	$ct := \text{RPProve}(C, m_1, r, k)$
return $\mathcal{A}^H(C, ct)$	return $\mathcal{A}^H(C, ct)$

Fig. 6. Sequence of games for the proof of indistinguishability of the BMRP decommitment encryption (Lemma 4). Boxed lines indicate changes from the preceding game. The algorithm Enc is inlined in the games.

pairwise disjoint and cover both decoding branches, so

$$\begin{aligned} \Pr[(\mu, \rho) \in \mathcal{D}(C)] \\ = \sum_{m' \in \mathcal{M}} \Pr[\rho = \log_G(C - m'H) \mid \mu = m'] \cdot \Pr[\mu = m']. \end{aligned}$$

We claim that $\Pr[\rho = \log_G(C - m'H) \mid \mu = m'] \leq 1/q$ for each $m' \in \mathcal{M}$. Observe that both the event $\{\mu = m'\}$ and the first summand of ρ are determined by the pads $\{otp_{n-1,j}\}_j$ alone, since the scalars $s_{i,j}$ and challenges $e_{i,j}$ are fixed by π . The blinders $x_0, \dots, x_{n-2}$ are distinct components of k , hence uniform and independent of those pads; as $n \geq 2$ the sum $\sum_{i=0}^{n-2} x_i$ is non-empty and acts as a one-time pad on ρ , making it uniform in $\mathbb{Z}_q$ conditioned on $\{\mu = m'\}$. Each conditional probability is therefore at most $1/q$. By disjointness of events $\{\mu = m'\}$ we have $\sum_{m'} \Pr[\mu = m'] \leq 1$ and therefore

$$\Pr[(\mu, \rho) \in \mathcal{D}(C)] \leq \frac{1}{q} \sum_{m' \in \mathcal{M}} \Pr[\mu = m'] \leq \frac{1}{q}.$$

Indistinguishability. Fix two messages $m_0, m_1 \in \mathcal{M}$ and oracle algorithm $\mathcal{A}$. Let $x = \log_G H \in \mathbb{Z}_q$. For $b \in \{0, 1\}$, let $r_b \in \mathbb{Z}_q$ denote the unique scalar such that $C = m_b H + r_b G$ for a given $C \in \mathbb{G}$. Define the permutation $\sigma : \mathbb{Z}_q \rightarrow \mathbb{Z}_q$ by $\sigma(r) = r + (m_0 - m_1)x$, so that $r_1 = \sigma(r_0)$.

The proof proceeds by a sequence of game hops from $\text{Ind}_{\text{DE}}^A(m_0)$ to $\text{Ind}_{\text{DE}}^A(m_1)$.

Hop Game₀ = Game₁. C is computed differently but the result is identical.

Hop Game₁ = Game₂. By perfect witness indistinguishability (Lemma 3).

Hop Game₂ = Game₃. Since σ is a permutation and r is chosen uniformly at random, r and $\sigma(r)$ are distributed identically.

All four games therefore have the same output distribution, so $\text{Ind}_{\text{DE}}^A(m_0)$ and $\text{Ind}_{\text{DE}}^A(m_1)$ return 1 with the same probability for every $\mathcal{A}$, i.e., the error is 0. $\square$

Instantiating Construction 3 with the Back-Maxwell decommitment encryption (Lemma 4) yields $\text{CoCom}[\text{PedCom}, \text{DE}[\text{PedCom}]]$, already deployed in Liquid: every Pedersen commitment there is accompanied by a Back-Maxwell range proof, generated deterministically from a per-output seed η derived from the ECDH shared secret. In Liquid's Confidential Assets extension, the generator

H of the value commitment is replaced by a per-output asset commitment, which we treat as the fixed generator H here; [App. B](#) treats the asset commitment itself. We now record what this gives concretely. Grin follows the same pattern through a witness-recoverable variant of Bulletproofs, and we describe it informally at the end.

Concrete Security. In Liquid, PedCom is instantiated over `secp256k1`, whose group order is $q \approx 2^{256}$; the seed η is a 256-bit hash output [Ele26], so $\ell = 256 \approx \log_2 q$; and amounts are 52 bits wide, so $\mathcal{M} = [0, 2^{52})$, which the range proof covers with $n = 26$ rings of size $B = 4$ [Blo26b]. The decommitment encryption of [Lemma 4](#) has correctness error $\delta \leq (|\mathcal{M}| - 1)/(q - 1) \approx 2^{-204}$, so an honest recipient’s decommitment is essentially never rejected.

With the random-key robustness error $\varepsilon \leq 1/q$ of [Lemma 4](#), [Theorem 1](#) bounds switch binding of the deployed scheme by $\text{Adv}_{\text{PedCom}}^{\text{bind}}(\mathcal{B}_c) + 4(d_q + 1)^2 w_q / q$, where $\mathcal{B}_c$ runs in time $t' \approx t + (B + 2)d_c t_{\text{mul}}$ because DE.Dec performs B scalar multiplications to recompute the hash chain of the last ring and PedCom.Verify performs two, and binding of PedCom reduces to DL on `secp256k1` ([Sect. 3.2](#)).

[Theorem 2](#) bounds post-quantum hiding by $4d_q \sqrt{w_q/q}$, provided the seed η stays hidden, which requires replacing ECDH with a quantum-secure key encapsulation mechanism. The replacement protects only outputs created afterwards: for outputs already on the chain, a quantum adversary breaks the ECDH key exchange and recovers η . For the attacker of [Table 1](#), where a scalar multiplication costs $t_{\text{mul}} \approx 2^8$ group operations, the reduction runs in time about 2^{111} , which gives the switch binding entry of the table.

Coincidental Switch Commitments in Grin. We conjecture that CoCom can also be instantiated with the witness-recoverable version of Bulletproofs [Bün+18] deployed in Grin [Gri20; Gri19b]. Unlike Liquid, which we treated in full detail, we describe the setting only informally and leave a proof to future work, since Grin has deployed the deliberate switch commitment construction OptCom in any case ([Sect. 6](#)), and the commitments in Grin are certainly post-quantum upgradable via that path.

Grin implements the Mimblewimble protocol [Jed16], so every output carries a Pedersen value commitment $C = mH + pG$ together with a Bulletproofs range proof [Bün+18]. Grin is the one network where both constructions we describe in this work are already present: the randomness is derived using a tweak, and the range proof is witness-recoverable.

Unlike in Liquid, in Grin the randomness is not chosen freely. Instead it is derived from a freely chosen r as

$$p = r + H(mH + rG, rJ)$$

for a third generator J , so that the output commitment becomes $C = mH + pG$. This is OptCom, the opt-in scheme deployed in Grin ([Sect. 6](#)).

From the other side, Grin’s range proofs are generated deterministically as well, but not from a secret shared with a sender. Grin transactions are built interactively, so the recipient creates its own output and range proof, and it derives two nonces for the proof from the commitment and hashes of its wallet’s root keys [Gri19b]. The first nonce determines the prover randomness from which the amount and the derivation path of r in the wallet’s key hierarchy can be recovered, and the second nonce determines the randomness from which p can be recovered. A wallet uses only the first nonce for recovery and re-derives p from the path, which is what lets it recover its outputs from the chain, and what lets a view-only wallet holding a hash of the public root key see amounts without being able to spend [Gri19b]. We conjecture that the two hashed root keys can serve as the seed η of a decommitment encryption whose ciphertext is the range proof and whose key derivation function K expands the seed, together with the commitment, into the two nonces and thus into the prover randomness, so that Grin’s Bulletproofs instantiate CoCom just as the Back-Maxwell proof does for Liquid. Unlike in Liquid and Monero, this seed belongs to the wallet rather than to a single output, so a proof would have to account for its reuse across the outputs of one wallet.

5.4 Application to Monero and Zano

Monero hides amounts with RingCT [NMM16], so every output carries a Pedersen commitment $C = mH + rG$ to its amount m . Zano follows the same design [sow24] and adds a confidential asset

layer, so its amount commitment is $C = mT + rG$ with a blinded asset generator T in place of H . The asset layer changes nothing that matters here, so we write $C = mH + rG$ for both networks and treat them together.

Both the randomness r and the encryption of the amount are derived from the per-output shared secret η that sender and recipient establish through ECDH. The sender expands η into a randomness and a pad, $(r, u) := K(\eta)$, forms $C = mH + rG$, and publishes the masked amount $\hat{m} = m \oplus u$ next to C [KAN20, Section 5.3]. The recipient recomputes η and thus (r, u) , recovers $m = \hat{m} \oplus u$, and checks that $C = mH + rG$. Monero and Zano differ only in the details of the expansion and of asset blinding [Mon26; Zan26].

The two verification modes of a switch commitment are already visible here: the check $C = mH + rG$ is partial verification, and recomputing (m, r) from η and $\hat{m}$ and comparing it with the claimed decommitment is full verification. We now make this precise: the mechanism is exactly a coincidental switch commitment. The masked-amount field is a decommitment encryption with key (r, u) (Lemma 5 below), the shared secret is the seed η of Construction 3, and the expansion is its key derivation function K .

Masked-Amount Decommitment Encryption. We capture the masked-amount field as a decommitment encryption whose key holds the randomness r and the pad u used to mask the message.

Lemma 5 (Masked-Amount Decommitment Encryption). *Let $\phi : \mathcal{M} \rightarrow \{0, 1\}^L$ be an injective message encoding function with $\phi^{-1}(y) = \perp$ off its image. The masked-amount encryption induces a decommitment encryption $\text{DE}[\text{PedCom}]$ for PedCom over $\mathbb{G}$ with message space $\mathcal{M}$ and randomness space $\text{PedCom}.\mathcal{R} = \mathbb{Z}_q$ (Construction 1), with key space $\text{DE}.\mathcal{K} = \mathbb{Z}_q \times \{0, 1\}^L$ and a singleton randomness space $\text{DE}.\mathcal{R}$, so that Enc ignores ω , via the following algorithms:*

$\text{Enc}(k, m, \omega)$	$\text{Dec}(k, C, ct)$
$(r, u) := k$	$(r, u) := k$
$ct := \phi(m) \oplus u$	$m := \phi^{-1}(ct \oplus u)$
return (ct, r)	if $m = \perp$ then return $\perp$
	return (m, r)

It has perfect correctness ($\delta = 0$), random-key robustness error $\varepsilon = 1/q$, and perfect indistinguishability.

Proof. We verify Definitions 9–11 in turn.

Correctness. On an honest ciphertext $ct = \phi(m) \oplus u$, Dec unmask $\phi(m)$ under the same key $k = (r, u)$ and returns (m, r) as ϕ is injective, so $\delta = 0$.

