The Design and Formal Analysis of the Quantum Message Authentication Code

John G. Underhill

Quantum Resistant Cryptographic Solutions Corporation

Abstract. QMAC is a message authentication code that combines a polynomial hash over the field $GF(2^{256})$ with keys derived from the cSHAKE customizable hash function. This paper presents a formal cryptanalysis of the construction and provides a detailed examination of its algebraic structure, security properties, and implementation alignment. The analysis includes an engineering description of the mechanism based directly on the reference source code, a complete treatment of the polynomial hash function and its ε almost universal bounds, and a reduction based proof of existential unforgeability under chosen message attack. The reduction separates the universal hashing component from the pseudo-randomness of the cSHAKE based key derivation and yields explicit bounds that depend on adversarial query limits and field size. Quantum security considerations are examined within a quantum oracle model and are related to known results on polynomial hashing and cSHAKE in the presence of quantum adversaries. The paper concludes with a comparative and implementation oriented evaluation of QMAC and a discussion of practical deployment conditions and limitations.

1 Introduction

Message authentication codes are essential components of modern cryptographic protocols. They provide integrity and authenticity guarantees for messages that travel across potentially adversarial networks. Many widely deployed constructions, such as GMAC and HMAC, rely on primitives whose long term security depends on the hardness of classical problems. The transition to quantum resistant cryptographic systems has renewed interest in MAC designs that rest purely on symmetric and information theoretic principles.

QMAC is a message authentication code that combines a polynomial hash over the field $GF(2^{256})$ with keys derived from the customizable hash function cSHAKE. Its structure follows the approach of GMAC, which uses polynomial hashing over $GF(2^{128})$, but QMAC replaces the smaller field with a larger one and adopts cSHAKE based key derivation rather than AES based processing. This offers a simple path to a symmetric authentication mechanism that avoids reliance on block ciphers and supports quantum resistant deployment objectives. The polynomial hash enables fast evaluation on common architectures, and the use of cSHAKE provides a flexible key derivation interface and strong pseudo-randomness guarantees.

This paper presents a formal cryptanalysis of QMAC, addressing both its algebraic structure and its security properties. A complete engineering level description of the construction is provided to establish a precise link between the reference implementation and the mathematical model used in the analysis. The algebraic properties of the polynomial hash are examined in detail, including proofs of its ε almost universal behavior over an irreducible polynomial field of degree 256. These properties are combined with assumptions on the pseudo-randomness of the cSHAKE based key derivation to obtain explicit unforgeability bounds in the chosen message setting. The security model separates the universal hashing component from the key derivation component, and the resulting reduction yields interpretable expressions for the adversarial advantage.

The analysis also considers attackers with quantum capabilities. The polynomial hash is evaluated in the context of quantum oracle access, and the use of cSHAKE is examined under standard assumptions about the behavior of extendable output functions in the presence of quantum queries. These results are related to work on polynomial MACs and hash based constructions in the quantum setting.

1.1 Background and Motivation

The design of QMAC is motivated by several goals. The first is to obtain a MAC that is suitable for use in symmetric systems that aim for quantum resistant security without depending on asymmetric primitives. The second is to create a construction that retains the simplicity and efficiency of GMAC while avoiding the need for AES. The third is to provide a mechanism that integrates well with cSHAKE, which already forms the basis for several modern post quantum schemes. By using a polynomial hash over a larger field, QMAC inherits strong diffusion properties and retains a compact and regular structure that maps well to vectorized instructions on common CPUs.

1.2 Contributions

This paper provides the following contributions.

- A complete engineering level description of QMAC based directly on the reference implementation. This description captures the exact operational details of the construction, including block processing, field arithmetic, reduction rules, cSHAKE based key derivation, and state management.

- A formal model of the polynomial hash over $GF(2^{256})$, including proofs of its ε almost universal collision bounds.
- An explicit reduction based proof of existential unforgeability under chosen message attack. The reduction separates the universal hashing component from the pseudorandomness of the cSHAKE based key derivation and yields a concrete bound involving both contributions.
- A discussion of quantum security in a quantum oracle model, including the behavior of polynomial hashing and cSHAKE under quantum queries.
- A comparative cryptanalytic evaluation of QMAC in relation to GMAC and KMAC, highlighting algebraic properties, performance considerations, and deployment characteristics.

1.3 Organization of the Paper

Section 2 provides an engineering level description of QMAC that reflects the exact behavior of the reference implementation. Section 3 introduces the mathematical preliminaries, including notation, the field $GF(2^{256})$, and the structure of the polynomial hash. Section 4 gives a formal specification of QMAC in functional terms. Section 5 defines the security notions and adversarial model. Section 6 presents the algebraic analysis of the polynomial hash and its universality bounds. Section 7 contains the security proofs, including the reduction for existential unforgeability. Section 8 addresses quantum security considerations. Section 9 provides comparative and cryptanalytic evaluations. Section 10 discusses implementation conformance and side channel issues. The paper concludes with a summary of results, limitations, and directions for future work.

2 Engineering Description of QMAC

This section provides a functional description of QMAC based on the reference C implementation. All operations are described at the level of mathematical data flow, independent of optimization details. The purpose is to capture the exact behavior of the algorithm as it is defined by the code, expressed in a form suitable for formal reasoning.

2.1 Parameters, Constants, and State Layout

QMAC operates on fixed size blocks and uses fixed size subkeys and internal state. These parameters are expressed in the implementation as constants and are formalized as follows.

$$\begin{aligned}\text{Block size} &= 32 \text{ bytes},\\ \text{Key size} &= 32 \text{ bytes},\\ \text{Tag size} &= 32 \text{ bytes},\\ \text{Nonce size} &= 32 \text{ bytes}.\end{aligned}$$

The internal state consists of three field elements in $GF(2^{256})$, each represented as a 256 bit vector:

$$H \in GF(2^{256}), \qquad F \in GF(2^{256}), \qquad Y \in GF(2^{256}).$$

During initialization Y is set to the zero element.

2.2 Key Derivation with cSHAKE

Initialization derives the pair of subkeys (H, F) using a single call to cSHAKE that absorbs a secret key K, a nonce N, and an optional information string info. The C implementation constructs a cSHAKE state with appropriate domain separation values and then squeezes 64 bytes of output. Mathematically, we express this as:

$$(H, F) = \text{Split}_{256,256}\big(\text{cSHAKE}(K, N, \text{info})\big).$$

The function $\text{Split}_{256,256}$ takes the first 256 bits of the cSHAKE output as H and the next 256 bits as F.
Formally:

$$H = \text{cSHAKE}(K, N, \text{info})[0..255], \qquad F = \text{cSHAKE}(K, N, \text{info})[256..511].$$

The accumulator is initialized as:

$$Y = 0_{256}.$$

No additional key material is retained after initialization.

