The Post-Quantum Security of Bitcoin's Taproot as a Commitment Scheme

Tim Ruffing

Blockstream Research
me@real-or-random.org

July 17, 2025

Abstract. As of November 2021, Bitcoin supports "Taproot" spending policies whose on-chain format is a single elliptic curve point. A transaction spending the funds associated with a Taproot policy can be authorized by interpreting the curve point either (a) as a public key of the Schnorr signature scheme and providing a suitable signature, or (b) as a commitment to alternative spending conditions and satisfying those.
Since a sufficiently powerful quantum adversary would be able to forge Schnorr signatures, an upgrade to Bitcoin may, at some point in the future, disable the ability to spend existing funds via Schnorr signatures in order to prevent the havoc created by leaving a large fraction of the currency supply prone to theft. However, to avoid irrevocably losing all funds not migrated in time to (yet to be added) post-quantum signature schemes, it will be desirable for an upgrade disabling Schnorr signatures to retain the ability to spend funds by interpreting the curve point in a Taproot policy as a commitment to alternative spending conditions.
This paper justifies such an upgrade strategy by demonstrating the post-quantum security of Taproot as a commitment scheme. Specifically, it provides concrete upper bounds on the probability that a quantum adversary making some number of queries to a quantum random oracle can break the binding or hiding property. Since the bounds follow from powerful existing results, which enable reasoning as if dealing with a classical adversary, the proofs are accessible without a background in quantum computing.

Table of Contents

1 Introduction

As of the Taproot upgrade in November 2021 [BIP340; BIP341; BIP342], Bitcoin supports a new type of spending policy with a compact on-chain representation. Concretely, let G be the generator of the additively written elliptic curve $\mathbb{G} = \texttt{secp256k1}$ [SEC2] as traditionally used in Bitcoin, and let H be a specific cryptographic hash function based on `SHA256`. A *Taproot spending policy* is represented by a single group element $C = rG + \mathsf{H}(rG, m)G$, where r is a random scalar and m describes a set of *scripts* (i.e., small programs) each specifying a possibly complex spending policy.

The owner of some funds stored under a Taproot policy C can spend them either (a) via a *key-path spend*, i.e., by interpreting C as a public key with secret key $r + \mathsf{H}(rG, m)$ and providing a Schnorr signature [Sch91] valid under C, or (b) via a *script-path spend*, i.e., by interpreting C as a commitment to m by revealing m and rG, and providing whatever authorization data is necessary to satisfy one of the scripts described by m. Key-path spends are preferable over script-path spends whenever possible because they offer efficiency and privacy benefits: the authorization data that needs to be stored on the blockchain is just a single Schnorr signature, and nothing is revealed about the set of scripts in m.

Since Schnorr signatures, as well as the legacy ECDSA still supported in non-Taproot policies, inherently rely on the discrete logarithm (DL) problem, which can be solved in polynomial time on a quantum computer [Sho94],[1] the Bitcoin community is currently debating potential future upgrades to Bitcoin that add support for post-quantum secure signature schemes [e.g., Bea24; Cor24; Hei25; Dry25; Lop25a].In an ideal scenario, such an upgrade will happen long before the availability of a sufficiently large quantum computer so that owners of funds currently secured by DL-based signatures will have enough time to move their funds to post-quantum secure signature schemes. Later, but before the quantum threat becomes imminent, another "kill switch" upgrade will be necessary that disables spending via DL-based signatures, effectively letting the non-upgraded funds expire to avoid the havoc resulting from leaving them vulnerable to theft.[2]

For funds stored under a Taproot policy, disabling Schnorr signatures implies disabling key-path spends. In that situation, script-path spends could come to the rescue. If $C = rG + \mathsf{H}(rG, m)G$, when interpreted as a commitment, is binding to m against quantum adversaries, then script-path spends will remain secure in a post-quantum world, and they could allow the recovery of funds that would otherwise be lost irrevocably.

This observation suggests an upgrade path in which the first upgrade, i.e., the one that adds support for post-quantum secure signatures, does so by introducing appropriate instructions to the script language [Cor24]. After this first upgrade has been completed, newly created Taproot policies would still primarily use Schnorr signatures but could additionally contain recovery scripts that allow spending securely via post-quantum signatures, keeping the funds accessible even after the deactivation of Schnorr signatures.

The advantage of such an upgrade path is that the efficiency and privacy benefits of Taproot will be retained until the kill switch upgrade. This will provide more time for research and development, allowing for further upgrades that can improve and refine Bitcoin's post-quantum cryptography before the kill switch upgrade makes its usage mandatory. Relevant research areas are not limited to efficiency and flexibility improvements of ordinary post-quantum signatures [e.g. Alk+20; DM20; ASY22; BTT22; Das+24; Zha+24; Aar+24; RK24; Dra+25; BEL25] but also include alternative ways of authorizing the spending of funds such as post-quantum zero-knowledge proofs [e.g., Cha+17; Ben+18; KKW18] and commit-delay-reveal protocols [And+98; BM14; Ruf18; Ste+18; IKS19; SW23; Dry25], both of which might make it possible to recover even funds not moved to policies with post-quantum recovery scripts in time by exploiting that secret keys are usually derived deterministically from a seed [BIP32].

The security of such an upgrade path, which eventually disables key-path spends but leaves script-path spends enabled, relies on the post-quantum security of the commitment scheme implicit

[1] Other parts of the Bitcoin protocol that use cryptography and may be affected by quantum adversaries include mining [Sat20; LRS19; Coj+19; BS24] and transport encryption on the P2P network [BIP324], but these are out of the scope of this work.

[2] Some believe that it will be preferable to leave the non-upgraded funds vulnerable to theft [e.g., Bea25]. See Lopp [Lop25b] for a summary of the debate, including convincing (as I believe) arguments for letting the funds expire.

in Taproot. However, while there has been work towards a formal security analysis of Taproot in the pre-quantum setting [Poe20; Fou20a; Fou20b], there have been no positive results in the post-quantum setting thus far.

The purpose of this paper is to justify the retention of script-path spends by quantifying the post-quantum security of Taproot as a binding and hiding commitment scheme. The security analysis relies purely on query complexity in the quantum random oracle model, i.e., the cost of an attack is measured only in the number of evaluations of a cryptographic hash function, otherwise ignoring running time and space.

Concretely, consider a quantum adversary that performs at most d query rounds, where each query round provides w evaluations of the hash function in parallel. Any such adversary can break the binding property no better than with probability $O(d^3w^2/p)$ and the hiding property no better than with probability $O(d\sqrt{w/p})$, where p is the order of the group $\mathbb{G}$, i.e., $p \approx 2^{256}$ in Bitcoin's instantiation $\mathbb{G} = \texttt{secp256k1}$.

These bounds follow almost immediately from a generalized collision bound in Chung, Fehr, Huang, and Liao's quantum query complexity framework [Chu+20; Chu+21] and from Ambainis, Hamburg, and Unruh's generalized One-Way to Hiding Theorem (O2H) [AHU19], respectively. Since both existing results enable a style of reasoning similar to that used for classical adversaries, the derivations of the bounds are accessible without a background in quantum computing.

2 Background

UTXOs and Spending Policies. The Bitcoin blockchain associates each *unspent transaction output (UTXO)*, representing an amount of unspent funds, with a *spending policy*. This policy specifies what authorization data the owner (or the owners) of the funds needs to provide in order to access, i.e., spend, the funds. For example, the most common spending policy is a *signature policy*, which requires the owner of the funds to provide a digital signature on the spending transaction.

Traditionally, spending policies have been stored on the blockchain in the form of small programs called *scripts*. A script takes some purported authorization data as input and decides whether the data satisfies the policy with respect to the transaction. For example, the aforementioned signature policy is encoded as a script that takes as input an ECDSA signature and consists of instructions for checking the validity of the signature under a public key of the owner hardcoded in the script.

Bitcoin's scripting language is expressive enough to implement more advanced policies, which are fundamental to layer-two applications built on top of Bitcoin's transaction layer, such as payment channel networks [e.g., PD16; DW15; Aum+21]. These advanced policies can require, e.g., that a preimage of a one-way function is revealed as part of the authorization data, or that the funds can only be spent after some timeout has expired.

Most advanced policies are used to express that multiple users share ownership of some funds. These policies lock up the funds so that they can, under normal circumstances, only be spent if all users cooperate. However, if any involved user fails to cooperate, i.e., the user is unavailable or outright malicious, then, after some timeout, the policy can allow the other users to spend the funds without involvement of the uncooperative one, e.g., to get their funds back.