Random-Key Robustness. Fix a commitment C and a ciphertext ct . In the game $\text{RKRob}_{\text{DE}}(C, ct)$, the key $k = (r, u)$ is uniform, i.e., $r \leftarrow \$ \mathbb{Z}_q$ and $u \leftarrow \$ \{0, 1\}^L$ are independent. On this input Dec returns (μ, r) with $\mu = \phi^{-1}(ct \oplus u)$, or $\perp$ if the decoding fails. The pair (μ, r) lies in $\mathcal{D}(C)$ exactly when $\mu = m'$ for some $m' \in \mathcal{M}$ and $r = \log_G(C - m'H)$. The message μ depends on u alone while the randomness r is an independent uniform scalar, so for each m' we have $\Pr[r = \log_G(C - m'H) \mid \mu = m'] = 1/q$. The events $\{\mu = m'\}_{m' \in \mathcal{M}}$ are pairwise disjoint since u decodes to at most one message, so summing over m' gives

$$\begin{aligned}
& \Pr[(\mu, r) \in \mathcal{D}(C)] \\
&= \sum_{m' \in \mathcal{M}} \Pr[r = \log_G(C - m'H) \mid \mu = m'] \cdot \Pr[\mu = m'] \\
&\leq \frac{1}{q} \sum_{m' \in \mathcal{M}} \Pr[\mu = m'] \leq \frac{1}{q}.
\end{aligned}$$

Indistinguishability. Fix two messages $m_0, m_1 \in \mathcal{M}$. Over a uniform key $k = (r, u)$, encrypting m_b reveals the pair $(C, ct) = (m_b H + rG, \phi(m_b) \oplus u)$. As u is uniform on $\{0, 1\}^L$, the masked amount ct is uniform on $\{0, 1\}^L$ and independent of m_b . As r is uniform on $\mathbb{Z}_q$ and independent of u , the commitment C is uniform on $\mathbb{G}$ and independent of both ct and m_b . Hence (C, ct) is uniform on $\mathbb{G} \times \{0, 1\}^L$ whatever b is, so $\Pr[\text{Ind}_{\text{DE}}^{\mathcal{A}}(m_0) = 1] = \Pr[\text{Ind}_{\text{DE}}^{\mathcal{A}}(m_1) = 1]$ for every $\mathcal{A}$, i.e., the error is 0. $\square$

Instantiating [Construction 3](#) with the masked-amount encryption ([Lemma 5](#)) thus yields $\text{CoCom}[\text{PedCom}, \text{DE}[\text{PedCom}]]$, which is exactly what every Monero and Zano output carries: a Pedersen commitment next to a masked amount, with the randomness and the pad derived from the seed η .

Concrete Security. In Monero and Zano, PedCom is instantiated over the prime-order subgroup of ed25519 , whose order is $q \approx 2^{252}$, and the shared secret η is the ECDH result hashed with Keccak-256 and reduced modulo q [[KAN20](#)], so $\ell \approx 252 \approx \log_2 q$. The masked-amount encryption of [Lemma 5](#) is perfectly correct, so an honest recipient’s decommitment is never rejected.

With the random-key robustness error $\varepsilon \leq 1/q$ of [Lemma 5](#), [Theorem 1](#) bounds switch binding of the deployed scheme by $\text{Adv}_{\text{PedCom}}^{\text{bind}}(\mathcal{B}_c) + 4(d_q + 1)^2 w_q / q$, where $\mathcal{B}_c$ runs in time $t' \approx t + 2d_c t_{\text{mul}}$ because DE.Dec performs no group operations and PedCom.Verify performs two scalar multiplications, and binding of PedCom reduces to DL on ed25519 ([Sect. 3.2](#)).

[Theorem 2](#) bounds post-quantum hiding by $4d_q \sqrt{w_q/q}$, provided the seed η stays hidden. This requires replacing ECDH with a quantum-secure key encapsulation mechanism, but that alone does not suffice: Monero obtains the one-time output key, i.e., the public key under which the output can be spent, by adding ηG to the recipient’s public spend key [[KAN20](#)], so a quantum adversary who knows the recipient’s address can recover η as a discrete logarithm. The two uses of η must therefore be separated, by deriving the seed and the offset of the one-time output key from the KEM plaintext under domain-separated derivations, which we believe is possible in the current protocol. Both changes protect only outputs created afterwards: for outputs already on the chain, a quantum adversary recovers η on either route. For the attacker of [Table 1](#), where a scalar multiplication costs $t_{\text{mul}} \approx 2^8$ group operations, the reduction runs in time about 2^{109} , which gives the switch binding entry of the table.

6 Opt-In Switch Commitments, Revisited

We now revisit the *opt-in* switch commitment scheme, suggested by Wuille and deployed by Grin [[Gri19a](#)]. It was put forward on the Mimbrawimble mailing list [[Ruf18](#)] as an improvement over the original construction of Ruffing and Malavolta [[RM17](#)], with a claim of security but no proof. A formal security analysis appeared later [[Ruf20](#), Section 6.5]; we show that what it claims does not hold.

The opt-in scheme constitutes a middle-ground between perfectly hiding but only computationally binding Pedersen commitments $mH + rG$, and the “dual” ElGamal commitments $\text{EGCom}_H.\text{Com}(m, r) = (mH + rG, rH)$ ([Construction 2](#)), which are perfectly binding but only computationally hiding. In more detail, the opt-in scheme shifts the randomness r of a Pedersen commitment $mH + rG$ by an additive “tweak” derived from the corresponding ElGamal commitment $(mH + rG, rH)$ via a random oracle. The new randomness becomes $p = r + H(mH + rG, rH)$, but the commitment retains the compact form of one group element. In fact, $c = mH + pG$ is also a valid Pedersen commitment with decommitment (m, p) , and the partial verification algorithm of the opt-in scheme checks precisely that.

The motivation for the opt-in scheme is to have a fallback for an emergency. In an ideal scenario, the network will have the post-quantum CT primitives ready before the switch, including the zero-knowledge proofs necessary to perform a privacy-preserving migration. But the switch might be forced early, for instance when a cryptographically relevant quantum computer suddenly appears and those primitives are not yet ready for deployment.

The opt-in scheme targets exactly this case, and it offers users a choice: After the switch, a user can, individually for each legacy output, opt to forgo the post-quantum hiding by revealing the

ElGamal commitment $(C_1, C_2) = (mH + rG, rH)$, which the network verifies with the last step of full verification, recomputing $c = C_1 + H(C_1, C_2)G$. The output can then behave as an ElGamal commitment to m , and the perfect binding property makes it possible that an ElGamal-based CT scheme (instead of a proper post-quantum scheme) can take over without the risk of inflation.

Because an ElGamal-based CT scheme is very close to the original Pedersen-based scheme, it is much easier to construct and implement. ElGamal commitments are additively homomorphic as well, so the balance equation (Sect. 2) carries over unchanged, and only range proofs need to be changed to cover the full ElGamal commitments. Suitable range proofs that remain sound against an unbounded prover can be constructed, e.g., from the folklore digit-wise OR-proof [CDS94; CGS97] of Chaum–Pedersen statements [CP93]; a construction that ports Back-Maxwell range proofs [Poe+19] for Pedersen commitments to ElGamal commitments [And17] has also been proposed, but it lacks a formal security analysis.

While an ElGamal commitment is not hiding against a quantum adversary, computational hiding against a classical adversary remains intact. Revealing (C_1, C_2) is therefore strictly better than opening m and r in the clear, and it keeps commitments binding while the proper post-quantum CT scheme is being rolled out.

Fixing the Security Analysis. Ruffing [Ruf20, Theorem 6.5] claims that the opt-in scheme is *statistically* switch binding, i.e., against adversaries that are not even bounded in the number of oracle queries they make. This does not hold:

For a counterexample, consider a switch binding adversary $\mathcal{A} = (\mathcal{A}_c, \mathcal{A}_q)$ (as in Definition 7 but with $\mathcal{A}_q$ unbounded) as follows. The classical part $\mathcal{A}_c$ outputs an honestly generated commitment c together with its decommitment (m, r) . A second candidate decommitment (m', r') with $m' \neq m$ is valid when $c = m'H + (r' + H(m'H + r'G, r'H))G$, i.e., when $H(m'H + r'G, r'H) = \log_G(c - m'H - r'G)$, which happens with probability $1/q$ over the choice of H . Since there are $(|\mathcal{M}| - 1) \cdot |\mathcal{R}| = (|\mathcal{M}| - 1) \cdot q$ candidate decommitments, hashed at pairwise distinct points and thus valid independently, on average $|\mathcal{M}| - 1$ of them will be valid, so for a message space as large as Grin’s, a second valid decommitment exists with high probability (and can be found by an unbounded $\mathcal{A}_q$).

Fortunately, the flaw lies only in assuming a too strong adversary,⁴ and we can save the construction by proving that it is *computationally* switch binding in the QROM if SqCDH is hard for classical adversaries. However, the reduction to SqCDH loses a factor of $|\mathcal{M}|$, so that for Grin’s $|\mathcal{M}| = 2^{64}$ we do not obtain convincing security bounds. Luckily, we can avoid the loss in the algebraic group model (AGM) [FKL18] by assuming that the classical part of the adversary is algebraic (Theorem 5; see Table 1).

A Generalized Construction. Formally, we introduce a generalized *tweaking scheme* parametrized by an injective *encoding function* $\varphi(m, r)$ that encodes the message and randomness before feeding it into H . For the opt-in scheme, the “encoding” is the full ElGamal commitment, i.e., $\varphi = \text{EGCom}_H.\text{Com}$.

Construction 4 (Tweaking Switch Commitments). *Let PedCom be the Pedersen commitment scheme over a message space $\mathcal{M} \subseteq \mathbb{Z}_q$ with randomness space $\mathcal{R} = \mathbb{Z}_q$ (Construction 1). Let $\varphi: \mathcal{M} \times \mathcal{R} \rightarrow \mathcal{X}$ be an injective encoding function into a set $\mathcal{X}$, and let $H: \mathcal{X} \rightarrow \mathbb{Z}_q$ be a cryptographic hash function. The Tweaking Switch Commitment Scheme TwCom $_{\varphi}$ is defined in Fig. 7.*

Full verification of an honest commitment always accepts, so tweaking switch commitments are perfectly correct.