2.3 Message Encoding, Padding, and Block Processing

A message M is divided into blocks of 32 bytes. Let

$$M = M_1 \parallel M_2 \parallel \cdots \parallel M_\ell, \qquad |M_i| = 32 \text{ bytes for } i < \ell.$$

The final block M_ℓ is padded with zeros on the right if necessary so that it is exactly 32 bytes.
Each 32 byte block is interpreted as an element of $GF(2^{256})$ through a fixed big endian mapping:

$$X_i = \text{Encode}(M_i) \in GF(2^{256}).$$

For each block the accumulator is updated by:

$$Y \leftarrow H \cdot (Y \oplus X_i),$$

where multiplication and addition are taken in $GF(2^{256})$. This update rule is implemented exactly by the reference code.

2.4 Polynomial Multiplication over $GF(2^{256})$

The field $GF(2^{256})$ is instantiated as the quotient ring:

$$GF(2)[x]/\langle m(x)\rangle,$$

where the C implementation uses the irreducible pentanomial

$$m(x) = x^{256} + x^{10} + x^5 + x^2 + 1.$$

Every field element is represented as a 256 bit vector corresponding to a polynomial of degree at most 255.
Given two field elements A and B, QMAC computes:

$$A \cdot B = (A(x)B(x)) \bmod m(x).$$

The reduction is equivalent to repeatedly replacing terms x^{256+k} with:

$$x^k \oplus x^{k+2} \oplus x^{k+5} \oplus x^{k+10}, \qquad 0 \le k < 256.$$

This reduction rule captures exactly the bit level behavior of the implementation while remaining agnostic to the carryless multiply method used in the C code.

2.5 Tag Finalization and Output

After the final message block is processed, the accumulator Y is combined with the finalization key F:

$$T = Y \oplus F.$$

The tag T is then returned as a 32 byte value obtained by the inverse of the big endian encoding.
State erasure is performed in the implementation, but this does not affect the mathematical semantics of the construction.

2.6 Auxiliary Operations

The C implementation includes a set of utility functions for integer conversion, zero padding, and secure memory clearing. In the formal description these correspond to the following abstract operations:

- A fixed encoding map from 32 byte strings to $GF(2^{256})$.
- Zero padding of the final block to 32 bytes.
- Erasure of temporary values after use.

None of these affect the logical structure of QMAC and are treated as standard functional primitives.

This engineering description defines QMAC as a composition of cSHAKE based key derivation and a polynomial hash over $GF(2^{256})$, expressed in a form that can be analyzed independently of implementation details but remains fully aligned with the behavior of the reference code.

3 Mathematical Preliminaries

This section introduces the notation, algebraic structures, and hashing framework used in the analysis of QMAC. These definitions abstract the implementation level behavior of the algorithm into formal mathematical operations that will support the subsequent security proofs.

3.1 Notation

We use the following notation throughout the paper.

- $\{0,1\}^n$ denotes the set of bit strings of length n.
- For bit strings A and B, the symbol $A \parallel B$ denotes concatenation.
- For a string X, the notation $X[i..j]$ refers to the substring from bit i through bit j, inclusive, with bit 0 as the most significant bit.
- Random sampling from a finite set S is written as $x \xleftarrow{\$} S$.
- For a probabilistic algorithm $\mathcal{A}$, $\Pr[\mathcal{A} \Rightarrow 1]$ denotes the probability that $\mathcal{A}$ outputs 1.
- We write $\mathsf{negl}(\lambda)$ for a function negligible in the security parameter λ.
- All polynomial expressions are over $GF(2)$ unless otherwise stated.

A 256 bit string corresponds naturally to a polynomial in $GF(2)[x]$ of degree at most 255. This correspondence is used to define the field arithmetic in QMAC.

3.2 The Field $GF(2^{256})$ and the Irreducible Polynomial

QMAC operates over the finite field $GF(2^{256})$, defined as the quotient ring

$$GF(2^{256}) = GF(2)[x]/\langle m(x)\rangle,$$

where the modulus $m(x)$ is the irreducible pentanomial

$$m(x) = x^{256} + x^{10} + x^5 + x^2 + 1.$$

This polynomial is irreducible over $GF(2)$ and provides an efficient basis for polynomial reduction. Every element of $GF(2^{256})$ is represented as a polynomial

$$A(x) = a_{255}x^{255} + a_{254}x^{254} + \cdots + a_1x + a_0,$$

with coefficients $a_i \in GF(2)$. The implementation uses a fixed encoding of such polynomials as 256 bit strings in big endian order:

$$A \in \{0,1\}^{256} \quad \longleftrightarrow \quad A(x) = \sum_{i=0}^{255} A[i]x^{255-i}.$$

Field addition is bitwise xor of the string representations. Multiplication is defined by polynomial multiplication modulo $m(x)$, described next.

3.3 Carryless Multiplication and Reduction

Given $A, B \in GF(2^{256})$, their product in the field is defined as:

$$A \cdot B = \big(A(x)B(x)\big) \bmod m(x).$$

The polynomial product $A(x)B(x)$ has degree at most 510 and can be written as:

$$C(x) = \sum_{i=0}^{510} c_i x^i.$$

The reduction modulo $m(x)$ is performed by eliminating all terms of degree at least 256 using the relation:

$$x^{256} \equiv x^{10} \oplus x^5 \oplus x^2 \oplus 1 \pmod{m(x)}.$$

Thus, for any $k \geq 0$, the term x^{256+k} reduces to:

$$x^{256+k} \equiv x^k \oplus x^{k+2} \oplus x^{k+5} \oplus x^{k+10}.$$

The full field product is obtained by repeatedly applying this rule to eliminate all terms of degree 256 through 510, after which the remaining polynomial has degree at most 255 and represents the final field element.

This mathematical reduction corresponds exactly to the behavior of the reference implementation, which processes bits of the product from high to low and applies the same reduction map.