Taproot. The Taproot upgrade to Bitcoin in November 2021 [BIP340; BIP341; BIP342] added the ability to use "BIP340-style" Schnorr signatures [BIP340] for authorizing transactions. The Schnorr signature scheme [Sch91] offers several advantages over ECDSA used thus far in Bitcoin: Even though the underlying elliptic curve `secp256k1` [SEC2] is kept, Schnorr signatures and public keys use a more compact encoding, which reduces the on-chain size of transactions, and they support efficient batch verification [Nac+95; BGR98; Ber+12]. Moreover, they enable much simpler and more efficient constructions of advanced signing protocols, e.g., multi-signatures [e.g., NRS21; Nic+20] and threshold signatures [e.g., KG20; Chu+23; Bel+22].

The primary change in Taproot, however, is the addition of a new and more efficient on-chain spending policy format [BIP341]. A *Taproot policy*, i.e., the spending policy of a Taproot UTXO, is represented by just a public key (instead of a script containing instructions and often a hardcoded public key), which optimizes for the common case of a signature policy as described above. A transaction spending the UTXO can be authorized with a BIP340-style Schnorr signature [BIP340]

on the transaction valid under the specified public key, or alternatively, in order to retain the support for more complex policies, by revealing and satisfying a script[3] *committed to within the public key.*

Setting up a Taproot Policy (Receiving Funds). Assume that either a single user or a group of users expects to receive funds. They can create a Taproot policy as a single group element $C \in \mathbb{G}$ constructed as

$$C = rG + \mathsf{H}(rG, m)G$$

for a random scalar r and some root m of a Merkle tree [Mer79; Mer90] committing to a (possibly empty) set of scripts in its leaves. In other words, C is constructed by additively "tweaking" the group element rG by the value $\mathsf{H}(rG, m)$. Then, the Taproot policy C, suitably encoded in the form of a payment address [BIP350], can be given out to potential senders.

When multiple users are involved in setting up C, they can generate $R = rG$ without anyone learning r using the key aggregation algorithm of a suitable Schnorr multi-signature scheme such as MuSig2 [NRS21; BIP327] (or, if a threshold spending policy is desired, using a distributed key generation protocol compatible with a Schnorr threshold signature scheme such as FROST [KG20; Bel+22; Chu+23; RN23; Dha24]). In this case, each involved user needs to compute $C = R + \mathsf{H}(R, m)G$ themselves, ensuring that it is a commitment to the right values m and R before giving out C to potential senders.

Satisfying a Taproot Policy (Spending Funds). The main cryptographic trick behind Taproot is that the group element C can serve both as a Schnorr public key and as a commitment to the Merkle root m. Correspondingly, there are two ways to satisfy a Taproot policy C and authorize a transaction that spends the funds received on it:

Key-Path Spend. This spending method requires a Schnorr signature on the spending transaction valid under public key $pk = C$.
This signature can be created by signing with the tweaked secret key $sk = r + \mathsf{H}(rG, m)$ corresponding to the tweaked public key $C = R + \mathsf{H}(R, m)G$.[4] When multiple users have been involved in setting up C, they can run the signing protocol of the Schnorr multi-signature scheme (or threshold signature scheme) used for generating R to create a single Schnorr signature valid under public key C. Similar to the single-user case, the signing protocol needs to support signing with tweaked keys [BIP327; Dha24].

Script-Path Spend. This spending method requires
- the Merkle root m and the randomness $R = rG$, plus
- one of the scripts committed to as a leaf s of the Merkle tree together with a Merkle path that proves that s is a leaf of the tree with root m, plus
- inputs to the script s that satisfy it with respect to the spending transaction.

(Note that revealing r instead of rG would allow everyone to compute $sk = r + \mathsf{H}(rG, m)$ and perform a key-path spend.)

Benefits of Taproot. By committing to the scripts within the public key, Taproot favors key-path spends. In the case of a key-path spend, the authorization data that needs to be stored on the blockchain is very compact, namely just a single Schnorr signature.

While this design directly improves the efficiency of single-owner UTXOs with a simple signature policy, optimizing for key-path spends is equally beneficial for complex multi-user spending policies as those used by layer-two protocols: as long as all users involved in the spending policy are cooperative, i.e., available and honest, they can perform a key-path spend by running a Schnorr multi-signature protocol such as MuSig2 [NRS21] to create a single Schnorr signature valid under the public key C.

[3] Tweaks and improvements to the scripting language are also part of the Taproot upgrade [BIP342], but these are not relevant to this work.

[4] This requires that BIP340-style Schnorr signatures are unforgeable with respect to tweaked keys (also known as re-randomized keys or blinded keys). which has been shown [ELW23] for the very similar EdDSA variant [Ber+12] of Schnorr signatures.

In fact, a Bitcoin node obtaining and verifying the blockchain does not even notice that a possibly complex multi-user protocol has been used behind the scenes. Only if one of the involved users is not cooperative, other involved users will need to perform a (less efficient and less private) script-path spend by revealing the necessary data, including the authorization script. This enhances privacy and renders multi-user UTXOs as efficient as single-user UTXOs in terms of overhead for the Bitcoin network, at least in the cooperative case.

3 Security of Taproot

Assume that an honest user has computed a Taproot policy $C = R + \mathsf{H}(R, m)G$ from the values R and m. In this situation, Taproot is supposed to provide two security guarantees to the honest user, which, while a full formal treatment of Taproot is beyond the scope of this work, can be described informally as follows.

Theft Resistance. No adversary choosing R and m should be able to spend funds in violation of the policy, i.e., either by forging a Schnorr signature valid under public key C on some transaction tx or by opening C to $m' \neq m$. The adversary is given a key-path spend oracle that outputs Schnorr signatures generated using the tweaked secret key $r + \mathsf{H}(R, m)$ for adversarially chosen transactions tx^*. (If R has been generated with a multi-signature scheme or a threshold signature scheme, the adversary is instead given a suitable signing oracle simulating the honest users in a signing session with joint secret key r, tweak $\mathsf{H}(R, m)$, and message tx^*.) If the adversary has queried a signature on tx^*, then tx^* no longer counts as a valid forgery.

Privacy. Assuming R is chosen randomly by the honest user, no adversary (without knowledge of R) should be able to learn anything about m. The adversary is given a key-path spend oracle analogous to the one for theft resistance.

Should the adversary also be given a script-path spend oracle, which would reveal R and m? No. In theft resistance, it is anyway the adversary who chooses R and m, and in privacy, revealing R and m would make attacks trivial.

Why do the above informal descriptions treat the Merkle root m simply as an opaque message? This is a simplified treatment, justified by the well-established fact that a Merkle tree [Mer88; Mer90] is a binding vector commitment to its input messages (i.e., the scripts) hashed and stored in its leaves if the underlying hash function is collision-resistant, and this holds even against quantum adversaries [e.g., Chi+22, Section 6.2]. As a result, if the adversary can open C only to m, and the user has computed m as the Merkle root for a set of scripts, then the adversary can in turn open m only to scripts expected by the honest user. Moreover, when it comes to privacy, the Merkle root will hide some or all of the scripts until they are revealed if they are unpredictable for the adversary, which honest users can ensure by adding entropy whenever desired.

Post-Quantum Security After Disabling Schnorr Signatures. Crucially for the meaning of this work, the two security guarantees become considerably simpler assuming that spending via Schnorr signatures has been disabled in a post-quantum world. As explained below, theft resistance and privacy then simply reduce to the binding property and the hiding property, respectively, of Taproot as a commitment scheme. This justifies modeling Taproot as a commitment scheme and focusing on a formal security analysis of these standard properties of a commitment scheme.

To see why binding and hiding are sufficient, first observe that it is not necessary to provide the adversary with a key-path spend oracle. While the adversary could have obtained Schnorr signatures from key-path spends before they were disabled (or from users with old clients that have not yet been upgraded to stop producing signatures), a quantum adversary can simply compute discrete logarithms and thus generate any desired signature without an explicit key-path spend oracle.

With this in mind, privacy is equivalent to keeping m confidential, assuming that R is generated honestly and m is chosen by the adversary. But this corresponds exactly to the standard notion of hiding. For theft resistance, observe that, since spending via Schnorr signatures is no longer possible, the adversary is restricted to forging script-path spends. This reduces to the standard notion of binding: since the honest user ensures that the policy is of the form $C = R + \mathsf{H}(R, m)G$

for an expected opening message m, binding implies that no adversary can open C to some different $m' \neq m$.