This generalization enables us to consider the identity encoding function, i.e., $\varphi(m, r) = (m, r)$. The resulting *simplified tweaking scheme* SimpCom (detailed in Sect. 6.1) provides a different trade-off as compared to the opt-in scheme: In the emergency case, the user must reveal (m, r) directly, which forgoes privacy even against classical adversaries. However, what we get in return is a tight reduction to DL without resorting to the AGM, and a more efficient full verification

⁴ In the proof, the flaw shows up in the claim that “the choice of m' determines r' fully” [Ruf20, p. 127].

SwCom(m, r)	Verify _{part} (c, m, p)
$f := r$	if $c = \perp$ then return 0
$C := \text{PedCom.Com}(m, r)$	return PedCom.Verify(c, m, p)
$h := H(\varphi(m, r))$	Verify _{full} (c, m, f)
$c := C + hG$	if $c = \perp \vee m \notin \mathcal{M}$ then return 0
$p := r + h$	$r := f$
return (c, p, f)	$h := H(\varphi(m, r))$
	return PedCom.Verify($c, m, r + h$)

Fig. 7. The algorithms of TwCom _{φ} (Construction 4).

algorithm, which may pay off when performed as part of the statement of a zero-knowledge proof during privacy-preserving migration.⁵

Treating the opt-in scheme and SimpCom as two instantiations of the same generic construction is beneficial when it comes to the security analysis. In particular, we prove post-quantum hiding for *every* encoding admitted by the generic construction, and thereby cover both the opt-in scheme and SimpCom at once.

Switch binding is more involved: while we prove switch binding for SimpCom if DL is hard for classical adversaries (Theorem 6), the proof relies on φ being efficiently invertible.⁶ In more detail, the reduction in the “classical part” of the proof recovers an alternative decommitment already discovered by $\mathcal{A}_c$ directly from $\mathcal{A}_c$ ’s random-oracle queries, which works only because the identity function exposes it in the clear.

Since the ElGamal commitment is classically hiding and thus cannot be inverted by a classical reduction, this is not possible for the opt-in scheme, and we give a dedicated proof instead. But as already mentioned, this proof loses a factor of $|\mathcal{M}|$ as opposed to that for SimpCom and additionally reduces to the stronger SqCDH assumption. In the AGM, this multiplicative loss can be traded for an *additively* increased running time in a reduction to DL (Theorem 5), and the resulting theorem provides a meaningful bound (Table 1).

Grin’s Deployment. What Grin deploys is the instance OptCom, whose ElGamal commitment ECom _{J} uses the third generator J (Sect. 3) in the second component:

$$c = mH + (r + H(mH + rG, rJ))G.$$

We treat both schemes at once by writing $\varphi_K := \text{ECom}_K.\text{Com}$ for $K \in \{H, J\}$, so that OptCom’ = TwCom _{φ_H} and OptCom = TwCom _{φ_J} . Both encodings are injective because rK determines r .

Security. TwCom _{φ} is partially binding for every encoding φ , and in particular so is TwCom _{φ_K} , since partial binding is exactly binding of PedCom and thus reduces to DL.

Theorem 3 (Post-Quantum Hiding of Tweaking Switch Commitments). *Let φ be any injective encoding function, and let $\mathcal{A} = \mathcal{A}_q$ be a quantum oracle algorithm with query depth d_q and query width w_q . Then, in the quantum random oracle model for H , TwCom _{φ} satisfies*

$$\text{Adv}_{\text{TwCom}_\varphi}^{\text{hide}}(\mathcal{A}) \leq 4d_q \sqrt{\frac{w_q}{q}}.$$

⁵ For the latter reason, a similar scheme with the (non-injective) encoding $\varphi(m, r) = r$ was proposed ad-hoc in the Monero community [Par22] without a formal security argument. Since a reduction cannot recover m from the random oracle queries directly in this case, a non-AGM reduction would (necessarily, as we conjecture) incur a similar tightness loss as our reductions for the opt-in scheme.

⁶ Not any injective encoding function φ yields a secure tweaking scheme, e.g., taking $\varphi(m, r) = (m, mH + rG)$ allows an attack: $\mathcal{A}_c$ computes the tweak $h = H(m', mH + rG)$ and the commitment $c = C + hG = mH + (r + H(m', mH + rG))G$, $\mathcal{A}_q$ solves $\log_G H = x$ and computes $r' = r + (m - m')x$. Then (m', r') opens c under Verify_{full} because $m'H + (r' + H(m', m'H + r'G))G = C + hG$.

The proof is conceptually similar to that of [Theorem 2](#), i.e., it is a standard sequence of game hops that applies the O2H lemma ([Lemma 1](#)) once, and we defer it to [App. A](#).

Theorem 4 (Switch Binding of the Opt-In Schemes). *Let $K \in \{H, J\}$, and let $\mathcal{A} = (\mathcal{A}_c, \mathcal{A}_q)$ be a switch binding adversary against TwCom_{φ_K} in the quantum random oracle model for H and HashToGroup , where $\mathcal{A}_c$ runs in time t and makes d_c classical queries to H , and $\mathcal{A}_q$ has query depth d_q and query width w_q . Then there is a classical adversary $\mathcal{B}_c$ against SqCDH in $\mathbb{G}$, running in time $t' \approx t + d_c t_{\text{mul}}$, such that*

$$\text{Adv}_{\text{TwCom}_{\varphi_K}}^{\text{swbind}}(\mathcal{A}) \leq |\mathcal{M}| \cdot \text{Adv}_{\mathbb{G}}^{\text{sqcdh}}(\mathcal{B}_c) + \frac{d_c^2}{2q} + \frac{4(d_q + 1)^2 w_q}{q}.$$

Proof. Consider a run of the switch binding game for TwCom_{φ_K} . After $\mathcal{A}_c^H()$ has finished with output (c, m, p, state) , let Q_c be the set of H queries it made, each of the form $(C_1, C_2) \in \mathbb{G} \times \mathbb{G}$; since $\mathcal{A}_c$ makes at most d_c classical random-oracle queries, $|Q_c| \leq d_c$. Let E be the event that some $(C_1, C_2) \in Q_c$ and decommitment (m', r') satisfy $(C_1, C_2) = (m'H + r'G, r'K)$, $m' \neq m$, $\text{Verify}_{\text{part}}(c, m, p) = 1$ and $\text{Verify}_{\text{full}}(c, m', r') = 1$.

Case E. The reduction can neither read (m', r') off a query (C_1, C_2) nor run $\text{Verify}_{\text{full}}$ (which needs the hidden r'). It can only test *consistency* $c = C_1 + H(C_1, C_2)G$, which for the unique decommitment (μ, ρ) encoded by (C_1, C_2) with $\mu \in \mathcal{M}$ is equivalent to $\text{Verify}_{\text{full}}(c, \mu, \rho) = 1$. Therefore, each consistent query whose encoded message lies in $\mathcal{M}$ corresponds to a full decommitment of c .

Moreover, except with small probability, at most one query is consistent. Let B be the event that two distinct queries $(C_1, C_2), (C'_1, C'_2) \in Q_c$ are both consistent, and order them so that (C'_1, C'_2) is the one $\mathcal{A}_c$ made later. Consistency of both means $C_1 + H(C_1, C_2)G = c = C'_1 + H(C'_1, C'_2)G$, i.e.,

$$H(C'_1, C'_2) = H(C_1, C_2) + \log_G(C_1 - C'_1).$$

When $\mathcal{A}_c$ makes the later query, the right-hand side is already determined by the execution so far, whereas $H(C'_1, C'_2)$ is a fresh uniformly random element of $\mathbb{Z}_q$ because (C'_1, C'_2) is a new input. Hence the equation holds with probability $1/q$, and this does not depend on how $\mathcal{A}_c$ chooses c afterwards. A union bound over the at most $\binom{d_c}{2}$ pairs of queries gives

$$\Pr[B] \leq \binom{d_c}{2} \frac{1}{q} \leq \frac{d_c^2}{2q}. \quad (13)$$

The reduction $\mathcal{B}_c$ in [Fig. 8](#) takes the unique consistent query, guesses its message m' , and uses $C_2 = r'K$ to solve SqCDH . The pair $(X, X + \lambda G)$ is uniformly distributed over $\mathbb{G} \times \mathbb{G}$ because λ is uniform and independent of X , so programming H and J with them simulates $\mathcal{A}_c$'s view perfectly. For $K = J$, programming J this way is what lets a single challenge suffice: the reduction does not know $x = \log_G H$, but it does know how K relates to H . Suppose the selected query underlies the alternative decommitment and m' is its message. Consistency gives $C_1 = c - hG = mH + (p - h)G$, while the selected query opens as $C_1 = m'H + r'G$. Equating both expressions for C_1 yields $r' = (m - m')x + (p - h)$. Substituting this into $C_2 = r'K = r'(x + \lambda)G$ gives

$$C_2 = ((m - m')x + (p - h))(x + \lambda)G = (m - m')(x^2 + \lambda x)G + (p - h)K,$$

and solving for x^2G shows that the returned value $(m - m')^{-1}(C_2 - (p - h)K) - \lambda X = x^2G$ is indeed the SqCDH solution. The reduction runs $\mathcal{A}_c$ once and spends one scalar multiplication on each consistency check, plus at most six to program J , to verify $\mathcal{A}_c$'s own decommitment and to compute its output; it never calls $\mathcal{A}_q$, so it runs in time $t' \leq t + (d_c + 6)t_{\text{mul}}$. Under E , the query underlying the alternative decommitment is consistent, and under $\neg B$ it is the only consistent query, so conditioned on $E \wedge \neg B$ the reduction selects exactly this query. Its guess m' is drawn independently of the execution of $\mathcal{A}_c$ and is correct with probability $1/|\mathcal{M}|$, so $\Pr[E \wedge \neg B] \leq |\mathcal{M}| \cdot \text{Adv}_{\mathbb{G}}^{\text{sqcdh}}(\mathcal{B}_c)$. Together with (13), we obtain

$$\Pr[E] \leq \Pr[E \wedge \neg B] + \Pr[B] \leq |\mathcal{M}| \cdot \text{Adv}_{\mathbb{G}}^{\text{sqcdh}}(\mathcal{B}_c) + \frac{d_c^2}{2q}. \quad (14)$$

```

 $\mathcal{B}_c^{\mathcal{A}_c}(\mathbb{G}, G, X)$ 


---


if  $K = J$  then  $\lambda \leftarrow \mathbb{Z}_q$  else  $\lambda := 0$ 
HashToGroup("H")  $:= X$  // program  $H := X$ 
if  $K = J$  then HashToGroup("J")  $:= X + \lambda G$  // program  $J := X + \lambda G$ 
 $(c, m, p, \text{state}) := \mathcal{A}_c^H()$ 
if  $c = \perp \vee \text{Verify}_{\text{part}}(c, m, p) = 0$  then return  $\perp$ 
 $V := \{(C_1, C_2) \in Q_c : c = C_1 + H(C_1, C_2) G\}$ 
if  $|V| \neq 1$  then return  $\perp$ 
 $\{(C_1, C_2)\} := V$ 
 $m' \leftarrow \mathcal{M}$ 
if  $m' = m$  then return  $\perp$ 
 $h := H(C_1, C_2)$ 
return  $(m - m')^{-1}(C_2 - (p - h)K) - \lambda X$ 

```

Fig. 8. The reduction $\mathcal{B}_c$ to SqCDH in the proof of [Theorem 4](#), where $K = X + \lambda G$ in both cases.