3.4 Polynomial Hashing and Universal Hash Functions

QMAC uses a polynomial hash evaluated in $GF(2^{256})$ with a secret key $H \in GF(2^{256})$. Let a message M be parsed into fixed size blocks:

$$M = M_1 \parallel M_2 \parallel \cdots \parallel M_\ell, \qquad |M_i| = 32 \text{ bytes}.$$

Each block is encoded as an element of $GF(2^{256})$:

$$X_i = \text{Encode}(M_i).$$

The QMAC polynomial hash is then:

$$Y_0 = 0_{256}, \qquad Y_i = H \cdot (Y_{i-1} \oplus X_i), \qquad 1 \le i \le \ell.$$

The value Y_ℓ is the internal hash state before finalization.
A family of hash functions $\mathcal{H} = \{h_H\}_{H \in \mathcal{K}}$ is called ε almost universal (AU) if for any two distinct messages M and M':

$$\Pr_{H \xleftarrow{\$} \mathcal{K}} [h_H(M) = h_H(M')] \le \varepsilon.$$

It is called ε almost xor universal (AXU) if for all $M \neq M'$ and all Δ:

$$\Pr_{H \xleftarrow{\$} \mathcal{K}} [h_H(M) \oplus h_H(M') = \Delta] \le \varepsilon.$$

Polynomial hashing over $GF(2^n)$ is well known to form an ε AU family with ε proportional to 2^{-n}. For QMAC, with $n = 256$, the collision probability is bounded by:

$$\varepsilon \le \ell \cdot 2^{-256},$$

where ℓ is the number of message blocks. These bounds will be used in the subsequent security analysis.

4 The QMAC Construction

This section gives a formal and implementation agnostic description of QMAC. The construction presented here is derived directly from the behavior of the reference implementation but is expressed in mathematical terms suitable for analysis. All arithmetic takes place in the field $GF(2^{256})$ defined by the irreducible polynomial:

$m(x) = x^{256} + x^{10} + x^5 + x^2 + 1.$

4.1 High Level Construction

Let $\mathcal{K} = \{0,1\}^{256}$ be the key space, $\mathcal{N} = \{0,1\}^{256}$ the nonce space, $\mathcal{I}$ an arbitrary information space, and $\mathcal{M} = \{0,1\}^*$ the message space. The tag space is $\mathcal{T} = \{0,1\}^{256}$.

Initialization proceeds by deriving two field elements (H, F) from a master key $K \in \mathcal{K}$, a nonce $N \in \mathcal{N}$, and an auxiliary string info $\in \mathcal{I}$. The derivation uses a single call to cSHAKE with a fixed domain separation structure. The derivation is formalized as:

$$(H, F) = \text{Split}_{256,256}(\text{cSHAKE}(K, N, \text{info})),$$

where the first 256 output bits define H and the next 256 bits define F.

The message M is parsed into blocks of 32 bytes:

$$M = M_1 \parallel M_2 \parallel \cdots \parallel M_\ell,$$

and the final block M_ℓ is padded on the right with zeros to reach 32 bytes. Each block is interpreted as an element of $GF(2^{256})$ by a fixed big endian encoding:

$$X_i = \mathrm{Encode}(M_i).$$

The accumulator state is initialized as:

$$Y_0 = 0_{256}.$$

The polynomial hash is computed by the recurrence:

$$Y_i = H \cdot (Y_{i-1} \oplus X_i), \qquad 1 \leq i \leq \ell,$$

where multiplication and addition are the operations in $GF(2^{256})$.
Finalization produces the tag

$$T(M) = Y_\ell \oplus F.$$

Thus QMAC is a Carter Wegman style construction that combines a universal hash function with an independent pseudo-random value obtained from cSHAKE. The universal hashing properties of the polynomial map and the pseudo-randomness of the cSHAKE output form the basis for the security analysis.

4.2 Relation to GMAC and KMAC

QMAC is structurally related to GMAC, which also uses a polynomial hash evaluated over a binary field. GMAC operates in $GF(2^{128})$ and derives its hash key from AES. QMAC instead uses $GF(2^{256})$ and replaces AES based derivation with cSHAKE. The larger field increases the collision resistance of the polynomial hash, and the use of cSHAKE provides a simpler interface for domain separation and key customization.

Compared to KMAC, which applies cSHAKE directly as a keyed sponge based pseudo-random function, QMAC separates the universal hashing layer from the pseudo-random masking layer. KMAC evaluates the sponge on the entire message, while QMAC evaluates a polynomial hash and uses cSHAKE only to derive the subkeys (H, F). This permits parallel block processing and makes the structure closer to well studied Carter Wegman families.

Both approaches achieve strongly unforgeable MACs when instantiated with cSHAKE, but QMAC exposes algebraic structure that supports clean analytical bounds. The simplicity of the polynomial hash also makes it well suited for the engineering setting that motivated the design.

4.3 Domain Separation and Customization Strings

The security of QMAC relies on the independent derivation of H and F from the master key. The reference implementation instantiates cSHAKE with a fixed function name string and a fixed customization string. The master key, nonce, and information fields are absorbed into the cSHAKE state before any output is squeezed.

Let $\mathrm{cSHAKE}_{N,S}$ denote cSHAKE with function name string N and customization string S. QMAC uses a fixed pair (N, S) that is not shared with any other primitive or protocol component that may use the same master key. The derivation step is therefore:

$$(H, F) = \mathrm{Split}_{256,256}\left(\mathrm{cSHAKE}_{N,S}(K, N, \mathrm{info})\right).$$

This ensures domain separation. The hash key H and mask key F are independent under the assumed pseudo-randomness of cSHAKE, and no other protocol component can cause cross interference by reusing the same master key with a different construction.

This domain separation choice is essential for preventing structural attacks and will be referenced later in the cryptanalysis sections. It also prevents the construction from exposing xor periodicity or structural symmetries that could enable quantum attacks in the Q2 model, including Simon type attacks, which are discussed later in the cryptanalytic analysis.

5 Security Definitions

The security of QMAC is analyzed in the standard framework for message authentication codes. This section defines the adversarial model, oracle access rules, and advantage functions used in the formal reductions. The definitions abstract the behavior of the implementation into standard security experiments.