4 Preliminaries

Quantum Oracle Algorithms. We work in the quantum random oracle model [Bon+11], i.e., the adversary is modeled as a *quantum oracle algorithm* $\mathcal{A}^{\mathsf{H}}$ that can perform classical and quantum computations and has classical and quantum oracle access to a random function H (i.e., H can be queried "in superposition").

Multiple oracle queries can be performed in parallel. A quantum oracle algorithm $\mathcal{A}^{\mathsf{H}}$ 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 (i.e., the total number of queries is at most $d \cdot w$.)

The motivation for the distinction between query depth and query width is that, in many cases, better bounds can be derived against attacks that exploit a parallelism. As phrased by Ambainis, Hamburg, and Unruh [AHU19], "It's easy to do 2^{64} hash queries on parallel machines — the Bitcoin network does this several times a minute — but it would take millennia to do them sequentially."

See the works underlying this paper [Chu+21; AHU19] for further details on the computational model, and see a standard textbook [NC10] for an introduction to quantum computation, which is not necessary to understand this work.

Notation. Given two sets $\mathcal{X}$ and $\mathcal{Y}$, we write $\{\mathcal{X} \to \mathcal{Y}\}$ for the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$. For an (oracle) function H, we write $\mathsf{H}\big[x' := y\big]$ for the "punctured" function such that $\mathsf{H}\big[x' := y\big](x) = y$ if $x = x'$ and $\mathsf{H}\big[x' := y\big](x) = \mathsf{H}(x)$ otherwise.

Given a finite non-empty set $\mathcal{X}$, we write $x \leftarrow\!\!\$\ \mathcal{X}$ for the operation of sampling x uniformly at random from $\mathcal{X}$. We write $A(y)$ for the operation of running an algorithm A on input y (with a uniformly-chosen random tape if A is randomized); we may just write A instead of $A()$ if A is not given any input. When the algorithm A has boolean output (e.g., when A is a game or a distinguishing adversary), we may write $\Pr[A(y)]$ as a shorthand for $\Pr[A(y) = \textbf{true}]$.

Commitment Schemes. We restrict our attention to non-interactive commitment schemes with canonical opening, i.e., the opening information for a commitment is simply the message and the randomness used to create it. Since the Taproot commitment scheme can be formalized without a setup algorithm in the random oracle model, we omit it from the definition. (Alternatively, one can think of a setup algorithm that outputs the empty string as a commitment key.) This simplifies the definition to a single algorithm.

Definition 1 (Commitment Scheme). *A* commitment scheme *with message space $\mathcal{M}$ and randomness space $\mathcal{R}$ is an algorithm $c := \mathsf{Com}(m; r)$ that takes as input a message $m \in \mathcal{M}$ and explicit randomness $r \in \mathcal{R}$, and outputs a commitment c.*

A commitment to a message m is created by choosing $r \leftarrow\!\!\$\ \mathcal{R}$ uniformly at random and returning $c = \mathsf{Com}(m; r)$. A commitment $c = \mathsf{Com}(m; r)$ is opened by revealing the message m and the randomness r used to create it, and the opening is verified by recomputing $\mathsf{Com}(m; r)$ and checking if it equals c.

5 Taproot as a Commitment Scheme

This section models Taproot as a formal commitment scheme called TwCom for "tweaking commitment". Compared to the informal descriptions in the previous sections, the model includes a function φ for encoding the resulting commitment as a bitstring. Modeling the encoding function explicitly is warranted because the encoding used in Bitcoin is not injective, which, as will become clear, affects the binding property quantitatively.

Construction 1 (TwCom). *Let $\mathbb{G}$ be a group of prime order $p = |\mathbb{G}|$ with generator G, let $\mathcal{R} \subseteq \mathbb{G}$ be a randomness space, let $\mathcal{M} \subseteq \{0,1\}^*$ be a message space, and let $\varphi : \mathbb{G} \to \{0,1\}^*$ be a (not necessarily injective) encoding function. Let $\mathsf{H} : \mathcal{R} \times \mathcal{M} \to \mathbb{Z}_p$ be a cryptographic hash function. The commitment scheme $\mathsf{TwCom}^{\mathsf{H}}[\mathbb{G}, \mathcal{R}, \mathcal{M}, \varphi]$ is defined as:*

$\mathsf{TwCom}^{\mathsf{H}}(m; R)$

$h \coloneqq \mathsf{H}(R, m)$

$C \coloneqq R + hG$

$c \coloneqq \varphi(C)$

return c

Bitcoin's Instantiation. In Bitcoin, this construction is instantiated as follows:

- $\mathbb{G} =$ `secp256k1`, the prime-order short Weierstrass elliptic curve $y^2 = x^3 + 7$ over the finite field $\mathbb{Z}_q = \{0, \ldots, q-1\}$ with $q = 2^{256} - 2^{32} - 977$, group order $p = |\mathbb{G}| \approx 2^{256}$, and base point G defined in [SEC2]
- $\mathcal{R} = \{(x, y) \in \mathbb{G} \setminus \{0_{\mathbb{G}}\} : y \in \{0, \ldots, q-1\} \text{ is even}\}$
- $\mathcal{M} = \{0,1\}^{256}$
- $\mathsf{H}(R, m) = \mathtt{SHA256}(t||x(R)||m)$, for the cryptographic hash function `SHA256` [FIPS180-4] and a constant $t \in \{0,1\}^{512}$ defined in [BIP341]
- $\varphi(C) = x(C)$

Here, $x(P)$ denotes the *x-only encoding* [BIP340] of an elliptic curve point P, which is defined as the bitstring in $\{0,1\}^{256}$ that represents the x-coordinate of P, most significant bit first, or the all-zero string $0^{256} \in \{0,1\}^{256}$ (which is the x-coordinate of no affine point) if $P = 0_{\mathbb{G}}$.[5]

Since $\mathcal{R}$ contains only elliptic curve points $R = (x, y)$ with even $y \in \{0, \ldots, q-1\}$, a point $R \in \mathcal{R}$ can be uniquely represented by (an encoding of) its x-coordinate $x(R)$, and thus the encoding of the inputs to H as bitstrings is injective.

In practice, users need to know the discrete logarithm r of the randomness R in order to produce key-path spends. A pair (r, R) with $R = rG$ and $R \in \mathcal{R}$ can be generated by sampling $r \leftarrow\!\!\$\ \mathbb{Z}_p \setminus \{0\}$, computing $R \coloneqq rG$, and negating both r and R if the latter has an odd y-coordinate. (By the negation law, negating a point (x, y) yields $-(x, y) = (x, -y)$. Since q is odd, we have that if $y \in \{0, \ldots, q-1\}$ is odd, then $-y \in \{0, \ldots, q-1\}$ is even.)

Note that `SHA256` is, in principle, vulnerable to length-extension attacks [e.g., Cor+05], but these do not apply here because whenever the output of `SHA256` is revealed, the inputs are also revealed.

6 Binding

Intuitively, one expects a commitment scheme to be binding if it is not possible for an adversary to produce a commitment and later be able to open it to two different messages. The "classical-style" formalization of this property is that it is hard for every adversary to produce a commitment along with two valid but different openings simultaneously.

However, this notion may be insufficient when dealing with quantum adversaries. Indeed, Ambainis, Rosmanis, and Unruh [ARU14] construct a commitment scheme for which a quantum adversary can hold many valid openings in superposition after outputting a commitment, making it possible to open the commitment to any message chosen later. But since the superposition

[5] In fact, Bitcoin's implementation of $x(\cdot)$ will simply reject the input $0_{\mathbb{G}}$ as invalid and error out instead of returning 0^{256}. The extended encoding function considered in this work maps $0_{\mathbb{G}}$ to 0^{256} explicitly. This avoids the need for error handling in the security definitions and does not constitute a problem because it will still be possible to prove security for the resulting extended commitment scheme. Furthermore, note that $R = 0_{\mathbb{G}}$ is excluded due to $0_{\mathbb{G}} \notin \mathcal{R}$, and $C \coloneqq R + hG = 0_{\mathbb{G}}$ will occur only with probability $1/p$ in a run of $\mathsf{TwCom}(m; R)$ if H is modeled as a (quantum) random oracle.

will collapse when producing an opening, doing so is only possible once. In other words, two valid openings can be produced, but because they cannot be produced *simultaneously*, the scheme satisfies the classical-style notion despite contradicting our informal understanding of a binding commitment.[6]

Nevertheless, in the case of Taproot, classical-style binding turns out to be sufficient. Since an honest user who would like to receive funds under a Taproot policy ensures that the commitment is of the form $c = \mathsf{TwCom}(m; R)$ for expected values m and R,[7] the "honest opening" (m, R) will be known to the user when setting up the policy. If, later, an adversary manages to publish a second, different opening (m', R'), then the openings (m, R) and (m', R') constitute a break of the classical-style binding property. As a result, classical-style binding is sufficient for Taproot.