Case $\neg E$. Since the game rejects $c = \perp$ and its winning condition includes $\text{Verify}_{\text{part}}(c, m, p) = 1$, a run in which $\mathcal{A}_c$ does not output a well-formed commitment that passes partial verification cannot be won. It therefore suffices to bound the winning probability over the *admissible* executions of $\mathcal{A}_c$, i.e., those that satisfy $\neg E$ and output a commitment $c \neq \perp$ together with a partial decommitment (m, p) such that $\text{Verify}_{\text{part}}(c, m, p) = 1$. We fix such an execution and the function **HashToGroup** for the remainder of this case; this fixes Q_c , the values of H on Q_c , and *state*, and leaves the values of H outside Q_c independent and uniform, so [Lemma 2](#) applies to $\mathcal{A}_q$ with auxiliary input $z := \text{state}$.

Under $\neg E$, $\mathcal{A}_c$ never queried H at a point (C_1, C_2) that is consistent with c and encodes a message $m' \neq m$, so $\mathcal{A}_q$ must find such a query after c has been fixed. Note that, by the bijection $(m', r') \mapsto (m'H + r'G, r'K)$ from $\mathbb{Z}_q^2$ to $\mathbb{G} \times \mathbb{G}$, every hash input (C_1, C_2) encodes a unique decommitment (m', r') , so the message m' is a function of (C_1, C_2) alone. We therefore apply [Lemma 2](#) with the predicate

$$\theta((C_1, C_2), H(C_1, C_2)) = 1 \iff c = C_1 + H(C_1, C_2)G \wedge m' \in \mathcal{M} \wedge m' \neq m,$$

where m' is the message encoded by (C_1, C_2) . For every $(C_1, C_2) \in Q_c$ the value $H(C_1, C_2)$ is fixed, and $\neg E$ gives $\theta = 0$. For $(C_1, C_2) \notin Q_c$, the value $H(C_1, C_2)$ is uniformly distributed over $\mathbb{Z}_q$, so $\varepsilon := \Pr[\theta = 1] \leq 1/q$. A winning output (m', r') of $\mathcal{A}_q$ determines the consistent query $(C_1, C_2) = (m'H + r'G, r'K)$ at which $\theta = 1$, so [Lemma 2](#) gives

$$\Pr\left[\text{SwBind}_{\text{TwCom}_{\varphi_K}}^{\mathcal{A}} = 1 \mid \neg E\right] \leq \frac{4(d_q + 1)^2 w_q}{q}. \quad (15)$$

Summing. Combining [\(14\)](#) and [\(15\)](#),

$$\begin{aligned} \text{Adv}_{\text{TwCom}_{\varphi_K}}^{\text{swbind}}(\mathcal{A}) &\leq \Pr[E] + \Pr\left[\text{SwBind}_{\text{TwCom}_{\varphi_K}}^{\mathcal{A}} = 1 \mid \neg E\right] \\ &\leq |\mathcal{M}| \cdot \text{Adv}_{\mathbb{G}}^{\text{sqcdh}}(\mathcal{B}_c) + \frac{d_c^2}{2q} + \frac{4(d_q + 1)^2 w_q}{q}. \quad \square \end{aligned}$$

The factor $|\mathcal{M}|$ in [Theorem 4](#) stems from the guess of m' , which the reduction $\mathcal{B}_c$ needs in order to divide the unknown factor $m - m'$ out of $C_2 - (p - h)K$ and obtain the SqCDH solution. In the AGM [\[FKL18\]](#), a reduction can do better. An algebraic algorithm $\mathcal{A}_c$ outputs, along with every group element Z , a representation of Z as a linear combination of the group elements it has received so far; as is customary when combining the AGM with the random oracle model, the group elements in its random-oracle queries count as outputs and come with representations, too. In the switch binding game, $\mathcal{A}_c$ receives only G, H and, for $K = J$, J . The representation

```

 $\mathcal{B}_c^{\mathcal{A}_c}(\mathbb{G}, G, X)$ 


---


if  $K = J$  then  $\lambda \leftarrow \$ \mathbb{Z}_q$  else  $\lambda := 0$ 
HashToGroup("H")  $:= X$     // program  $H := X$ 
if  $K = J$  then HashToGroup("J")  $:= X + \lambda G$     // program  $J := X + \lambda G$ 
 $(c, m, p, state) := \mathcal{A}_c^H()$ 
if  $c = \perp \vee \text{Verify}_{\text{part}}(c, m, p) = 0$  then return  $\perp$ 
 $V := \{(C_1, C_2) \in Q_c : c = C_1 + H(C_1, C_2)G\}$ 
if  $|V| \neq 1$  then return  $\perp$ 
 $\{(C_1, C_2)\} := V$ 
let  $(\alpha_1, \beta_1, \alpha_2, \beta_2)$  be such that  $C_1 = \alpha_1 G + \beta_1 X$  and  $C_2 = \alpha_2 G + \beta_2 X$ 
 $s := p - H(C_1, C_2)$ 
if  $\beta_1 \neq m$  then return  $(\alpha_1 - s)/(m - \beta_1)$ 
for  $\mu \in \mathcal{M} \setminus \{m\}$  do
    for each root  $x' \in \mathbb{Z}_q$  of  $P_\mu$  do
       if  $x'G = X$  then return  $x'$ 
return  $\perp$ 

```

Fig. 9. The reduction $\mathcal{B}_c$ to DL in the proof of [Theorem 5](#), where $P_\mu(T) := (m - \mu)T^2 + ((m - \mu)\lambda + s - \beta_2)T + (s\lambda - \alpha_2)$.

of C_2 turns the unknown group element into an equation over $\mathbb{Z}_q$ of degree two in $x = \log_G H$, whose only unknown coefficient is $m - m'$ and thus ranges over a set of size $|\mathcal{M}| - 1$. Trying all values of this coefficient means an *additive* increase in running time instead of a *multiplicative* loss of success probability as compared to the non-AGM reduction, and the AGM reduction solves DL instead of SqCDH. Note that the representations still do not reveal m' : the consistent query encodes a decommitment (m', r') with $m' \neq m$, but its component C_1 is necessarily represented as a decommitment to m , since any other representation yields $\log_G H$ directly. Only $\mathcal{A}_c$ is required to be algebraic; $\mathcal{A}_q$ remains an arbitrary quantum oracle algorithm.

Theorem 5 (Switch Binding of the Opt-In Schemes, AGM). *Let $K \in \{H, J\}$, and let $\mathcal{A} = (\mathcal{A}_c, \mathcal{A}_q)$ be a switch binding adversary against TwCom_{φ_K} in the quantum random oracle model for H and HashToGroup , where $\mathcal{A}_c$ is an algebraic algorithm that runs in time t and makes d_c classical queries to H , and $\mathcal{A}_q$ has query depth d_q and query width w_q . Then there is a classical adversary $\mathcal{B}_c$ against DL in $\mathbb{G}$, running in time $t' \approx t + (d_c + 2|\mathcal{M}|)t_{\text{mul}}$, such that*

$$\text{Adv}_{\text{TwCom}_{\varphi_K}}^{\text{swbind}}(\mathcal{A}) \leq \text{Adv}_{\mathbb{G}}^{\text{dl}}(\mathcal{B}_c) + \frac{d_c^2}{2q} + \frac{4(d_q + 1)^2 w_q}{q}.$$

Proof. We reuse the proof of [Theorem 4](#): the event E , the bound (13) on the event B that two queries are consistent, and the analysis of case $\neg E$ carry over unchanged, and we only replace the reduction in case E by the reduction $\mathcal{B}_c$ to DL in [Fig. 9](#). We may assume without loss of generality that $\mathcal{A}_c$ queries HashToGroup only at "H" and "J": the outputs at all other inputs are uniform and independent of H and J , so $\mathcal{A}_c$ can sample them itself as uG for $u \leftarrow \$ \mathbb{Z}_q$ of its own choosing, which leaves its view unchanged and gives it a representation of each over G . As in the proof of [Theorem 4](#), the reduction programs H with its challenge, $H := X$, and, for $K = J$, sets $J := X + \lambda G$ for $\lambda \leftarrow \$ \mathbb{Z}_q$ of its own choosing; for $K = H$ we set $\lambda := 0$, so that $K = X + \lambda G$ in both cases. The pair $(X, X + \lambda G)$ is uniformly distributed over $\mathbb{G} \times \mathbb{G}$ because λ is uniform and independent of X , so $\mathcal{B}_c$ perfectly simulates $\mathcal{A}_c$'s view. Write $H = X$ and $x = \log_G H$, so that $\log_G K = x + \lambda$. Since $\mathcal{A}_c$ is algebraic, every query (C_1, C_2) comes with a representation in terms of G, H and J , and since $\mathcal{B}_c$ knows λ , it can fold the component along J back into the first two and rewrite the representation as $C_1 = \alpha_1 G + \beta_1 H$ and $C_2 = \alpha_2 G + \beta_2 H$ with $\alpha_1, \beta_1, \alpha_2, \beta_2 \in \mathbb{Z}_q$, as used in the pseudocode. The reduction runs $\mathcal{A}_c$ once and spends one scalar multiplication on the consistency check of each of

$\mathcal{A}_c$'s at most d_c queries to H . It further tries $|\mathcal{M}| - 1$ values of μ , each of which requires computing at most two roots of a polynomial of degree two over $\mathbb{Z}_q$, i.e., one square root in $\mathbb{Z}_q$, and at most two scalar multiplications to test them; the square roots are field operations, which our cost model does not charge. Counting three more scalar multiplications to program J and to verify $\mathcal{A}_c$'s own decommitment, and noting that $\mathcal{B}_c$ never calls $\mathcal{A}_q$, it runs in time $t' \leq t + (d_c + 2|\mathcal{M}| + 1)t_{\text{mul}}$.