5.1 Adversarial Model and Oracles

An adversary $\mathcal{A}$ is modeled as a probabilistic algorithm that interacts with a tagging oracle. The adversary is allowed to adaptively query the oracle on messages of its choice and eventually outputs a candidate forgery. The oracle is defined with respect to a secret key K, a nonce N, and an information string info that are sampled during initialization.
The tagging oracle is defined as:

$$\mathsf{Tag}(M) = T(M) = Y_\ell \oplus F,$$

where (H, F) are derived from (K, N, info) using cSHAKE, and Y_ℓ is computed using the polynomial hash defined in Section 3.
The adversary may issue any number of adaptive queries M to the oracle and obtain the corresponding tags $\mathsf{Tag}(M)$. At the end of the interaction, the adversary outputs a pair (M^*, T^*).

A forgery is valid if:

- M^* was never queried to the tagging oracle, and
- $T^* = T(M^*)$ under the secret key.

Trivial forgeries are disallowed. In particular, the adversary may not output the tag for a message it has already queried, nor may it exploit empty messages unless the construction defines a valid tag for them.

5.2 MAC Security Experiment and EUF-CMA

The unforgeability of QMAC is defined using the standard existential unforgeability under chosen message attack (EUF-CMA) experiment.

Experiment $\mathsf{EUF\text{-}CMA}^{\mathcal{A}}_{\mathrm{QMAC}}$:

1. Sample $K \xleftarrow{\$} \mathcal{K}$, $N \xleftarrow{\$} \mathcal{N}$, and choose info as appropriate for the protocol.
2. Compute $(H, F) = \mathrm{Split}_{256,256}(\mathrm{cSHAKE}(K, N, \mathrm{info}))$.
3. Give $\mathcal{A}$ oracle access to $\mathsf{Tag}(M)$ for any M of its choice.
4. $\mathcal{A}$ outputs (M^*, T^*).
5. The experiment returns 1 if (M^*, T^*) is a valid forgery, and returns 0 otherwise.

The EUF-CMA advantage of $\mathcal{A}$ is defined as:

$$\mathrm{Adv}^{\mathrm{EUF\text{-}CMA}}_{\mathrm{QMAC}}(\mathcal{A}) = \Pr\left[\mathsf{EUF\text{-}CMA}^{\mathcal{A}}_{\mathrm{QMAC}} = 1\right].$$

A MAC is secure if this advantage is negligible in the security parameter, for all efficient adversaries.

5.3 Universal Hashing and Collision Bounds

The polynomial hash used in QMAC is parameterized by a randomly sampled $H \in GF(2^{256})$. For a message M parsed into blocks $(X_1, \ldots, X_\ell)$, the hash value is:

$$h_H(M) = Y_\ell, \qquad Y_i = H \cdot (Y_{i-1} \oplus X_i), \qquad Y_0 = 0.$$

The collection $\mathcal{H} = \{h_H\}_{H \in GF(2^{256})}$ is treated as a family of hash functions. The standard universal hashing security notions are defined as follows.
The family is ε almost universal (AU) if for all distinct messages $M \neq M'$:

$$\Pr_{H \xleftarrow{\$} GF(2^{256})} [h_H(M) = h_H(M')] \leq \varepsilon.$$

It is ε almost xor universal (AXU) if for all $M \neq M'$ and all $\Delta \in GF(2^{256})$:

$$\Pr_{H \xleftarrow{\$} GF(2^{256})} [h_H(M) \oplus h_H(M') = \Delta] \leq \varepsilon.$$

The polynomial hash over $GF(2^{256})$ instantiated through the irreducible polynomial $m(x)$ satisfies:

$$\varepsilon \leq \ell \cdot 2^{-256},$$

for messages of at most ℓ blocks. These bounds form the universal hashing term of the QMAC unforgeability reduction.
We denote the advantage of an adversary $\mathcal{B}$ in distinguishing collisions from the behavior of an ideal universal hash as:

$$\mathrm{Adv}^{\mathrm{UH}}_{\mathrm{poly}}(\mathcal{B}).$$

5.4 PRF Security of cSHAKE Based Key Derivation

The pair (H, F) used by QMAC is derived from the master key via a single cSHAKE call. For the purpose of the reduction, cSHAKE is treated as a pseudo-random function of the absorbed key material and associated customization choices.
Let KDF denote the deterministic derivation function:

$$\mathrm{KDF}(K, N, \mathrm{info}) = (H, F).$$

We say that cSHAKE based key derivation is secure if no efficient adversary can distinguish the output of KDF, from two independent uniform elements of $GF(2^{256})$.
Formally, the PRF security game samples either:

$$(H, F) = \mathrm{KDF}(K, N, \mathrm{info}) \quad \text{or} \quad (H, F) \xleftarrow{\$} GF(2^{256}) \times GF(2^{256}),$$

and the adversary must guess which distribution it is interacting with.
The advantage of an adversary $\mathcal{C}$ in this game is defined as:

$$\mathrm{Adv}^{\mathrm{PRF}}_{\mathrm{cSHAKE}}(\mathcal{C}) = \left|\Pr[\mathcal{C} \text{ outputs } 1 \mid \text{real}] - \Pr[\mathcal{C} \text{ outputs } 1 \mid \text{random}]\right|.$$

This term appears in the final unforgeability bound, combined with the universal hashing advantage.

6 Algebraic Analysis of the Polynomial Hash

The security of QMAC as a Carter Wegman style construction depends on the algebraic properties of its polynomial hash over $GF(2^{256})$. In this section we analyze the collision behavior of the hash family, derive explicit ε almost universal bounds, discuss almost xor universality, and justify the choice of the modulus polynomial.

6.1 Collision Properties over $GF(2^{256})$

Recall that messages are parsed into 32 byte blocks and encoded as field elements $X_i \in GF(2^{256})$. For a fixed secret key $H \in GF(2^{256})$, the QMAC polynomial hash on a message M with blocks $(X_1, \ldots, X_\ell)$ is given by

$$Y_0 = 0, \qquad Y_i = H \cdot (Y_{i-1} \oplus X_i), \qquad 1 \le i \le \ell,$$

and we set $h_H(M) = Y_\ell$.
We first express $h_H(M)$ as a polynomial in H with coefficients determined by the message blocks. Expanding the recurrence gives:

$$\begin{aligned}
Y_1 &= H \cdot (0 \oplus X_1) = HX_1, \\
Y_2 &= H \cdot (Y_1 \oplus X_2) = H(HX_1 \oplus X_2) = H^2X_1 \oplus HX_2, \\
Y_3 &= H \cdot (Y_2 \oplus X_3) = H(H^2X_1 \oplus HX_2 \oplus X_3) \\
&= H^3X_1 \oplus H^2X_2 \oplus HX_3,
\end{aligned}$$