Specifically, it turns out to be beneficial to consider a variant of classical-style binding in which the adversary wins even if the two openings differ only in the randomness but not in the message. This treatment has the minor benefit of covering the security of `OP_INTERNALKEY` [BIP349], a proposed extension to Taproot, which would allow R to be accessed from a script and used as public key in a signature verification operation (without the need to hardcode R in the script, effectively saving space). However, the primary reason for this treatment is that it simplifies the security definition: a commitment scheme Com is binding simply if Com is a collision-resistant function.

Definition 2 (Binding). *A commitment scheme* Com *that uses a cryptographic hash function* $\mathsf{H} : \mathcal{X} \to \mathcal{Y}$ *is* post-quantum collision binding *in the quantum random oracle model with error* ε *if for all quantum oracle adversaries* $\mathcal{A}$ *with query depth* d *and query width* w*,*

$$\Pr\left[\mathsf{CBIND}^{\mathcal{A}}_{\mathsf{Com}}\right] \leq \varepsilon(d, w)$$

where the game $\mathsf{CBIND}^{\mathcal{A}}_{\mathsf{Com}}$ *is defined as:*

$\mathsf{CBIND}^{\mathcal{A}}_{\mathsf{Com}}$

$\mathsf{H} \leftarrow\!\!\$\ \{\mathcal{X} \to \mathcal{Y}\}$
$((m_0, r_0), (m_1, r_1)) \coloneqq \mathcal{A}^{\mathsf{H}}()$
$c_0 \coloneqq \mathsf{Com}^{\mathsf{H}}(m_0; r_0)$
$c_1 \coloneqq \mathsf{Com}^{\mathsf{H}}(m_1; r_1)$
return $(m_0, r_0) \neq (m_1, r_1) \wedge c_0 = c_1$

The following theorem provides an upper bound for the probability that a quantum adversary can break the binding property of TwCom in the quantum random oracle model.

Theorem 1 (Binding). *The commitment scheme* $\mathsf{TwCom}^{\mathsf{H}}[\mathbb{G}, \mathcal{R}, \mathcal{M}, \varphi]$ *is post-quantum collision binding in the quantum random oracle model with error*

$$\varepsilon(d, w) \leq \left(2e(d+1)w\sqrt{10\Gamma_\varphi \frac{d+1}{p}} + \frac{2}{\sqrt{p}}\right)^2 = O\left(\frac{d^3 w^2 \Gamma_\varphi}{p}\right)$$

for $p = |\mathbb{G}|$ *and* $\Gamma_\varphi = \max_{P'} \left|\{P \in \mathbb{G} : \varphi(P) = \varphi(P')\}\right|$.

Binding of Bitcoin's Instantiation. What does this imply for Bitcoin's instantiation of TwCom? For the encoding function x used in Bitcoin, we have that each image $x(P')$ with $P' \neq 0_{\mathbb{G}}$ has exactly two preimages P' and $-P'$, and thus

$$\Gamma_\varphi = \Gamma_x = \max_{P'} \left|\{P \in \mathbb{G} : x(P) = x(P')\}\right| = 2.$$

[6] To overcome this insufficiency, Unruh [Unr16] introduced the notion of "collapse-binding".

[7] This work assumes that an empty set of scripts is represented by an invalid Merkle root, e.g., the empty bitstring as recommended in the Taproot specification [BIP341], instead of simply setting $C = R$. If $C = R$ is used instead, then privacy holds trivially, and I conjecture that theft resistance, which then corresponds to the preimage resistance of TwCom, can, like collision resistance below, be shown in the framework by Chung, Fehr, Huang, and Liao [Chu+20; Chu+21], but I have not attempted to do so.

(This is due to the fact that on a short Weierstrass curve, each element of the underlying finite field appears either as an x-coordinate of zero or two points $P' \neq 0_{\mathbb{G}}$.)

Tightness. The bound in Theorem 1 is asymptotically tight for constant Γ_φ in the sense that a w-parallel variant [Chu+21, p. 2] of the Brassard–Høyer–Tapp algorithm [BHT97; BHT98], which can evaluate TwCom on w inputs in parallel, finds a collision in TwCom with constant probability after expected $O\big((p/w^2)^{1/3}\big)$ w-parallel evaluations.[8]

6.1 Proof

Theorem 1 follows from a generalized collision-finding bound presented by Chung, Fehr, Huang, and Liao [Chu+20; Chu+21] as part of their framework for obtaining query bounds in the quantum random oracle model.

Lemma 1 (Chung, Fehr, Huang, and Liao [Chu+20, Theorem 5.30]). *Let $\mathsf{H} : \mathcal{X} \to \mathcal{Y}$ be a quantum random oracle. For every function $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$, the probability ε of any quantum oracle algorithm with query depth d and query width w outputting $x, x' \in \mathcal{X}$ such that $x \neq x'$ and $f(x, \mathsf{H}(x)) = f(x', \mathsf{H}(x'))$ is bounded by*

$$\varepsilon \leq \left(2e(d+1)w\sqrt{10\Gamma\frac{d+1}{|\mathcal{Y}|}} + \frac{2}{\sqrt{|\mathcal{Y}|}}\right)^2 = O\left(\frac{d^3w^2\Gamma}{|\mathcal{Y}|}\right)$$

for $\Gamma = \max_{x \neq x',\, y'} \big|\{y \in \mathcal{Y} : f(x,y) = f(x',y')\}\big|$.

Proof of Theorem 1. Recall that our notion of binding is equivalent to the collision resistance of the commitment algorithm. Thus, Theorem 1 is an immediate consequence of Lemma 1 when invoked with $f : (\mathcal{R} \times \mathcal{M}) \times \mathbb{Z}_p \to \mathcal{C}$ defined as

$$f\big((R,m),h\big) = \varphi(R + hG).$$

Indeed, by the definitions of f and TwCom, we have for all $m \in \mathcal{M}$ and $R \in \mathcal{R}$,

$$f\big((R,m), \mathsf{H}(R,m)\big) = \varphi(R + \mathsf{H}(R,m)G) = \mathsf{TwCom}(m;R),$$

and by the definitions of Γ and Γ_φ, we have

$$\begin{aligned}
\Gamma &= \max_{(R,m)\neq(R',m'),\, h'} \big|\{h \in \mathbb{Z}_p : f\big((R,m),h\big) = f\big((R',m'),h'\big)\}\big| \\
&= \max_{(R,m)\neq(R',m'),\, h'} \big|\{h \in \mathbb{Z}_p : \varphi(R+hG) = \varphi(R'+h'G)\}\big| \\
&= \max_{(R,m)\neq(R',m')} \max_{h'} \big|\{h \in \mathbb{Z}_p : \varphi(R+hG) = \varphi(R'+h'G)\}\big| \\
&= \max_{(R,m)\neq(R',m')} \max_{P'} \big|\{P \in \mathbb{G} : \varphi(P) = \varphi(P')\}\big| \\
&= \max_{P'} \big|\{P \in \mathbb{G} : \varphi(P) = \varphi(P')\}\big| \\
&= \Gamma_\varphi.
\end{aligned}$$

□

7 Hiding

A commitment scheme is hiding if no adversary can distinguish commitments to different messages. In the notion of hiding considered in this work, the message can depend on the randomness. This models that users may want to hardcode R (or group elements derived from it) in one of the scripts committed to in the Merkle root with root $m = m(R)$.[9] Formally, a message in the hiding game is represented as a function m from the randomness space $\mathcal{R}$ to the message space $\mathcal{M}$.

[8] In a more realistic cost model, which takes into account the actual running time and space, the classical van Orschoot–Wiener algorithm [OW99] outperforms the Brassard-Høyer–Tapp algorithm [Ber09].

[9] The proposed `OP_INTERNALKEY` extension [BIP349] mentioned in the previous section would make it unnecessary to hardcode exactly R, but not group elements derived from R.