Suppose that $E \wedge \neg B$ holds. Then, as in the proof of [Theorem 4](#), the selected query is the one underlying the alternative decommitment (m', r') with $m' \in \mathcal{M} \setminus \{m\}$, consistency gives $C_1 = mH + sG$, and equating this with $C_1 = m'H + r'G$ yields $r' = (m - m')x + s$. We show that $\mathcal{B}_c$ returns x . If $\beta_1 \neq m$, then $C_1 = \alpha_1 G + \beta_1 H = mH + sG$ gives $\alpha_1 + \beta_1 x = s + mx$, so $x = (\alpha_1 - s)/(\beta_1 - m)$ is what $\mathcal{B}_c$ returns. Otherwise $\beta_1 = m$, and $C_2 = r'K = ((m - m')x + s)(x + \lambda)G$ together with $C_2 = \alpha_2 G + \beta_2 H$ gives

$$\alpha_2 + \beta_2 x = (m - m')x^2 + ((m - m')\lambda + s)x + s\lambda,$$

i.e., x is a root of $P_{m'}$, whose leading coefficient $m - m'$ is nonzero because $m' \neq m$ by the winning condition. In the iteration with $\mu = m'$, the reduction therefore encounters the root $x' = x$, which passes the check $x'G = X$, unless it has returned earlier; in either case it returns a value x' with $x'G = X$, i.e., $x' = x$. For $K = H$, where $\lambda = 0$, the polynomial is $P_\mu(T) = (m - \mu)T^2 + (s - \beta_2)T - \alpha_2$. Consequently,

$$\Pr[E] \leq \Pr[E \wedge \neg B] + \Pr[B] \leq \text{Adv}_{\mathbb{G}}^{\text{dl}}(\mathcal{B}_c) + \frac{d_c^2}{2q},$$

and summing up with (15) as in the proof of [Theorem 4](#) yields the claim. $\square$

Concrete Security. For Grin, whose messages are 64-bit amounts, the search over $\mathcal{M}$ that the algebraic reduction performs costs about 2^{65} scalar multiplications, i.e., roughly 2^{73} group operations, far below the roughly 2^{128} group operations needed to solve DL on `secp256k1` generically. For the attacker of [Table 1](#), where a scalar multiplication costs $t_{\text{mul}} \approx 2^8$ group operations, both reductions thus run in time about 2^{108} , which dominates t as well as the search. [Theorem 4](#) then bounds switch binding by about $|\mathcal{M}| \cdot 2^{216}/2^{257} = 2^{23}$, which is vacuous, whereas [Theorem 5](#) bounds it by about $(2^{108} + 2^{73})^2/2^{257} \approx 2^{-41}$, the entry of the table; the remaining terms are at most 2^{-57} .

6.1 Simplified Tweaking Switch Commitments

We now analyze the simplified tweaking scheme $\text{SimpCom} := \text{TwCom}_{\text{id}}$, i.e., the tweaking switch commitment scheme of [Construction 4](#) instantiated with the identity encoding $\varphi(m, r) = (m, r)$:

$$\text{SimpCom}(m, r) = mH + (r + H(m, r))G,$$

where we write $\text{SimpCom}(m, r)$ for the commitment that SwCom outputs on message m and randomness r . Its tweak is thus the hash of the decommitment itself, whereas OptCom and OptCom' hash an ElGamal commitment, i.e., use the encodings $\varphi_K = \text{EGCom}_K.\text{Com}$. The same tweaking idea appears in Bitcoin's Taproot, whose commitment is a tweaked public key rather than a tweaked Pedersen commitment [[Ruf25](#)]. Like every instance of TwCom , SimpCom is perfectly correct and partially binding, since its partial verification algorithm is exactly that of PedCom .

The two remaining security properties depend on the encoding very differently. We prove that SimpCom is switch binding under DL in the quantum random oracle model using the predicate-based search bound of [Lemma 2](#) ([Theorem 6](#)). The reduction recovers an alternative decommitment found by $\mathcal{A}_c$ directly from its random-oracle queries, which works only because the identity encoding exposes the decommitment in the clear; OptCom and OptCom' therefore require a separate proof, either under the stronger SqCDH assumption ([Theorem 4](#)) or under DL in the stronger algebraic group model ([Theorem 5](#)). In contrast, our proof of post-quantum hiding never inverts the encoding, so we state it for *every* injective encoding φ and thereby cover SimpCom and TwCom_{φ_K} at once ([Theorem 3](#)).

Theorem 6 (Switch Binding of the Simplified Tweaking Scheme). *Let $\mathcal{A} = (\mathcal{A}_c, \mathcal{A}_q)$ be a switch binding adversary against SimpCom in the quantum random oracle model for H and*

$$\mathcal{B}_c^{\mathcal{A}_c}(\mathbb{G}, G, X)$$

HashToGroup("H") := X $\parallel$ program $H := X$
 $(c, m, p, \text{state}) := \mathcal{A}_c^H()$
if $c = \perp \vee \text{Verify}_{\text{part}}(c, m, p) = 0$ **then return** $\perp$
for $(m', r') \in Q_c$ **do**
 if $m = m' \vee \text{Verify}_{\text{full}}(c, m', r') = 0$ **then continue**
 $h := H(m', r')$
 $p' := r' + h$
 $x := (p - p') / (m' - m) \parallel x = \log_G H$ as $c = mH + pG = m'H + p'G$
 return x
return $\perp$

Fig. 10. The reduction $\mathcal{B}_c$ to DL in the proof of [Theorem 6](#).

HashToGroup, where $\mathcal{A}_c$ runs in time t and makes d_c classical queries to H , and $\mathcal{A}_q$ has query depth d_q and query width w_q . Then there is a classical adversary $\mathcal{B}_c$ against DL in $\mathbb{G}$, running in time $t' \approx t + 2d_c t_{\text{mul}}$, such that

$$\text{Adv}_{\text{SimpCom}}^{\text{swbind}}(\mathcal{A}) \leq \text{Adv}_{\mathbb{G}}^{\text{dl}}(\mathcal{B}_c) + \frac{4(d_q + 1)^2 w_q}{q}.$$

The proof resembles the proof of [Theorem 1](#).

Proof of Theorem 6. Consider the game $\text{SwBind}_{\text{SimpCom}}^{\mathcal{A}}$ after the classical algorithm $\mathcal{A}_c^H()$ has finished with some output (c, m, p, state) , and let Q_c be the set of H queries performed by $\mathcal{A}_c$. Let E be the event that there is $(m', r') \in Q_c$ such that $m \neq m' \wedge \text{Verify}_{\text{part}}(c, m, p) \wedge \text{Verify}_{\text{full}}(c, m', r')$.

Case E . If E happens, then the reduction $\mathcal{B}_c^{\mathcal{A}_c}(\mathbb{G}, G, X)$ in [Fig. 10](#) will be successful in solving DL. The reduction $\mathcal{B}_c$ runs $\mathcal{A}_c$ once and performs two scalar multiplications per random-oracle query, because full verification recomputes a Pedersen commitment, and two more to verify $\mathcal{A}_c$'s own decommitment, so it runs in time $t' \leq t + 2(d_c + 1)t_{\text{mul}}$ (note that it never calls $\mathcal{A}_q$). The DL challenge X is uniformly distributed over $\mathbb{G}$, so programming H with it leaves H distributed exactly as in $\text{SwBind}_{\text{SimpCom}}^{\mathcal{A}}$ and therefore perfectly simulates $\mathcal{A}_c$'s view in the real game. Hence E implies that $\mathcal{B}_c$ solves the DL problem:

$$\Pr[E] = \Pr[\mathcal{B}_c^{\mathcal{A}_c}(\mathbb{G}, G, X) \neq \perp] \leq \text{Adv}_{\mathbb{G}}^{\text{dl}}(\mathcal{B}_c). \quad (16)$$

Case $\neg E$. Since the game rejects $c = \perp$ and its winning condition includes $\text{Verify}_{\text{part}}(c, m, p) = 1$, a run in which $\mathcal{A}_c$ does not output a well-formed commitment that passes partial verification cannot be won. It therefore suffices to bound the winning probability over the *admissible* executions of $\mathcal{A}_c$, i.e., those that satisfy $\neg E$ and output a commitment $c \neq \perp$ together with a partial decommitment (m, p) such that $\text{Verify}_{\text{part}}(c, m, p) = 1$. We fix such an execution and the function **HashToGroup** for the remainder of this case; this fixes Q_c , the values of H on Q_c , and state , and leaves the values of H outside Q_c independent and uniform, so [Lemma 2](#) applies to $\mathcal{A}_q$ with auxiliary input $z := \text{state}$.

Under $\neg E$, $\mathcal{A}_c$ did not make a query $H(m', r')$ with $m' \neq m$ that satisfies the full verification equation $\text{SimpCom}(m', r') = c$, so $\mathcal{A}_q$ has to find such (m', r') after c has already been determined.

Let θ be the boolean predicate defined as

$$\theta((m', r'), H(m', r')) = 1 \iff \text{SimpCom}(m', r') = c \wedge m' \neq m.$$

For all $(m', r') \in Q_c$ the value $H(m', r')$ is fixed, and $\neg E$ gives

$$\theta((m', r'), H(m', r')) = 0.$$

For all other queries $(m', r') \notin Q_c$, the value $H(m', r')$ is uniformly distributed over $\mathbb{Z}_q$, hence we obtain

$$\begin{aligned}
\Pr[\theta((m', r'), H(m', r')) = 1] &= \Pr[\text{SimpCom}(m', r') = c \wedge m' \neq m] \\
&\leq \Pr[\text{SimpCom}(m', r') = c] \\
&= \Pr[m'H + (r' + H(m', r'))G = c] \\
&= \Pr[H(m', r')G = c - m'H - r'G] \\
&= \Pr[H(m', r') = \log_G(c - m'H - r'G)] \\
&= 1/q.
\end{aligned}$$

The last equality holds because H is a random oracle with uniformly distributed output in $\mathbb{Z}_q$. Altogether we obtain

$$\varepsilon = \max_{(m', r') \in \mathcal{M} \times \mathcal{R}} \Pr[\theta((m', r'), H(m', r')) = 1] \leq \max\{0, 1/q\} = 1/q. \quad (17)$$

Moreover, a winning run of the game satisfies $m \neq m'$ and $\text{Verify}_{\text{full}}(c, m', r') = 1$ for the decommitment (m', r') that $\mathcal{A}_q$ returns, i.e., $\theta((m', r'), H(m', r')) = 1$. By Lemma 2, we have

$$\begin{aligned}
&\Pr[\text{SwBind}_{\text{SimpCom}}^A = 1 | \neg E] \\
&\leq \Pr[\theta((m', r'), H(m', r')) = 1 | (m', r') := \mathcal{A}_q^H(\text{state})] \\
&\leq 4(d_q + 1)^2 w_q \varepsilon.
\end{aligned} \quad (18)$$

Summing. Putting together (16)–(18), we obtain

$$\begin{aligned}
\text{Adv}_{\text{SimpCom}}^{\text{swbind}}(\mathcal{A}) &= \Pr[\text{SwBind}_{\text{SimpCom}}^A = 1 | E] \cdot \Pr[E] \\
&\quad + \Pr[\text{SwBind}_{\text{SimpCom}}^A = 1 | \neg E] \cdot \Pr[\neg E] \\
&\leq \Pr[E] + \Pr[\text{SwBind}_{\text{SimpCom}}^A = 1 | \neg E] \\
&\leq \text{Adv}_{\mathbb{G}}^{\text{dl}}(\mathcal{B}_c) + \frac{4(d_q + 1)^2 w_q}{q}. \quad \square
\end{aligned}$$

Acknowledgements. We thank Jonas Nick, Oleksandr Kurbatov, and Dmytro Zakharov for their helpful review and comments.