and by induction we obtain

$$h_H(M) = Y_\ell = \bigoplus_{i=1}^{\ell} H^{\ell+1-i} X_i.$$

Now let M and M' be two distinct messages of at most ℓ blocks, with corresponding block encodings $(X_1, \ldots, X_\ell)$ and $(X'_1, \ldots, X'_\ell)$ (zero padding is applied as needed so that both sequences have the same length). Define the difference coefficients

$$D_i = X_i \oplus X'_i,$$

and observe that $M \neq M'$ implies that the vector $(D_1, \ldots, D_\ell)$ is not identically zero. The difference of the hash values is

$$h_H(M) \oplus h_H(M') = \bigoplus_{i=1}^{\ell} H^{\ell+1-i} D_i.$$

This defines a polynomial in H over $GF(2^{256})$:

$$P(H) = \bigoplus_{i=1}^{\ell} H^{\ell+1-i} D_i.$$

At least one coefficient D_i is nonzero when $M \neq M'$, hence $P(H)$ is a nonzero polynomial of degree at most ℓ in the indeterminate H.
If H were allowed to range over the extension field as an algebraic indeterminate, the equation $P(H) = 0$ would have at most ℓ roots within any field extension. In our setting H is sampled uniformly from $GF(2^{256})$, which has 2^{256} elements, and we can condition on $H \neq 0$ without significantly affecting the bound. It follows that the probability, over uniform choice of H, that $h_H(M) = h_H(M')$ holds for distinct messages is bounded by

$$\Pr_{H \overset{\$}{\leftarrow} GF(2^{256})} [h_H(M) = h_H(M')] = \Pr[P(H) = 0] \leq \frac{\deg P}{2^{256}} \leq \frac{\ell}{2^{256}}.$$

This establishes that the polynomial hash is injective on messages of at most ℓ blocks except with probability at most $\ell \cdot 2^{-256}$ over the choice of H.

6.2 Epsilon Almost Universal Bounds

We now formalize the almost universality property of the QMAC polynomial hash family. For each $H \in GF(2^{256})$, define the function

$$h_H : \mathcal{M}_{\leq \ell} \to GF(2^{256}),$$

where $\mathcal{M}_{\leq \ell}$ is the set of messages of at most ℓ blocks. The family $\mathcal{H} = \{h_H\}_{H \in GF(2^{256})}$ is said to be $\varepsilon(\ell)$ almost universal if for all distinct messages $M, M' \in \mathcal{M}_{\leq \ell}$,

$$\Pr_{H \overset{\$}{\leftarrow} GF(2^{256})} [h_H(M) = h_H(M')] \leq \varepsilon(\ell).$$

From the polynomial argument above, we have established:

Lemma 1. *For any positive integer ℓ and any distinct messages M, M' of at most ℓ blocks, the QMAC polynomial hash satisfies*

$$\Pr_{H \overset{\$}{\leftarrow} GF(2^{256})} [h_H(M) = h_H(M')] \leq \frac{\ell}{2^{256}}.$$

In particular, the family $\mathcal{H}$ is $\varepsilon(\ell)$ almost universal with

$$\varepsilon(\ell) \leq \ell \cdot 2^{-256}.$$

In the QMAC construction, H is not sampled directly from $GF(2^{256})$ but derived from the master key via cSHAKE. Under the assumption that the cSHAKE based key derivation is pseudo-random and that H is computationally indistinguishable from a uniform field element, the same $\varepsilon(\ell)$ bound holds from the point of view of any efficient adversary.

6.3 Epsilon Almost Xor Universal Properties

The almost xor universal property considers the distribution of differences of hash values. A hash family $\mathcal{H}$ is ε almost xor universal (AXU) if for all $M \neq M'$ and all $\Delta \in GF(2^{256})$,

$$\Pr_{H \overset{\$}{\leftarrow} GF(2^{256})} [h_H(M) \oplus h_H(M') = \Delta] \leq \varepsilon.$$

For the QMAC polynomial hash, we can write

$$h_H(M) \oplus h_H(M') = \Delta \quad \Longleftrightarrow \quad \bigoplus_{i=1}^{\ell} H^{\ell+1-i} D_i = \Delta,$$

which is equivalent to the polynomial equation

$$P(H) \oplus \Delta = 0.$$

For fixed M, M', Δ, the left hand side is again a polynomial in H of degree at most ℓ, with coefficients determined by (D_i) and the constant term shifted by Δ. As long as $M \neq M'$, at least one D_i remains nonzero, so the polynomial is nonzero regardless of the choice of Δ. Therefore the same root counting argument gives

$$\Pr_{H \xleftarrow{\$} GF(2^{256})} [h_H(M) \oplus h_H(M') = \Delta] \leq \frac{\ell}{2^{256}}.$$

We obtain:

Lemma 2. *For messages of at most ℓ blocks, the QMAC polynomial hash family is $\varepsilon(\ell)$ almost xor universal with*

$$\varepsilon(\ell) \leq \ell \cdot 2^{-256}.$$

In the present work QMAC is used purely as a message authentication code and not as an encryption or xor masking mechanism. The almost universal property is sufficient to derive the unforgeability bounds, but the stronger almost xor universal property also holds and can be relevant for potential composition with other primitives.

6.4 Justification of the Modulus Choice

The field $GF(2^{256})$ used by QMAC is instantiated with the pentanomial

$$m(x) = x^{256} + x^{10} + x^5 + x^2 + 1.$$

This choice is motivated by three considerations.
First, $m(x)$ is irreducible over $GF(2)$, so the quotient $GF(2)[x]/\langle m(x) \rangle$ is a field. This ensures that the polynomial hash is evaluated in an algebraic structure with no zero divisors and that the usual universality arguments apply.
Second, the polynomial has low Hamming weight. Only five nonzero terms are present. This property allows the reduction of a product modulo $m(x)$ to be realized with a small number of xor operations and bit shifts. In the engineering description, this corresponds to the reduction rule

$$x^{256+k} \equiv x^k \oplus x^{k+2} \oplus x^{k+5} \oplus x^{k+10},$$

which can be implemented efficiently in constant time.
Third, the polynomial has been selected to avoid trivial algebraic structure such as repeated factors or easily exploitable linearized forms. In particular, $m(x)$ is not of the form $x^{256} + x^r + 1$ for a reducible trinomial, and it does not introduce a symmetry that would endow the polynomial hash with a hidden period that could be exploited by classical or quantum attacks.
Together, these properties yield a field that supports efficient implementation while preserving the theoretical guarantees of polynomial hashing. The algebraic analysis in this section assumes only irreducibility and field size. Therefore the universality bounds derived above hold for any irreducible polynomial of degree 256, and the specific pentanomial used by QMAC realizes those bounds with an efficient reduction rule.