Definition 3 (Hiding). *A commitment scheme* Com *that uses a cryptographic hash function* $\mathsf{H} : \mathcal{X} \to \mathcal{Y}$ *is* post-quantum hiding in the quantum random oracle model with error ε *if for all message pairs* $(m_0, m_1) \in \{\mathcal{R} \to \mathcal{M}\}^2$ *(which may depend on the randomness) and for all quantum oracle algorithms* $\mathcal{A}$ *with query depth* d *and query width* w*,*

$$\left|\Pr\left[\mathsf{HIDE}^{\mathcal{A}}_{\mathsf{Com}}(m_0)\right] - \Pr\left[\mathsf{HIDE}^{\mathcal{A}}_{\mathsf{Com}}(m_1)\right]\right| \leq \varepsilon(d, w)$$

where the game $\mathsf{HIDE}^{\mathcal{A}}_{\mathsf{Com}}(m_b)$ *is defined as:*

$\mathsf{HIDE}^{\mathcal{A}}_{\mathsf{Com}}(m_b)$
$\mathsf{H} \hookleftarrow \{\mathcal{X} \to \mathcal{Y}\}$
$r \hookleftarrow \mathcal{R}$
$c \coloneqq \mathsf{Com}^{\mathsf{H}}(m_b(r); r)$
return $\mathcal{A}^{\mathsf{H}}(c)$

The following theorem provides an upper bound for the probability that a post-quantum adversary can break the hiding property of TwCom in the quantum random oracle model.

Theorem 2 (Hiding). *The commitment scheme* $\mathsf{TwCom}^{\mathsf{H}}[\mathbb{G}, \mathcal{R}, \mathcal{M}, \varphi]$ *is post-quantum hiding in the quantum random oracle model with error*

$$\varepsilon(d, w) \leq 4d\sqrt{\frac{w}{|\mathcal{R}|}}.$$

Hiding of Bitcoin's Instantiation. Recall that Bitcoin's instantiation uses a restricted randomness space $\mathcal{R} = \{(x, y) \in \mathbb{G} \setminus \{0_{\mathbb{G}}\} : y \in \{0, \ldots, q-1\} \text{ is even}\}$. Since exactly half of the $p-1$ points in $\mathbb{G} \setminus \{0_{\mathbb{G}}\}$ have an even y-coordinate, we have

$$|\mathcal{R}| = \frac{p-1}{2}.$$

(For every point (x, y) in $\mathbb{G} \setminus \{0_{\mathbb{G}}\}$, its negation $-(x, y) = (x, -y)$ is also in $\mathbb{G} \setminus \{0_{\mathbb{G}}\}$, and since q is odd, we have that if y is odd, then $-y$ is even, and vice-versa.)

7.1 Proof

The proof of Theorem 2 relies on the generalized One-Way to Hiding Theorem (O2H) due to Ambainis, Hamburg, and Unruh [AHU19, Theorem 3].[10] The following Lemma 2 is an immediate consequence of their Theorem 3.

Lemma 2 (One-Way to Hiding (O2H) [AHU19, Theorem 3], Simplified). *Let* $\mathcal{S} \subseteq \mathcal{X}$ *be a random set, and let* c *be a random bitstring. Let* $\mathsf{G}, \mathsf{H} : \mathcal{X} \to \mathcal{Y}$ *be random functions satisfying* $\forall x \notin \mathcal{S}.\ \mathsf{G}(x) = \mathsf{H}(x)$*. (*$\mathcal{S}, \mathsf{G}, \mathsf{H}, z$ *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 such that*

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

Proof of Theorem 2. The proof will proceed by a sequence of game hops as defined in Fig. 1. 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

$$\mathsf{HIDE}^{\mathcal{A}}_{\mathsf{TwCom}}(m_b) = \mathsf{Game}^{\mathcal{A}}_1(m_b) \approx \mathsf{Game}^{\mathcal{A}}_2(m_b) = \mathsf{Game}^{\mathcal{A}}_3(m_b),$$

[10] See Heidler and Unruh [HU25] for a full formalization in the Isabelle theorem prover (except that the bound in their formalization is worse by a factor of 2).

$\mathsf{HIDE}^{\mathcal{A}}_{\mathsf{TwCom}}(m_b)$	$\mathsf{Game}^{\mathcal{A}}_1(m_b)$	$\mathsf{Game}^{\mathcal{A}}_2(m_b)$	$\mathsf{Game}^{\mathcal{A}}_3(m_b)$
$\mathsf{H} \leftarrow\!\$\ \{\mathbb{Z}_p \to \{0,1\}^*\}$	$\mathsf{H} \leftarrow\!\$\ \{\mathbb{Z}_p \to \{0,1\}^*\}$	$\mathsf{H} \leftarrow\!\$\ \{\mathbb{Z}_p \to \{0,1\}^*\}$	$\mathsf{H} \leftarrow\!\$\ \{\mathbb{Z}_p \to \{0,1\}^*\}$
	$h' \leftarrow\!\$\ \mathbb{Z}_p$	$h' \leftarrow\!\$\ \mathbb{Z}_p$	$h' \leftarrow\!\$\ \mathbb{Z}_p$
	$\mathsf{G} \coloneqq \mathsf{H}\big[(R, m_b(R)) \coloneqq h'\big]$	$\mathsf{G} \coloneqq \mathsf{H}\big[(R, m_b(R)) \coloneqq h'\big]$	$\mathsf{G} \leftarrow\!\$\ \{\mathbb{Z}_p \to \{0,1\}^*\}$
$R \leftarrow\!\$\ \mathbb{G}$	$R \leftarrow\!\$\ \mathbb{G}$	$R \leftarrow\!\$\ \mathbb{G}$	$R \leftarrow\!\$\ \mathbb{G}$
$h \coloneqq \mathsf{H}(R, m_b(R))$	$h \coloneqq \mathsf{H}(R, m_b(R))$	$h \coloneqq \mathsf{H}(R, m_b(R))$	$h \leftarrow\!\$\ \mathbb{Z}_p$
$C \coloneqq R + hG$	$C \coloneqq R + hG$	$C \coloneqq R + hG$	$C \coloneqq R + hG$
$c \coloneqq \varphi(C)$	$c \coloneqq \varphi(C)$	$c \coloneqq \varphi(C)$	$c \coloneqq \varphi(C)$
return $\mathcal{A}^{\mathsf{H}}(c)$	**return** $\mathcal{A}^{\mathsf{H}}(c)$	**return** $\mathcal{A}^{\mathsf{G}}(c)$	**return** $\mathcal{A}^{\mathsf{G}}(c)$

Fig. 1. Games for the proof of Theorem 2. Lines in gray indicate changes from the preceding game.

and then we will show

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

This will enable us to bound the difference between $\mathsf{HIDE}^{\mathcal{A}}_{\mathsf{TwCom}}(m_0)$ and $\mathsf{HIDE}^{\mathcal{A}}_{\mathsf{TwCom}}(m_1)$ via

$$\begin{aligned}\mathsf{HIDE}^{\mathcal{A}}_{\mathsf{TwCom}}(m_0) = \mathsf{Game}^{\mathcal{A}}_1(m_0) &\approx \mathsf{Game}^{\mathcal{A}}_2(m_0) = \mathsf{Game}^{\mathcal{A}}_3(m_0) \\ &= \mathsf{Game}^{\mathcal{A}}_3(m_1) = \mathsf{Game}^{\mathcal{A}}_2(m_1) \approx \mathsf{Game}^{\mathcal{A}}_1(m_1) = \mathsf{HIDE}^{\mathcal{A}}_{\mathsf{TwCom}}(m_1).\end{aligned} \tag{1}$$

Fix a quantum oracle algorithm $\mathcal{A}$ with query depth d and query width w and fix a message pair $(m_0, m_1) \in \{\mathcal{R} \to \mathcal{M}\}^2$. Also fix $b \in \{0,1\}$.

$\mathsf{HIDE}^{\mathcal{A}}_{\mathsf{TwCom}}(m_b)$. This is the hiding game from Definition 3 after inlining TwCom.