Claude AI models (Fable 5.1, Opus 5, and Sonnet 5) were used heavily in non-technical and technical sections to proofread the manuscript, verify correctness of the claims and proofs, fix factual and grammatical issues, generate paragraphs, rephrase text and sharpen writing, and conduct mechanical editing work. Notably, when Fable 5.1 was asked to check the correctness of an earlier handwritten proof of Theorem 4 with a looser security bound, it pointed out that the bound can be tightened (“event B ”), and we accepted its suggestion. Moreover, the initial proof of Theorem 5 (in the AGM) was generated by Fable 5.1; it was given the structurally similar non-AGM proof (Theorem 4) as a prompt, plus human guidance on how the arguments will differ. All AI-produced content including proofs was manually verified by the authors.

References

- [ABN10] Michel Abdalla, Mihir Bellare, and Gregory Neven. *Robust Encryption*. In: TCC 2010. DOI: [10.1007/978-3-642-11799-2_28](https://doi.org/10.1007/978-3-642-11799-2_28).
- [AHU18] Andris Ambainis, Mike Hamburg, and Dominique Unruh. *Quantum security proofs using semi-classical oracles*. 2018. IACR: [2018/904](https://eprint.iacr.org/2018/904).
- [And17] Oleg Andreev. *Confidential Assets Specification*. 2017. URL: <https://github.com/chain/chain/blob/master/confidential-spec/docs/protocol/specifications/ca.md#value-range-proof>.
- [ARU14] Andris Ambainis, Ansis Rosmanis, and Dominique Unruh. *Quantum Attacks on Classical Proof Systems: The Hardness of Quantum Rewinding*. In: 55TH FOCS. DOI: [10.1109/FOCS.2014.57](https://doi.org/10.1109/FOCS.2014.57).

- [Bab+26] Ryan Babbush, Adam Zalcman, Craig Gidney, Michael Broughton, Tanuj Khattar, Hartmut Neven, Thiago Bergamaschi, Justin Drake, and Dan Boneh. *Securing Elliptic Curve Cryptocurrencies against Quantum Vulnerabilities: Resource Estimates and Mitigations*. In: PRX QUANTUM 7.3 (2026). DOI: [10.1103/j3xf-bw18](https://doi.org/10.1103/j3xf-bw18).
- [Bac13] Adam Back. *Bitcoins with homomorphic value*. Bitcointalk Forum Post. Accessed: 2026-01-23. 2013. URL: <https://bitcointalk.org/index.php?topic=305791.0>.
- [Ben+14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. *ZeroCash: Decentralized Anonymous Payments from Bitcoin*. In: 2014 IEEE SYMPOSIUM ON SECURITY AND PRIVACY. DOI: [10.1109/SP.2014.36](https://doi.org/10.1109/SP.2014.36).
- [BH22] Mihir Bellare and Viet Tung Hoang. *Efficient Schemes for Committing Authenticated Encryption*. In: EUROCRYPT 2022. DOI: [10.1007/978-3-031-07085-3_29](https://doi.org/10.1007/978-3-031-07085-3_29).
- [Blo26a] Blockchain.com. *Total Hash Rate (TH/s)*. 2026. URL: <https://www.blockchain.com/explorer/charts/hash-rate> (visited on 09/14/2026).
- [Blo26b] Blockstream Research. *secp256k1-zkp: a fork of libsecp256k1 with support for advanced and experimental features such as Confidential Assets and MuSig2*. GitHub repository. Accessed: 2026-07-29. 2026. URL: <https://github.com/BlockstreamResearch/secp256k1-zkp>.
- [Bon+11] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. *Random Oracles in a Quantum World*. In: ASIACRYPT 2011. DOI: [10.1007/978-3-642-25385-0_3](https://doi.org/10.1007/978-3-642-25385-0_3).
- [Bün+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. *Bulletproofs: Short Proofs for Confidential Transactions and More*. In: 2018 IEEE SYMPOSIUM ON SECURITY AND PRIVACY. DOI: [10.1109/SP.2018.00020](https://doi.org/10.1109/SP.2018.00020).
- [CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. *Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols*. In: CRYPTO'94. DOI: [10.1007/3-540-48658-5_19](https://doi.org/10.1007/3-540-48658-5_19).
- [CFS26] Clémence Chevalignard, Pierre-Alain Fouque, and André Schrottenloher. *Reducing the Number of Qubits in Quantum Discrete Logarithms on Elliptic Curves*. In: EUROCRYPT 2026. DOI: [10.1007/978-3-032-25291-3_13](https://doi.org/10.1007/978-3-032-25291-3_13).
- [CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. *A Secure and Optimally Efficient Multi-Authority Election Scheme*. In: EUROCRYPT'97. DOI: [10.1007/3-540-69053-0_9](https://doi.org/10.1007/3-540-69053-0_9).
- [Chu+20] Heewon Chung, Kyoohyung Han, Chanyang Ju, Myungsun Kim, and Jae Hong Seo. *Bulletproofs+: Shorter Proofs for Privacy-Enhanced Distributed Ledger*. 2020. IACR: [2020/735](https://arxiv.org/abs/2020.735).
- [CP93] David Chaum and Torben P. Pedersen. *Wallet Databases with Observers*. In: CRYPTO'92. DOI: [10.1007/3-540-48071-4_7](https://doi.org/10.1007/3-540-48071-4_7).
- [CR22] John Chan and Phillip Rogaway. *On Committing Authenticated-Encryption*. In: ESORICS 2022. DOI: [10.1007/978-3-031-17146-8_14](https://doi.org/10.1007/978-3-031-17146-8_14).
- [Cru25] Agustin Cruz. *Proposal for Quantum-Resistant Address Migration Protocol (QRAMP) BIP*. Posting to Bitcoin Development Mailing List (bitcoin-dev). 2025. URL: <https://groups.google.com/g/bitcoinddev/c/8PM6iZCeDMc>.
- [Ele26] ElementsProject Contributors. *Elements: an open source, sidechain-capable blockchain platform*. GitHub repository. Accessed: 2026-04-02. 2026. URL: <https://github.com/ElementsProject/elements>.
- [Esg+19] Muhammed F. Esgin, Raymond K. Zhao, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. *MatRiCT: Efficient, Scalable and Post-Quantum Blockchain Confidential Transactions Protocol*. In: ACM CCS 2019. DOI: [10.1145/3319535.3354200](https://doi.org/10.1145/3319535.3354200).
- [ESZ22] Muhammed F. Esgin, Ron Steinfeld, and Raymond K. Zhao. *MatRiCT+: More Efficient Post-Quantum Private Blockchain Payments*. In: 2022 IEEE SYMPOSIUM ON SECURITY AND PRIVACY. DOI: [10.1109/SP46214.2022.9833655](https://doi.org/10.1109/SP46214.2022.9833655).
- [FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. *The Algebraic Group Model and its Applications*. In: CRYPTO 2018. DOI: [10.1007/978-3-319-96881-0_2](https://doi.org/10.1007/978-3-319-96881-0_2).
- [FOS19] Georg Fuchsbauer, Michele Orrù, and Yannick Seurin. *Aggregate Cash Systems: A Cryptographic Investigation of Mimblewimble*. In: EUROCRYPT 2019. DOI: [10.1007/978-3-030-17653-2_22](https://doi.org/10.1007/978-3-030-17653-2_22).