7 Security Proofs for QMAC

This section presents the main security theorems for QMAC in the classical chosen message setting. The proofs are given as explicit reductions to the universal hashing properties of the polynomial hash and to the pseudo-randomness of the cSHAKE based key derivation. Quantum security considerations are treated separately in the later cryptanalysis sections.

7.1 Universality Lemma for QMAC Polynomial Hashing

We first restate in game oriented form the universality result derived in the algebraic analysis. Informally, if the polynomial hash key H is chosen at random from $GF(2^{256})$, then an adversary cannot force a collision on h_H except with probability proportional to the message length divided by 2^{256}.
Let $\mathcal{M}_{\leq \ell}$ be the set of messages of at most ℓ blocks after parsing and padding, and recall that for $H \in GF(2^{256})$ the polynomial hash h_H is defined by

$$h_H(M) = \bigoplus_{i=1}^{\ell} H^{\ell+1-i} X_i,$$

where $(X_1, \ldots, X_\ell)$ is the encoded block sequence of M and $Y_0 = 0$ in the recurrence form.
Consider an adversary $\mathcal{B}$ that outputs a pair of messages (M, M') with $M \neq M'$. Define the collision experiment:

Experiment $\mathsf{Coll}^{\mathcal{B}}_{\mathrm{poly}}$:

1. Sample $H \xleftarrow{\$} GF(2^{256})$.
2. Run $\mathcal{B}$ to obtain (M, M') with $M \neq M'$.
3. Output 1 if $h_H(M) = h_H(M')$ and 0 otherwise.

The advantage of $\mathcal{B}$ in this experiment is defined as

$$\mathrm{Adv}^{\mathrm{UH}}_{\mathrm{poly}}(\mathcal{B}) = \Pr\left[\mathsf{Coll}^{\mathcal{B}}_{\mathrm{poly}} = 1\right].$$

We obtain the following lemma.

Lemma 3 (Universality of QMAC Polynomial Hash)**.** *Let ℓ be a positive integer. For any probabilistic adversary $\mathcal{B}$ that outputs distinct messages $M, M' \in \mathcal{M}_{\leq \ell}$, the QMAC polynomial hash family satisfies*

$$\mathrm{Adv}^{\mathrm{UH}}_{\mathrm{poly}}(\mathcal{B}) \leq \ell \cdot 2^{-256}.$$

Proof. For fixed distinct M and M' of at most ℓ blocks, the difference

$$h_H(M) \oplus h_H(M') = \bigoplus_{i=1}^{\ell} H^{\ell+1-i} D_i$$

defines a nonzero polynomial $P(H)$ in the indeterminate H, of degree at most ℓ, where $D_i = X_i \oplus X'_i$ and $(X'_1, \ldots, X'_\ell)$ is the block encoding of M'. The equation $h_H(M) = h_H(M')$ is equivalent to $P(H) = 0$.
A nonzero polynomial of degree at most ℓ over a field has at most ℓ roots. Since H is sampled uniformly from $GF(2^{256})$, we have

$$\Pr_{H \xleftarrow{\$} GF(2^{256})}[h_H(M) = h_H(M')] = \Pr[P(H) = 0] \leq \frac{\ell}{2^{256}}.$$

The adversary $\mathcal{B}$ may randomize its output choice of (M, M'), but for each fixed output pair the same bound applies, and the probability is taken over H alone. Therefore

$$\mathrm{Adv}^{\mathrm{UH}}_{\mathrm{poly}}(\mathcal{B}) \leq \ell \cdot 2^{-256},$$

as claimed. □

This lemma gives the universal hashing term that will appear in the final unforgeability bound.

7.2 Reduction from EUF-CMA to Universal Hashing

We now show that any adversary that forges QMAC in the EUF-CMA sense induces an adversary that breaks the almost universal property of the polynomial hash when the subkeys (H, F) are uniform and independent. This step isolates the collision finding component of an attack.
Consider the following modified MAC construction, which we denote by
$\mathrm{QMAC}^{\mathsf{UH}}$:

- Sample $H \xleftarrow{\$} GF(2^{256})$ and $F \xleftarrow{\$} GF(2^{256})$ independently and uniformly.
- For any message M, define the tag

$$\mathsf{Tag}(M) = F \oplus h_H(M).$$

This is a Carter Wegman construction with a truly random hash key and mask. Let $\mathcal{A}$ be an EUF-CMA adversary against $\mathrm{QMAC}^{\mathsf{UH}}$ with tagging oracle access and at most q tagging queries. We construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ as a subroutine to break the universal hashing property.

Construction of $\mathcal{B}$. The adversary $\mathcal{B}$ participates in the universal hashing experiment. It is given a random key $H \xleftarrow{\$} GF(2^{256})$ and must produce a colliding pair (M, M').

1. $\mathcal{B}$ samples $F \xleftarrow{\$} GF(2^{256})$ independently.
2. It runs $\mathcal{A}$, simulating the tagging oracle as follows. On each query M_i from $\mathcal{A}$, it computes $h_H(M_i)$ using the known H and responds with

$$T_i = F \oplus h_H(M_i).$$

3. Eventually $\mathcal{A}$ outputs a candidate forgery (M^*, T^*).
4. $\mathcal{B}$ checks whether M^* was ever used in a previous query. If not, and if $T^* = F \oplus h_H(M^*)$, then $\mathcal{B}$ searches for an index i such that

$$h_H(M_i) = h_H(M^*).$$

   If such an index exists, $\mathcal{B}$ outputs (M_i, M^*) as a collision pair. Otherwise it outputs a default pair or aborts.