$\mathsf{HIDE}^{\mathcal{A}}_{\mathsf{TwCom}}(m_b) = \mathsf{Game}^{\mathcal{A}}_1(m_b)$. The code of $\mathsf{Game}^{\mathcal{A}}_1(m_b)$ is the result of adding the assignments of variables h' and G to $\mathsf{HIDE}^{\mathcal{A}}_{\mathsf{TwCom}}$. 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

$$\Big|\Pr\big[\mathsf{HIDE}^{\mathcal{A}}_{\mathsf{TwCom}}(m_b)\big] - \Pr\big[\mathsf{Game}^{\mathcal{A}}_1(m_b)\big]\Big| = 0. \tag{2}$$

$\mathsf{Game}^{\mathcal{A}}_1(m_b) \approx \mathsf{Game}^{\mathcal{A}}_2(m_b)$. The code of $\mathsf{Game}^{\mathcal{A}}_2(m_b)$ is like that of $\mathsf{Game}^{\mathcal{A}}_1(m_b)$ with the difference that $\mathcal{A}$ is given oracle access to the function $\mathsf{G} \coloneqq \mathsf{H}\big[(R, m_b(R)) \coloneqq h'\big]$ instead of H, where $h' \leftarrow\!\$\ \mathbb{Z}_p$ is freshly sampled. In other words, G is a random function that is identical to H except at position $(R, m_b(R))$, at which its value has been resampled.

In order to bound the probability that $\mathcal{A}$ distinguishes H and G, we apply Lemma 2 instantiated with $\mathcal{S} = \{(R, m_b(R))\}$ and $z = c$ to $\mathcal{A}$. Here, the values R, c, b, H, G have the same distribution as the values of the corresponding variables in $\mathsf{Game}^{\mathcal{A}}_1(m_b)$ and $\mathsf{Game}^{\mathcal{A}}_2(m_b)$; note that the code assigning these variables is identical in the two games. This yields

$$\begin{aligned}\Big|\Pr\big[\mathsf{Game}^{\mathcal{A}}_1(m_b)\big] - \Pr\big[\mathsf{Game}^{\mathcal{A}}_2(m_b)\big]\Big| &= \Big|\Pr\big[\mathcal{A}^{\mathsf{H}}(c) \text{ in } \mathsf{Game}^{\mathcal{A}}_1\big] - \Pr\big[\mathcal{A}^{\mathsf{G}}(c) \text{ in } \mathsf{Game}^{\mathcal{A}}_2\big]\Big| \\ &\le 2d\sqrt{\Pr\big[\mathcal{E}^{\mathsf{G}}(c) \cap \mathcal{S} \neq \emptyset \text{ in } \mathsf{Game}^{\mathcal{E}}_2(m_b)\big]}.\end{aligned} \tag{3}$$

As will be shown below, when discussing the hop from $\mathsf{Game}^{\mathcal{A}}_2(m_b)$ to $\mathsf{Game}^{\mathcal{A}}_3(m_b)$, no quantum oracle algorithm $\mathcal{A}$ can observe the difference between $\mathsf{Game}^{\mathcal{A}}_2(m_b)$ to $\mathsf{Game}^{\mathcal{A}}_3(m_b)$. This is true in particular for $\mathcal{E}$, and thus we have

$$\Pr\big[\mathcal{E}^{\mathsf{G}}(c) \cap \mathcal{S} \neq \emptyset \text{ in } \mathsf{Game}^{\mathcal{E}}_2(m_b)\big] = \Pr\big[\mathcal{E}^{\mathsf{G}}(c) \cap \mathcal{S} \neq \emptyset \text{ in } \mathsf{Game}^{\mathcal{E}}_3(m_b)\big]. \tag{4}$$

In $\mathsf{Game}_3^{\mathcal{E}}(m_b)$, the inputs to $\mathcal{E}^{\mathsf{G}}(c)$, namely $\mathsf{G} \leftarrow\!\!\$\ \{\mathbb{Z}_p \to \{0,1\}^*\}$ and $c = \varphi(R + hG)$ with $h \leftarrow\!\!\$\ \mathbb{Z}_p$, are distributed independently of $\mathcal{S} = \{(R, m_b(R))\}$. Thus $\mathcal{E}^{\mathsf{G}}(c)$ has no better chance of outputting a set of size at most w containing the only element $(R, m_b(R))$ of $\mathcal{S}$ than guessing w different values of $R \in \mathcal{R}$, i.e.,

$$\Pr\left[\mathcal{E}^{\mathsf{G}}(c) \cap \mathcal{S} \neq \emptyset \text{ in } \mathsf{Game}_3^{\mathcal{E}}(m_b)\right] = \frac{\left|\mathcal{E}^{\mathsf{G}}(c)\right|}{|\mathcal{R}|} \leq \frac{w}{|\mathcal{R}|}. \tag{5}$$

By combining (3)–(5), we get

$$\left|\Pr\left[\mathsf{Game}_1^{\mathcal{A}}(m_b)\right] - \Pr\left[\mathsf{Game}_2^{\mathcal{A}}(m_b)\right]\right| \leq 2d\sqrt{\frac{w}{|\mathcal{R}|}}. \tag{6}$$

$\mathsf{Game}_2^{\mathcal{A}}(m_b) = \mathsf{Game}_3^{\mathcal{A}}(m_b)$. In $\mathsf{Game}_2^{\mathcal{A}}$, the random value $\mathsf{H}(R, m_b(R))$ is used only in the assignment to h. (Recall that $\mathcal{A}^{\mathsf{G}}$ is not given H but only G, which is identical to H except at position $(R, m_b(R))$ where the value $\mathsf{G}(R, m_b(R))$ has been resampled.) Thus $\mathsf{H}(R, m_b(R))$ will be distributed independently of everything else in the game, and can be replaced by a freshly sampled random value without changing its distribution. Similarly, G can be replaced by a freshly sampled random oracle without changing its distribution. As a result, $\mathcal{A}$ cannot observe these two changes, and since they constitute the only differences between $\mathsf{Game}_2^{\mathcal{A}}(m_b)$ and $\mathsf{Game}_3^{\mathcal{A}}(m_b)$, we have

$$\left|\Pr\left[\mathsf{Game}_2^{\mathcal{A}}(m_b)\right] - \Pr\left[\mathsf{Game}_3^{\mathcal{A}}(m_b)\right]\right| = 0. \tag{7}$$

$\mathsf{Game}_3^{\mathcal{A}}(m_0) = \mathsf{Game}_3^{\mathcal{A}}(m_1)$. Since $\mathsf{Game}_3^{\mathcal{A}}(m_b)$ does not access its input m_b at all, it is clear that

$$\left|\Pr\left[\mathsf{Game}_3^{\mathcal{A}}(m_0)\right] - \Pr\left[\mathsf{Game}_3^{\mathcal{A}}(m_1)\right]\right| = 0. \tag{8}$$

Finalizing the proof. We combine the bounds (2) and (6)–(8), following the sequence of hops laid out in (1). This yields

$$\begin{aligned}\left|\Pr\left[\mathsf{HIDE}_{\mathsf{TwCom}}^{\mathcal{A}}(m_0)\right] - \Pr\left[\mathsf{HIDE}_{\mathsf{TwCom}}^{\mathcal{A}}(m_1)\right]\right| &\leq \left|\Pr\left[\mathsf{HIDE}_{\mathsf{TwCom}}^{\mathcal{A}}(m_0)\right] - \Pr\left[\mathsf{Game}_3^{\mathcal{A}}(m_0)\right]\right| \\ &\quad + \left|\Pr\left[\mathsf{Game}_3^{\mathcal{A}}(m_0)\right] - \Pr\left[\mathsf{Game}_3^{\mathcal{A}}(m_1)\right]\right| \\ &\quad + \left|\Pr\left[\mathsf{Game}_3^{\mathcal{A}}(m_1)\right] - \Pr\left[\mathsf{HIDE}_{\mathsf{TwCom}}^{\mathcal{A}}(m_1)\right]\right| \\ &\leq 4d\sqrt{\frac{w}{|\mathcal{R}|}}.\end{aligned}$$

□

Acknowledgements. I thank Jonas Nick, Pieter Wuille, and Ethan Heilman for their valuable comments.

References

[Aar+24] Marius A. Aardal, Diego F. Aranha, Katharina Boudgoust, Sebastian Kolby, and Akira Takahashi. "Aggregating Falcon Signatures with LaBRADOR". In: *CRYPTO 2024*. DOI: 10.1007/978-3-031-68376-3_3.

[AHU19] Andris Ambainis, Mike Hamburg, and Dominique Unruh. "Quantum Security Proofs Using Semi-classical Oracles". In: *CRYPTO 2019*. DOI: 10.1007/978-3-030-26951-7_10.

[Alk+20] Nabil Alkeilani Alkadri, Poulami Das, Andreas Erwig, Sebastian Faust, Juliane Krämer, Siavash Riahi, and Patrick Struck. "Deterministic Wallets in a Quantum World". In: *ACM CCS 2020*. DOI: 10.1145/3372297.3423361.