- [FS90] Uriel Feige and Adi Shamir. *Witness Indistinguishable and Witness Hiding Protocols*. In: 22ND ACM STOC. DOI: [10.1145/100216.100272](https://doi.org/10.1145/100216.100272).
- [Gao+25] Shang Gao, Tianyu Zheng, Yu Guo, Zhe Peng, and Bin Xiao. *Lattice-Based Zero-Knowledge Proofs for Blockchain Confidential Transactions*. In: PKC 2025. DOI: [10.1007/978-3-031-91832-2_5](https://doi.org/10.1007/978-3-031-91832-2_5).
- [GLR17] Paul Grubbs, Jiahui Lu, and Thomas Ristenpart. *Message Franking via Committing Authenticated Encryption*. In: CRYPTO 2017. DOI: [10.1007/978-3-319-63697-9_3](https://doi.org/10.1007/978-3-319-63697-9_3).
- [Gri19a] Grin Developers. *Introduction to Switch Commitments*. Grin Documentation. Accessed: 2026-08-29. 2019. URL: https://github.com/mimblewimble/grin/blob/master/doc/switch_commitment.md.
- [Gri19b] Grin Developers. *Range Proof Builder*. Grin Source Code. Accessed: 2026-09-16. 2019. URL: <https://github.com/mimblewimble/grin/blob/master/core/src/libtx/proof.rs>.
- [Gri20] Grin Developers. *Range Proof Format*. Grin Documentation. Accessed: 2026-06-17. 2020. URL: <https://quentinlesceller.github.io/grin-docs/technical/miscellaneous/range-proof-format/>.
- [Jed16] Tom Elvis Jedusor. *Mimblewimble*. Posted to a Bitcoin research IRC channel. Accessed: 2026-08-07. 2016. URL: <https://scalingbitcoin.org/papers/mimblewimble.txt>.
- [KAN20] Koe, Kurt M. Alonso, and Sarang Noether. *Zero to Monero: Second Edition*. Monero Research Lab. 2020. URL: <https://www.getmonero.org/library/Zero-to-Monero-2-0-0.pdf>.
- [LNS21] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. *SMILE: Set Membership from Ideal Lattices with Applications to Ring Signatures and Confidential Transactions*. In: CRYPTO 2021. DOI: [10.1007/978-3-030-84245-1_21](https://doi.org/10.1007/978-3-030-84245-1_21).
- [Lop25] Jameson Lopp. *Against Allowing Quantum Recovery of Bitcoin*. 2025. URL: <https://blog.lopp.net/against-quantum-recovery-of-bitcoin/>.
- [Max15] Gregory Maxwell. *Confidential Transactions*. 2015. URL: https://web.archive.org/web/20150628230410/https://people.xiph.org/~greg/confidential_values.txt.
- [Mon26] Monero Project. *RingCT Operations*. Monero Source Code. Accessed: 2026-09-16. See `ecdhEncode` and the two functions it calls to derive the commitment mask and the amount pad from the shared secret. 2026. URL: <https://github.com/monero-project/monero/blob/master/src/ringct/rctOps.cpp>.
- [MP15] Gregory Maxwell and Andrew Poelstra. *Borromean Ring Signatures*. Technical draft. Version 0.01, commit 34241bb. 2015. URL: https://raw.githubusercontent.com/Blockchain/borromean_paper/master/borromean_draft_0.01_34241bb.pdf.
- [Nak08] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008. URL: <https://bitcoin.org/bitcoin.pdf>.
- [Nat16] National Institute of Standards and Technology. *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. 2016. URL: <https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf>.
- [NMM16] Shen Noether, Adam Mackenzie, and Monero Research Lab. *Ring Confidential Transactions*. Monero Research Lab, MRL-0005. 2016. URL: <https://www.getmonero.org/resources/research-lab/pubs/MRL-0005.pdf>.
- [Par22] Luke Parker. *Comment on Monero Research Lab issue 105, “Consider Switch commitments for future supply security”*. GitHub issue comment. Accessed: 2026-09-18. See the paragraph starting with “Side note”. 2022. URL: <https://github.com/monero-project/research-lab/issues/105#issuecomment-1233370551>.
- [Ped92] Torben P. Pedersen. *Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing*. In: CRYPTO’91. DOI: [10.1007/3-540-46766-1_9](https://doi.org/10.1007/3-540-46766-1_9).
- [Poe+19] Andrew Poelstra, Adam Back, Mark Friedenbach, Gregory Maxwell, and Pieter Wuille. *Confidential Assets*. In: FC 2018 WORKSHOPS. DOI: [10.1007/978-3-662-58820-8_4](https://doi.org/10.1007/978-3-662-58820-8_4).
- [RM17] Tim Ruffing and Giulio Malavolta. *Switch Commitments: A Safety Switch for Confidential Transactions*. In: FC 2017 WORKSHOPS. DOI: [10.1007/978-3-319-70278-0_10](https://doi.org/10.1007/978-3-319-70278-0_10).

- [Ruf18] Tim Ruffing. *switch commitments (again)*. Mumblewimble mailing list. Accessed: 2026-09-18. 2018. URL: <https://archive.lists.launchpad.net/mumblewimble/msg00479.html>.
- [Ruf20] Tim Ruffing. *Cryptography for Bitcoin and friends*. PhD thesis. Saarland University, Germany, 2020. URL: <https://d-nb.info/1209947285>.
- [Ruf25] Tim Ruffing. *The Post-Quantum Security of Bitcoin's Taproot as a Commitment Scheme*. 2025. IACR: 2025/1307.
- [Sho97] Victor Shoup. *Lower Bounds for Discrete Logarithms and Related Problems*. In: EU-ROCRYPT'97. DOI: 10.1007/3-540-69053-0_18.
- [sow24] sowle. *Zano: Confidential Assets Scheme for RingCT and Zarcantum*. Zano project technical report. Version 1.1, last update 2024-02-05. 2024. URL: https://hyle-team.github.io/docs/zano/CA_paper/Zano_CA_for_RingCT_and_Zarcantum_v1.1.pdf.
- [Tor+18] Wilson Abel Alberto Torres, Ron Steinfeld, Amin Sakzad, Joseph K. Liu, Veronika Kuchta, Nandita Bhattacharjee, Man Ho Au, and Jacob Cheng. *Post-Quantum One-Time Linkable Ring Signature and Application to Ring Confidential Transactions in Blockchain (Lattice RingCT v1.0)*. In: ACISP 18. DOI: 10.1007/978-3-319-93638-3_32.
- [Tor+19] Wilson Abel Alberto Torres, Veronika Kuchta, Ron Steinfeld, Amin Sakzad, Joseph K. Liu, and Jacob Cheng. *Lattice RingCT V2.0 with Multiple Input and Multiple Output Wallets*. In: ACISP 19. DOI: 10.1007/978-3-030-21548-4_9.
- [Zan26] Zano Project. *Currency Format Utilities*. Zano Source Code. Accessed: 2026-09-16. See the construction of a `tx_out_zarcantum` output, which derives the amount blinding mask and the amount pad from the shared secret. 2026. URL: https://github.com/hyle-team/zano/blob/master/src/currency_core/currency_format_utils.cpp.

A Proof of Theorem 3

The proof mirrors the proof of Theorem 2 and follows a similar One-Way to Hiding argument (Lemma 1) as the hiding proof of Ruffing [Ruf25, Theorem 2].

Proof of Theorem 3. The proof will proceed by a sequence of hops as defined in Fig. 11. For each hop, we will show that two games have identical (=) or similar ($\approx$) output distribution.

We will first show that for all $b \in \{0, 1\}$, we have

$$\text{Hide}_{\text{TwCom}_\varphi}^A(m_b) = \text{Game}_1^A(m_b) \approx \text{Game}_2^A(m_b) = \text{Game}_3^A(m_b)$$

and then we will show

$$\text{Game}_3^A(m_0) = \text{Game}_3^A(m_1).$$

This will enable us to bound the difference between the games $\text{Hide}_{\text{TwCom}_\varphi}^A(m_0)$ and $\text{Hide}_{\text{TwCom}_\varphi}^A(m_1)$ via

$$\begin{aligned} \text{Hide}_{\text{TwCom}_\varphi}^A(m_0) &= \text{Game}_1^A(m_0) \approx \text{Game}_2^A(m_0) = \text{Game}_3^A(m_0) \\ &= \text{Game}_3^A(m_1) = \text{Game}_2^A(m_1) \approx \text{Game}_1^A(m_1) = \text{Hide}_{\text{TwCom}_\varphi}^A(m_1). \end{aligned} \quad (19)$$

Fix a quantum algorithm $\mathcal{A}$ with query depth d_q and query width w_q and fix a message pair (m_0, m_1) . Also fix a $b \in \{0, 1\}$.

Hop $\text{Hide}_{\text{TwCom}_\varphi}^A \rightarrow \text{Game}_1$. $\text{Hide}_{\text{TwCom}_\varphi}^A(m_b)$ is the hiding game of Definition 4 for TwCom_φ with the commitment algorithm of Construction 4 inlined and the random oracle H explicit. The code of $\text{Game}_1^A(m_b)$ is the result of adding the assignments of variables h' and G to $\text{Hide}_{\text{TwCom}_\varphi}^A$. Since these additional assignments are dead stores (i.e., the assigned variables are never read in the subsequent code), this change does not influence the return value of the game, and thus

$$\left| \Pr[\text{Hide}_{\text{TwCom}_\varphi}^A(m_b) = 1] - \Pr[\text{Game}_1^A(m_b) = 1] \right| = 0. \quad (20)$$

$\text{Hide}_{\text{TwCom}_\varphi}^A(m_b)$	$\text{Game}_1^A(m_b)$
$H \leftarrow \$ \mathbb{Z}_q^{\mathcal{X}}$	$H \leftarrow \$ \mathbb{Z}_q^{\mathcal{X}}$
$r \leftarrow \$ \mathbb{Z}_q$	$r \leftarrow \$ \mathbb{Z}_q$
$C := m_b H + rG$	$C := m_b H + rG$
$h := H(\varphi(m_b, r))$	$h := H(\varphi(m_b, r))$
$c := C + hG$	$h' \leftarrow \$ \mathbb{Z}_q$
return $\mathcal{A}^H(c)$	$G := H[\varphi(m_b, r) := h']$
	$c := C + hG$
	return $\mathcal{A}^H(c)$
$\text{Game}_2^A(m_b)$	$\text{Game}_3^A(m_b)$
$H \leftarrow \$ \mathbb{Z}_q^{\mathcal{X}}$	$H \leftarrow \$ \mathbb{Z}_q^{\mathcal{X}}$
$r \leftarrow \$ \mathbb{Z}_q$	$r \leftarrow \$ \mathbb{Z}_q$
$C := m_b H + rG$	$G \leftarrow \$ \mathbb{Z}_q^{\mathcal{X}}$
$h := H(\varphi(m_b, r))$	$C := m_b H + rG$
$h' \leftarrow \$ \mathbb{Z}_q$	$h \leftarrow \$ \mathbb{Z}_q$
$G := H[\varphi(m_b, r) := h']$	$c := C + hG$
$c := C + hG$	return $\mathcal{A}^G(c)$
return $\mathcal{A}^G(c)$	

Fig. 11. Sequence of games for the proof of [Theorem 3](#). Boxed lines indicate changes from the preceding game. The commitment algorithm of [Construction 4](#) is inlined in the games.

Hop $\text{Game}_1 \rightarrow \text{Game}_2$. The code of $\text{Game}_2^A(m_b)$ is identical to that of $\text{Game}_1^A(m_b)$ except that $\mathcal{A}$ is given oracle access to the function $G := H[\varphi(m_b, r) := h']$ instead of H , where $h' \leftarrow \$ \mathbb{Z}_q$ is freshly sampled from the codomain of H . In other words, G is a random function that is identical to H except at position $\varphi(m_b, r)$, at which its value has been resampled.