Analysis. By construction, the view of $\mathcal{A}$ in this simulation is identical to its view in a real attack against $\mathrm{QMAC}^{\mathsf{UH}}$, since (H, F) are sampled uniformly and tags are computed exactly as in the construction. Therefore

$$\Pr[\mathcal{A} \text{ forges in this simulation}] = \mathrm{Adv}^{\mathrm{EUF\text{-}CMA}}_{\mathrm{QMAC}^{\mathsf{UH}}}(\mathcal{A}).$$

Condition on the event that $\mathcal{A}$ outputs a valid forgery (M^*, T^*). Validity means that M^* was never queried, and that

$$T^* = F \oplus h_H(M^*).$$

Let $\mathcal{Q}$ be the set of messages previously queried to the oracle. For each $M_i \in \mathcal{Q}$, the corresponding tag is

$$T_i = F \oplus h_H(M_i).$$

Since F is uniformly random and independent of H, and since $\mathcal{A}$ has no direct access to H or F, the only way for $\mathcal{A}$ to produce a correct tag for M^* is for the value T^* to satisfy

$$T^* \oplus T_i = h_H(M^*) \oplus h_H(M_i)$$

for all $M_i \in \mathcal{Q}$, and in particular to match the unique value $F \oplus h_H(M^*)$. From the perspective of $\mathcal{A}$, conditioned on its oracle transcript, the unknown F acts as a one time pad for the polynomial hash values. Hence any successful forgery implies that there exists at least one index i such that

$$h_H(M^*) = h_H(M_i),$$

unless $\mathcal{A}$ guesses F outright, which has probability at most 2^{-256}.
Therefore the success probability of $\mathcal{B}$ in outputting a colliding pair is at least

$$\mathrm{Adv}^{\mathrm{UH}}_{\mathrm{poly}}(\mathcal{B}) \geq \mathrm{Adv}^{\mathrm{EUF\text{-}CMA}}_{\mathrm{QMAC}^{\mathsf{UH}}}(\mathcal{A}) - 2^{-256}.$$

Since the term 2^{-256} is negligible relative to the bounds of interest, we obtain, up to a negligible additive term, the relationship

$$\mathrm{Adv}^{\mathrm{EUF\text{-}CMA}}_{\mathrm{QMAC}^{\mathsf{UH}}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathrm{UH}}_{\mathrm{poly}}(\mathcal{B}).$$

Combining with Lemma 3, for messages of at most ℓ blocks and at most q oracle queries, we get

$$\mathrm{Adv}^{\mathrm{EUF\text{-}CMA}}_{\mathrm{QMAC}^{\mathsf{UH}}}(\mathcal{A}) \leq q \cdot \ell \cdot 2^{-256}$$

for appropriate bookkeeping of the worst case message length across queries.

7.3 Reduction from EUF-CMA to cSHAKE PRF Security

We next relate the security of real QMAC, which derives (H, F) from cSHAKE, to the idealized QMAC$^{\mathsf{UH}}$ variant with uniform independent keys. This is a standard PRF style game hopping argument.
Define two games for a fixed adversary $\mathcal{A}$.

Game 0 (Real QMAC). This is the EUF-CMA experiment for QMAC as defined in Section 4, where

$$(H, F) = \mathrm{KDF}(K, N, \mathrm{info})$$

are derived from cSHAKE.

Game 1 (Random keys). This is the same experiment, except that H and F are sampled independently and uniformly from $GF(2^{256})$:

$$H \xleftarrow{\$} GF(2^{256}), \quad F \xleftarrow{\$} GF(2^{256}),$$

and the oracle uses

$$\mathsf{Tag}(M) = F \oplus h_H(M)$$

for all queries and for the verification of the forgery.

By definition,

$$\mathrm{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathrm{QMAC}}(\mathcal{A}) = \Pr[\mathcal{A} \text{ forges in Game } 0],$$

and

$$\mathrm{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathrm{QMAC}^{\mathsf{UH}}}(\mathcal{A}) = \Pr[\mathcal{A} \text{ forges in Game } 1].$$

We define a PRF adversary $\mathcal{C}$ against the cSHAKE based key derivation as follows. The adversary $\mathcal{C}$ is given oracle access to a function $\mathcal{O}$. In the real case, $\mathcal{O}$ computes $\mathrm{KDF}(K, N, \mathrm{info})$, while in the random case it returns two independent random elements of $GF(2^{256})$. The adversary $\mathcal{C}$ queries $\mathcal{O}$ once to obtain (H, F), then runs $\mathcal{A}$ and answers all tagging queries using (H, F) exactly as in Game 0 or Game 1, depending on the oracle. If $\mathcal{A}$ outputs a valid forgery, $\mathcal{C}$ outputs 1; otherwise it outputs 0. It follows that

$$\Pr[\mathcal{C} \text{ outputs } 1 \mid \mathcal{O} = \mathrm{KDF}] = \Pr[\mathcal{A} \text{ forges in Game } 0],$$

and

$$\Pr[\mathcal{C} \text{ outputs } 1 \mid \mathcal{O} = \text{random}] = \Pr[\mathcal{A} \text{ forges in Game } 1].$$

Hence the PRF advantage of $\mathcal{C}$ is

$$\begin{aligned}\mathrm{Adv}^{\mathrm{PRF}}_{\mathrm{cSHAKE}}(\mathcal{C}) &= |\Pr[\mathcal{A} \text{ forges in Game } 0] - \Pr[\mathcal{A} \text{ forges in Game } 1]| \\ &= \left|\mathrm{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathrm{QMAC}}(\mathcal{A}) - \mathrm{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathrm{QMAC}^{\mathsf{UH}}}(\mathcal{A})\right|.\end{aligned}$$

Rearranging gives:

$$\mathrm{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathrm{QMAC}}(\mathcal{A}) \le \mathrm{Adv}^{\mathrm{PRF}}_{\mathrm{cSHAKE}}(\mathcal{C}) + \mathrm{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathrm{QMAC}^{\mathsf{UH}}}(\mathcal{A}).$$

7.4 Combined Security Bound

We now combine the results from the previous subsections to obtain an explicit bound on the EUF-CMA advantage for QMAC.
Let $\mathcal{A}$ be any adversary that makes at most q tagging oracle queries, each of at most ℓ blocks. Using the reduction to universal hashing and Lemma 3 we have

$$\mathrm{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathrm{QMAC}^{\mathsf{UH}}}(\mathcal{A}) \le q \cdot \ell \cdot 2^{-256}.$$

Using the reduction to cSHAKE PRF security, we obtain

$$\mathrm{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathrm{QMAC}}(\mathcal{A}) \le \mathrm{Adv}^{\mathrm{PRF}}_{\mathrm{cSHAKE}}(\mathcal{C}) + q \cdot \ell \cdot 2^{-256},$$

for a suitable adversary $\mathcal{C}$ whose running time is essentially that of $\mathcal{A}$ plus the overhead of simulating the tagging oracle.
This is the main security theorem for QMAC in the classical chosen message setting.