[And+98] Ross J. Anderson, Francesco Bergadano, Bruno Crispo, Jong-Hyeon Lee, Charalampos Manifavas, and Roger M. Needham. "A New Family of Authentication Protocols". In: *ACM SIGOPS Oper. Syst. Rev.* 32.4 (1998). DOI: 10.1145/302350.302353.

[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.

[ASY22] Shweta Agrawal, Damien Stehlé, and Anshu Yadav. "Round-Optimal Lattice-Based Threshold Signatures, Revisited". In: *ICALP 2022*. DOI: 10.4230/LIPIcs.ICALP.2022.8.

[Aum+21] Lukas Aumayr, Matteo Maffei, Oguzhan Ersoy, Andreas Erwig, Sebastian Faust, Siavash Riahi, Kristina Hostáková, and Pedro Moreno-Sanchez. "Bitcoin-Compatible Virtual Channels". In: *2021 IEEE Symposium on Security and Privacy*. DOI: 10.1109/SP40001.2021.00097.

[Bea24] Hunter Beast. "Proposing a P2QRH BIP Towards a Quantum Resistant Soft Fork". Posting to Bitcoin Development Mailing List (bitcoin-dev). 2024. URL: https://groups.google.com/g/bitcoindev/c/Aee8xKuIC2s/m/cu6xej1mBQAJ.

[Bea25] Hunter Beast. Posting to Bitcoin Development Mailing List (bitcoin-dev). 2025. URL: https://groups.google.com/g/bitcoindev/c/8PM6iZCeDMc.

[Bel+22] Mihir Bellare, Elizabeth C. Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu. "Better than Advertised Security for Non-interactive Threshold Signatures". In: *CRYPTO 2022*. DOI: 10.1007/978-3-031-15985-5_18.

[BEL25] Sathvika Balumuri, Edward Eaton, and Philippe Lamontagne. "Quantum-Safe Public Key Blinding from MPC-in-the-Head Signature Schemes". 2025. DOI: 10.1007/978-3-031-95761-1_13.

[Ben+18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. "Scalable, transparent, and post-quantum secure computational integrity". 2018. IACR: 2018/046.

[Ber+12] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. "High-speed high-security signatures". In: *Journal of Cryptographic Engineering* 2.2 (2012). DOI: 10.1007/s13389-012-0027-1.

[Ber09] Daniel J. Bernstein. "Cost Analysis of Hash Collisions: Will Quantum Computers Make SHARCS obsolete?" In: URL: https://cr.yp.to/hash/collisioncost-20090823.pdf.

[BGR98] Mihir Bellare, Juan A. Garay, and Tal Rabin. "Fast Batch Verification for Modular Exponentiation and Digital Signatures". In: *EUROCRYPT'98*. DOI: 10.1007/BFb0054130.

[BHT97] Gilles Brassard, Peter Høyer, and Alain Tapp. "Quantum Algorithm for the Collision Problem". 1997. arXiv: quant-ph/9705002.

[BHT98] Gilles Brassard, Peter Høyer, and Alain Tapp. "Quantum Cryptanalysis of Hash and Claw-Free Functions". In: *LATIN'98*. DOI: 10.1007/BFB0054319.

[BIP32] Pieter Wuille. "Hierarchical Deterministic Wallets". Bitcoin Improvement Proposal (BIP) 32. 2012. URL: https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki.

[BIP324] Dhruv Mehta, Tim Ruffing, Jonas Schnelli, and Pieter Wuille. "Version 2 P2P Encrypted Transport Protocol". Bitcoin Improvement Proposal (BIP) 324. 2019. URL: https://github.com/bitcoin/bips/blob/master/bip-0324.mediawiki.

[BIP327] Jonas Nick, Tim Ruffing, and Elliott Jin. "MuSig2 for BIP340-compatible Multi-Signatures". Bitcoin Improvement Proposal (BIP) 327. 2022. URL: https://github.com/bitcoin/bips/blob/master/bip-0327.mediawiki.

[BIP340] Pieter Wuille, Jonas Nick, and Tim Ruffing. "Schnorr Signatures for secp256k1". Bitcoin Improvement Proposal (BIP) 340. 2020. URL: https://github.com/bitcoin/bips/blob/master/bip-0340.mediawiki.

[BIP341] Pieter Wuille, Jonas Nick, and Anthony Towns. "Taproot: SegWit Version 1 Spending Rules". Bitcoin Improvement Proposal (BIP) 341. 2020. URL: https://github.com/bitcoin/bips/blob/master/bip-0341.mediawiki.

[BIP342] Pieter Wuille, Jonas Nick, and Anthony Towns. "Validation of Taproot Scripts". Bitcoin Improvement Proposal (BIP) 342. 2020. URL: https://github.com/bitcoin/bips/blob/master/bip-0342.mediawiki.

[BIP349] Brandon Black and Jeremy Rubin. "OP_INTERNALKEY". Bitcoin Improvement Proposal (BIP) 349. 2024. URL: https://github.com/bitcoin/bips/blob/master/bip-0349.md.

[BIP350] Pieter Wuille. "Bech32m Format for v1+ Witness Addresses". Bitcoin Improvement Proposal (BIP) 350. 2020. URL: https://github.com/bitcoin/bips/blob/master/bip-0350.mediawiki.

[BM14] Joseph Bonneau and Andrew Miller. "Fawkescoin: A Cryptocurrency Without Public-Key Cryptography". In: *SPW 2014*. DOI: 10.1007/978-3-319-12400-1_35.

[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.

[BS24] Bolton Bailey and Or Sattath. "51% Attack via Difficulty Increase with a Small Quantum Miner". 2024. arXiv: 2403.08023 [quant-ph].

[BTT22] Cecilia Boschini, Akira Takahashi, and Mehdi Tibouchi. "MuSig-L: Lattice-Based Multi-signature with Single-Round Online Phase". In: *CRYPTO 2022*. DOI: 10.1007/978-3-031-15979-4_10.

[Cha+17] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. "Post-Quantum Zero-Knowledge and Signatures from Symmetric-Key Primitives". In: *ACM CCS 2017*. DOI: 10.1145/3133956.3133997.

[Chi+22] Alessandro Chiesa, Fermi Ma, Nicholas Spooner, and Mark Zhandry. "Post-Quantum Succinct Arguments: Breaking the Quantum Rewinding Barrier". In: *62nd FOCS*. DOI: 10.1109/FOCS52979.2021.00014.

[Chu+20] Kai-Min Chung, Serge Fehr, Yu-Hsuan Huang, and Tai-Ning Liao. "On the Compressed-Oracle Technique, and Post-Quantum Security of Proofs of Sequential Work". 2020. IACR: 2020/1305.

[Chu+21] Kai-Min Chung, Serge Fehr, Yu-Hsuan Huang, and Tai-Ning Liao. "On the Compressed-Oracle Technique, and Post-Quantum Security of Proofs of Sequential Work". In: *EUROCRYPT 2021*. DOI: 10.1007/978-3-030-77886-6_21.

[Chu+23] Hien Chu, Paul Gerhart, Tim Ruffing, and Dominique Schröder. "Practical Schnorr Threshold Signatures Without the Algebraic Group Model". In: *CRYPTO 2023*. DOI: 10.1007/978-3-031-38557-5_24.

[Coj+19] Alexandru Cojocaru, Juan Garay, Aggelos Kiayias, Fang Song, and Petros Wallden. "The Bitcoin Backbone Protocol Against Quantum Adversaries". 2019. IACR: 2019/1150.

[Cor+05] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. "Merkle-Damgård Revisited: How to Construct a Hash Function". In: *CRYPTO 2005*. DOI: 10.1007/11535218_26.

[Cor24] Matt Corallo. "Trivial QC Signatures with Clean Upgrade Path". Posting to Bitcoin Development Mailing List (bitcoin-dev). 2024. URL: https://groups.google.com/g/bitcoindev/c/8O857bRSVV8/m/j4ngMjlHCAAJ.

[Das+24] Poulami Das, Andreas Erwig, Michael Meyer, and Patrick Struck. "Efficient Post-Quantum Secure Deterministic Threshold Wallets from Isogenies". In: *ASIACCS 24*. DOI: 10.1145/3634737.3657008.

[Dha24] Sivaram Dhakshinamoorthy. "FROST Signing for BIP340-Compatible Threshold Signatures". Draft of Bitcoin Improvement Proposal (BIP). Work in progress. 2024. URL: https://github.com/siv2r/bip-frost-signing.