In order to bound the probability that $\mathcal{A}$ distinguishes H and G , we apply [Lemma 1](#) instantiated with $S = \{\varphi(m_b, r)\}$ and $z = c$ to $\mathcal{A}$. Here, the values C, c, r, m_b, H, G have the same distribution as the values of the corresponding variables in $\text{Game}_1^A(m_b)$ and $\text{Game}_2^A(m_b)$, and the code assigning these variables is identical in the two games. This yields

$$\begin{aligned}
& \left| \Pr[\text{Game}_1^A(m_b) = 1] - \Pr[\text{Game}_2^A(m_b) = 1] \right| \\
&= \left| \Pr[\mathcal{A}^H(c) = 1 \text{ in } \text{Game}_1^A] - \Pr[\mathcal{A}^G(c) = 1 \text{ in } \text{Game}_2^A] \right| \\
&\leq 2d_q \sqrt{\Pr[\mathcal{E}^G(c) \cap S \neq \emptyset \text{ in } \text{Game}_2^{\mathcal{E}}(m_b)]}.
\end{aligned} \tag{21}$$

As will be shown below, $\text{Game}_2(m_b)$ and $\text{Game}_3(m_b)$ induce the same joint distribution of all their variables, whichever algorithm they run, in particular of the randomness r and that algorithm's output. Applying this to $\mathcal{E}$, we have

$$\Pr[\mathcal{E}^G(c) \cap S \neq \emptyset \text{ in } \text{Game}_2^{\mathcal{E}}(m_b)] = \Pr[\mathcal{E}^G(c) \cap S \neq \emptyset \text{ in } \text{Game}_3^{\mathcal{E}}(m_b)]. \tag{22}$$

Hop $\text{Game}_2 \rightarrow \text{Game}_3$. In Game_2^A , the random value $H(\varphi(m_b, r))$ is used only in the assignment to h . (Recall that $\mathcal{A}^G$ is not given H but only G , which is identical to H except at position $\varphi(m_b, r)$, where the value has been resampled.) Thus $H(\varphi(m_b, r))$ is distributed independently of r and of G , i.e., of everything in the game except the values computed from it, and can be replaced by a freshly sampled $h \leftarrow \$ \mathbb{Z}_q$ without changing the joint distribution. Similarly, G can be replaced by a freshly sampled random oracle without changing its distribution. As a result, the two games induce the same joint distribution, and since these are the only differences between $\text{Game}_2^A(m_b)$

and $\text{Game}_3^A(m_b)$, we have

$$\left| \Pr[\text{Game}_2^A(m_b) = 1] - \Pr[\text{Game}_3^A(m_b) = 1] \right| = 0. \quad (23)$$

In $\text{Game}_3^\mathcal{E}(m_b)$, the inputs to $\mathcal{E}^G(c)$, namely $G \leftarrow \mathbb{Z}_q^{\mathcal{X}}$ and $c = C + hG = m_b H + (h + r)G$ with $h \leftarrow \mathbb{Z}_q$, are distributed independently of r , since the fresh mask h acts as a one-time pad that makes c uniform over $\mathbb{G}$, hiding $C = m_b H + rG$ entirely. Since m_b is fixed, they are in particular distributed independently of $S = \{\varphi(m_b, r)\}$. So $\mathcal{E}$ hits the only element of S only if one of the at most w_q points it outputs equals $\varphi(m_b, r)$. Consider one such point x , and recall that the inputs of $\mathcal{E}$, and hence x itself, are independent of r . Since φ is injective, at most one $r \in \mathbb{Z}_q$ satisfies $\varphi(m_b, r) = x$, and r is uniform over $\mathbb{Z}_q$, so $x = \varphi(m_b, r)$ holds with probability at most $1/q$. A union bound over the at most w_q outputs gives

$$\Pr[\mathcal{E}^G(c) \cap S \neq \emptyset \text{ in } \text{Game}_3^\mathcal{E}(m_b)] \leq \frac{w_q}{q}. \quad (24)$$

By combining (21), (22) and (24), we get

$$\left| \Pr[\text{Game}_1^A(m_b) = 1] - \Pr[\text{Game}_2^A(m_b) = 1] \right| \leq 2d_q \sqrt{\frac{w_q}{q}}. \quad (25)$$

Identity of $\text{Game}_3(m_0)$ and $\text{Game}_3(m_1)$. In $\text{Game}_3^A(m_b)$, the variable h is freshly sampled uniformly from $\mathbb{Z}_q$, independently of b ; hence $c := C + hG$ acts as a one-time pad encryption of C and is independent of m_b . Thus the output distribution is independent of m_b , so we have:

$$\left| \Pr[\text{Game}_3^A(m_0) = 1] - \Pr[\text{Game}_3^A(m_1) = 1] \right| = 0. \quad (26)$$

Summing. We combine the bounds (20), (23) and (25), and (26), following the sequence of hops laid out in (19). This yields

$$\begin{aligned} & \left| \Pr[\text{Hide}_{\text{TwCom}_\varphi}^A(m_0) = 1] - \Pr[\text{Hide}_{\text{TwCom}_\varphi}^A(m_1) = 1] \right| \\ & \leq \left| \Pr[\text{Hide}_{\text{TwCom}_\varphi}^A(m_0) = 1] - \Pr[\text{Game}_3^A(m_0) = 1] \right| \\ & \quad + \left| \Pr[\text{Game}_3^A(m_0) = 1] - \Pr[\text{Game}_3^A(m_1) = 1] \right| \\ & \quad + \left| \Pr[\text{Game}_3^A(m_1) = 1] - \Pr[\text{Hide}_{\text{TwCom}_\varphi}^A(m_1) = 1] \right| \\ & \leq 4d_q \sqrt{\frac{w_q}{q}}. \end{aligned}$$

The message pair was arbitrary, so the same bound holds for $\text{Adv}_{\text{TwCom}_\varphi}^{\text{hide}}(\mathcal{A})$. $\square$

B Extending the Coincidental Scheme to Assets

CoCom (Sect. 5) binds only the amount, but on a multi-asset ledger, e.g., Liquid, the asset type is hidden as well, and a quantum attack can target it too. In Confidential Assets [Poe+19], each asset has a public generator $H_A = \text{HashToGroup}(A)$, where HashToGroup maps an asset identifier to a group element with unknown discrete logarithm relation to G , and an output commits to it under a blinding factor s , so it carries an asset commitment $C_A = H_A + sG$ alongside its value commitment $C_m = mC_A + pG$. This commitment has the same Pedersen structure as the value commitment, so an adversary who solves DL can transmute one asset into another exactly as it can forge an amount. Knowing the scalar difference x between two asset generators, $H_A - H'_A = xG$, it reopens C_A to a target asset H'_A under the forged blinder $s' = s + x$, and the pair (H'_A, s') passes one asset off as another without detection.

We sketch how the coincidental construction extends to the asset commitment, keeping its advantages and drawbacks. We develop it for Liquid, where the Back-Maxwell range proof already embeds the asset data, so the same proof that binds the amount also binds the asset type.

We denote the algorithms of the asset switch commitment schemes analogously to those of the value switch commitment schemes, but with subscript A . For instance, $\text{SwCom}_A(A, s)$ denotes the asset commitment algorithm that takes an asset identifier A and an asset blinding factor s . Similarly, $\text{Verify}_{\text{part}, A}$ and $\text{Verify}_{\text{full}, A}$ denote partial and full verification for asset commitments.

The currently implemented scheme extends to asset commitments by the same mechanism that provides switch binding for values: the range proof π already serves as a commitment to the asset data. Recall from [Remark 1](#) that the deployed RPProve algorithm additionally takes the asset identifier A and the asset blinding factor s as inputs, embedding them steganographically into the Borromean signature scalars of the first ring:

$$s_{0,0} := s_{0,0} \oplus A, \quad s_{0,1} := s_{0,1} \oplus s.$$

Correspondingly, RPWitRec recovers the full tuple (m, p, A, s) from the range proof given the correct seed η . The asset commitment algorithm $\text{SwCom}_A(A, s)$ computes $H_A = \text{HashToGroup}(A)$ and outputs the pair (C_A, π) , where $C_A = H_A + sG$ is the blinded asset generator and π is the range proof accompanying the value commitment. Note that π is produced by the value commitment algorithm SwCom , which requires C_A as input. In practice, the committer first computes C_A , then runs SwCom (which internally invokes RPProve with the asset data A, s embedded), and finally packages (C_A, π) as the asset switch commitment. The verification algorithms operate as follows:

- $\text{Verify}_{\text{part}, A}((C_A, \pi), A, s)$ opens C_A as a standard Pedersen commitment: the verifier computes $H_A = \text{HashToGroup}(A)$ and checks that $C_A = H_A + sG$.
- $\text{Verify}_{\text{full}, A}((C_A, \pi), A, (s, \eta))$ first runs $\text{Verify}_{\text{part}, A}((C_A, \pi), A, s)$ and rejects if it fails, then recovers the value commitment $C = \sum_i C_i$ from π ([Definition 12](#)), recovers (m', p', A', s') from π using C and η , and accepts iff $A = A'$ and $s = s'$.

Since $\text{Verify}_{\text{part}, A}$ checks $C_A = \text{HashToGroup}(A) + sG$, matching the extracted (A', s') against the claimed decommitment (A, s) guarantees that (A', s') itself opens C_A , exactly the two conditions the switch binding argument below relies on.

The switch binding argument mirrors the value case ([Theorem 1](#)) but now applies to the asset data encoded in the first ring. As in the value proof, the analysis splits into two cases. If the classical adversary $\mathcal{A}_c$ queries K on a seed η that yields a valid alternative asset decommitment, one can extract a discrete logarithm relation between the two asset generators, which breaks DL. The interesting case is $\neg E$: the quantum adversary $\mathcal{A}_q$ must find a seed η such that RPWitRec extracts a tuple (m', p', A', s') satisfying two *simultaneous* conditions:

1. $C_A = \text{HashToGroup}(A') + s'G$, i.e., (A', s') is a valid Pedersen decommitment of the asset commitment, and
2. witness recovery with the key material $K(\eta)$ returns exactly (A', s') from the steganographic encoding.

For a fresh (unexplored) seed η , the one-time-pad masks $otp_{0,0}, otp_{0,1}$ output by $K(\eta)$ are uniformly distributed, so the extracted asset identifier $A' := s_{0,0} \oplus otp_{0,0}$ and blinding factor $s' := s_{0,1} \oplus otp_{0,1}$ are uniform and independent; in the branches where witness recovery instead subtracts a challenge term $e x_0$ ([Fig. 5](#)), the recovered value is shifted by a term that the uniform and independent blinder x_0 makes uniform as well. The probability that the extracted s' simultaneously equals $\log_G(C_A - \text{HashToGroup}(A'))$ for the extracted A' is therefore $1/q$. Since HashToGroup is independent of K and can be fixed first, this holds for every η and every value of HashToGroup , and applying [Lemma 2](#) with $\varepsilon = 1/q$ yields the same bound as in the value case ([5](#)).