[DM20] Luca De Feo and Michael Meyer. "Threshold Schemes from Isogeny Assumptions". In: *PKC 2020*. DOI: 10.1007/978-3-030-45388-6_7.

[Dra+25] Justin Drake, Dmitry Khovratovich, Mikhail A. Kudinov, and Benedikt Wagner. "Hash-Based Multi-Signatures for Post-Quantum Ethereum". In: *CiC* 2.1 (2025). DOI: 10.62056/aey7qjp10.

[Dry25] Tadge Dryja. "Post-Quantum Commit / Reveal Fawkescoin Variant as a Soft Fork". Posting to Bitcoin Development Mailing List (bitcoin-dev). 2025. URL: https://groups.google.com/g/bitcoindev/c/LpWOcXMcvk8/m/DjaiWnViAQAJ.

[DW15] Christian Decker and Roger Wattenhofer. "Fast and Scalable Payment Channel Networks". In: *SSS 2015*. DOI: 10.1007/978-3-662-48653-5_1.

[ELW23] Edward Eaton, Tancrède Lepoint, and Christopher A. Wood. "Security Analysis of Signature Schemes with Key Blinding". 2023. IACR: 2023/380.

[FIPS180-4] "Secure Hash Standard (SHS)". The Federal Information Processing Standards Publication (FIPS) 180-4. NIST. 2015. DOI: 10.6028/NIST.FIPS.180-4.

[Fou20a] Lloyd Fournier. "Hash Function Requirements for Taproot". Posting to Bitcoin Development Mailing List (bitcoin-dev). 2020. URL: https://web.archive.org/web/20231114131302/https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2020-March/017670.html.

[Fou20b] Lloyd Fournier. "Taproot in the Generic Group Model". Poster at FC'20. See also [Fou20a]. 2020. URL: https://github.com/LLFourn/taproot-ggm/blob/master/main.pdf.

[Hei25] Ethan Heilman. "Post Quantum Signatures and Scaling Bitcoin". Posting to Bitcoin Development Mailing List (bitcoin-dev). 2025. URL: https://groups.google.com/g/bitcoindev/c/wKizvPUfO7w/m/hG9cwpOABQAJ.

[HU25] Katharina Heidler and Dominique Unruh. "Formalizing the One-Way to Hiding Theorem". In: *CPP 2025*. DOI: 10.1145/3703595.3705887.

[IKS19] Dragos Ioan Ilie, William J. Knottenbelt, and Iain D. Stewart. "Committing to Quantum Resistance, Better: A Speed-and-Risk-Configurable Defence for Bitcoin Against a Fast Quantum Computing Attack". In: *MARBLE 2019*.

[KG20] Chelsea Komlo and Ian Goldberg. "FROST: Flexible Round-Optimized Schnorr Threshold Signatures". In: *SAC 2020*. DOI: 10.1007/978-3-030-81652-0_2.

[KKW18] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. "Improved Non-Interactive Zero Knowledge with Applications to Post-Quantum Signatures". In: *ACM CCS 2018*. DOI: 10.1145/3243734.3243805.

[Lop25a] Jameson Lopp. "A Post Quantum Migration Proposal". Posting to Bitcoin Development Mailing List (bitcoin-dev). 2025. URL: https://groups.google.com/g/bitcoindev/c/uEaf4bj07rE/m/RMkPWnrSBwAJ.

[Lop25b] Jameson Lopp. "Against Allowing Quantum Recovery of Bitcoin". 2025. URL: https://blog.lopp.net/against-quantum-recovery-of-bitcoin/.

[LRS19] Troy Lee, Maharshi Ray, and Miklos Santha. "Strategies for Quantum Races". In: *ITCS 2019*. DOI: 10.4230/LIPICS.ITCS.2019.51.

[Mer79] Ralph C. Merkle. "Secrecy, Authentication, and Public Key Systems". PhD thesis. Stanford University, 1979.

[Mer88] Ralph C. Merkle. "A Digital Signature Based on a Conventional Encryption Function". In: *CRYPTO'87*. DOI: 10.1007/3-540-48184-2_32.

[Mer90] Ralph C. Merkle. "A Certified Digital Signature". In: *CRYPTO'89*. DOI: 10.1007/0-387-34805-0_21.

[Nac+95] David Naccache, David M'Raïhi, Serge Vaudenay, and Dan Raphaeli. "Can D.S.A. be Improved? Complexity Trade-Offs with the Digital Signature Standard". In: *EUROCRYPT'94*. DOI: 10.1007/BFb0053426.

[NC10] Michael A. Nielsen and Isaac L. Chuang. "Quantum Computation and Quantum Information (10th Anniversary edition)". Cambridge University Press, 2010. ISBN: 978-1-10-700217-3.

[Nic+20] Jonas Nick, Tim Ruffing, Yannick Seurin, and Pieter Wuille. "MuSig-DN: Schnorr Multi-Signatures with Verifiably Deterministic Nonces". In: *ACM CCS 2020*. DOI: 10.1145/3372297.3417236.

[NRS21] Jonas Nick, Tim Ruffing, and Yannick Seurin. "MuSig2: Simple Two-Round Schnorr Multi-signatures". In: *CRYPTO 2021*. DOI: 10.1007/978-3-030-84242-0_8.

[OW99] Paul C. van Oorschot and Michael J. Wiener. "Parallel Collision Search with Cryptanalytic Applications". In: *Journal of Cryptology* 12.1 (1999). DOI: 10.1007/PL00003816.

[PD16] Joseph Poon and Thaddeus Dryja. "The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments". 2016. URL: https://lightning.network/lightning-network-paper.pdf.

[Poe20] Andrew Poelstra. "Taproot Security Proof". 2020. URL: https://github.com/apoelstra/taproot/blob/master/main.pdf.

[RK24] Farzin Renan and Péter Kutas. "SQIAsignHD: SQIsignHD Adaptor Signature". 2024. IACR: 2024/561.

[RN23] Tim Ruffing and Jonas Nick. "ChillDKG: Distributed Key Generation for FROST". Draft of Bitcoin Improvement Proposal (BIP). Work in progress. 2023. URL: https://github.com/BlockstreamResearch/bip-frost-dkg.

[Ruf18] Tim Ruffing. "Transition to Post-Quantum". Posting to Bitcoin Development Mailing List (bitcoin-dev). 2018. URL: https://web.archive.org/web/20180317155642/https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2018-February/015758.html.

[Sat20] Or Sattath. "On the Insecurity of Quantum Bitcoin Mining". In: *Int. J. Inf. Sec.* 19.3 (2020). DOI: 10.1007/S10207-020-00493-9.

[Sch91] Claus-Peter Schnorr. "Efficient Signature Generation by Smart Cards". In: *Journal of Cryptology* 4.3 (1991). DOI: 10.1007/BF00196725.

[SEC2] "Recommended Elliptic Curve Domain Parameters". Standards for Efficient Crytopgrahy (SEC) 2. Version 2.0. Certicom Research. 2010. URL: http://www.secg.org/sec2-v2.pdf.

[Sho94] Peter W. Shor. "Algorithms for Quantum Computation: Discrete Logarithms and Factoring". In: *35th FOCS*. DOI: 10.1109/SFCS.1994.365700.

[Ste+18] Iain Stewart, Dragos Ioan Ilie, Alexei Zamyatin, Sam Werner, M. F. Torshizi, and William J. Knottenbelt. "Committing to Quantum Resistance: A Slow Defence for Bitcoin Against a Fast Quantum Computing Attack". In: *Royal Society Open Science* 5.6 (2018). DOI: 10.1098/rsos.180410.

[SW23] Or Sattath and Shai Wyborski. "Protecting Quantum Procrastinators with Signature Lifting: A Case Study in Cryptocurrencies". 2023. IACR: 2023/362.

[Unr16] Dominique Unruh. "Computationally Binding Quantum Commitments". In: *EUROCRYPT 2016*. DOI: 10.1007/978-3-662-49896-5_18.

[Zha+24] Xinyu Zhang, Ron Steinfeld, Muhammed F. Esgin, Joseph K. Liu, Dongxi Liu, and Sushmita Ruj. "Loquat: A SNARK-Friendly Post-quantum Signature Based on the Legendre PRF with Applications in Ring and Aggregate Signatures". In: *CRYPTO 2024*. DOI: 10.1007/978-3-031-68376-3_1.