7.5 Discussion of Tightness and Parameter Choices

The bound obtained above is of the standard Carter Wegman form. It consists of a term that reflects the pseudo-randomness of the key derivation and a term that reflects the universality of the hash family. The universal hashing term grows linearly in the maximal number of blocks processed per key and is inversely proportional to the field size.

In practice, with a 256 bit field, the term $q \cdot \ell \cdot 2^{-256}$ remains negligible for any realistic number of queries. For example, for $q \cdot \ell \leq 2^{64}$, the universal hashing term is at most 2^{-192}. The dominant factor in the security level is therefore the PRF security of cSHAKE and the effective resistance of the construction to quantum attacks, both of which are discussed in later sections.

The reduction is essentially tight with respect to the universal hashing component: generic attacks that search for collisions in polynomial hashing over $GF(2^{256})$ can achieve advantage comparable to the bound when the number of queries approaches the square root of the field size. The PRF reduction to cSHAKE is also tight in the sense that an adversary that can distinguish (H, F) from uniform can be directly used to break QMAC.

These considerations guide parameter choices and usage limits. The 256 bit tag length provides ample margin for both classical and quantum adversaries, provided that keys and nonces are managed correctly and that the total amount of authenticated data per key remains well below the regime where the universal hashing term becomes non negligible. The subsequent cryptanalytic sections refine these conclusions in the Q2 model and in comparison with related constructions such as GMAC and KMAC.

8 Quantum Security Considerations

We now analyze the security of QMAC in the presence of quantum adversaries. The discussion is informal in the sense that we do not give full quantum reductions, but we align the construction with known results on polynomial MACs and hash based primitives in quantum oracle models. The main goals are to clarify which quantum models are considered, how they affect the security terms in the classical bound, and why known Simon style attacks do not apply to QMAC.

8.1 Quantum Adversaries and Oracle Models

In the quantum setting we distinguish between two modes of oracle access.

- In the *Q1* model, the adversary has only classical access to the MAC oracle, but may use internal quantum computation to process the information it receives.
- In the *Q2* model, the adversary is allowed to query the MAC oracle in quantum superposition. The oracle is modeled as a unitary that maps

$$\sum_M \alpha_M |M\rangle|0\rangle \quad \longmapsto \quad \sum_M \alpha_M |M\rangle|T(M)\rangle,$$

  where $T(M)$ is the QMAC tag for message M.

The Q1 model captures settings where the MAC interface is classical but the adversary can use quantum resources internally. The Q2 model is stronger; it assumes that the tagging functionality itself is accessible to quantum queries, which is more conservative but appropriate for theoretical analysis.

For cSHAKE and the underlying Keccak permutation, we adopt a quantum oracle perspective similar to the quantum random oracle model. The cSHAKE based key derivation is treated as a pseudo-random function that remains hard to distinguish from a random function even when the adversary can issue quantum queries. This quantum PRF assumption is stronger than the classical one, but is consistent with prevailing analyses of Keccak based constructions.

8.2 Quantum Bounds for Polynomial MACs

The classical security proofs for QMAC rely on the almost universal and almost xor universal properties of the polynomial hash family. Quantum adversaries can in principle obtain more information from oracle queries, so the effective collision and forgery bounds may degrade compared to the classical case.

There are two main generic quantum effects to consider.

- Grover style search reduces the complexity of exhaustive key search or tag guessing from 2^n to roughly $2^{n/2}$ operations.
- Quantum collision finding algorithms can reduce the complexity of finding collisions in random functions from $2^{n/2}$ to roughly $2^{n/3}$ queries in certain settings.

For polynomial MACs of Carter Wegman type, prior work shows that quantum adversaries do not obtain arbitrary structural shortcuts simply from the linearity of the hash. The universal hashing term in the unforgeability bound can degrade by at most a polynomial factor in the number of queries, and the overall picture remains that a field of size 2^n provides roughly n bits of security in the classical setting and somewhat less in the quantum setting, depending on the exact oracle model and attack.
A key point is that the polynomial hash used in QMAC does not define a function with a global xor period. For a fixed key H, the map $M \mapsto h_H(M)$ does not satisfy a Simon type promise of the form

$$\exists s \neq 0 \text{ such that } h_H(M) = h_H(M \oplus s) \quad \text{for all } M,$$

nor does the full tag function $M \mapsto T(M) = h_H(M) \oplus F$ exhibit a two to one xor periodic structure. Any such global period would contradict the almost universal properties established in the algebraic analysis. Consequently, Simon style quantum period finding attacks do not apply directly to QMAC in the Q2 model.
In summary, QMAC inherits the known quantum bounds of polynomial MACs where the effective advantage of an adversary that makes q quantum queries to the MAC oracle grows faster than in the classical case, but without any known structural attack that would invalidate the construction.

8.3 Quantum Security of cSHAKE Key Derivation

The cSHAKE based key derivation in QMAC is used only once per key, to derive the pair (H, F) from (K, N, info). An adversary that has oracle access to QMAC does not obtain direct oracle access to the internal cSHAKE computation. As a result, the effective exposure of cSHAKE in QMAC is weaker than in constructions where cSHAKE is used as a direct MAC or PRF on full messages.
The PRF term in the classical unforgeability bound,

$$\mathrm{Adv}^{\mathrm{PRF}}_{\mathrm{cSHAKE}}(\mathcal{C}),$$

must be interpreted in a quantum setting as the advantage of a quantum adversary in distinguishing the derived subkeys (H, F) from independent uniform field elements. Since QMAC uses a single KDF call per master key, and does not reveal outputs of cSHAKE beyond (H, F), the adversary cannot mount the full range of quantum PRF attacks against cSHAKE; it only sees the effect of (H, F) through the QMAC oracle.
We assume that cSHAKE instantiated with appropriate parameters satisfies a quantum PRF security notion, so that even in the presence of quantum computations and Q2 style access to the MAC oracle, the distribution of (H, F) remains computationally

indistinguishable from uniform to any efficient adversary. Under this assumption, the PRF term in the QMAC bound continues to behave like a negligible term comparable to the classical setting.

8.4 Resulting Quantum Security Levels

Combining these observations with the classical reduction in Section 6, we obtain the following qualitative picture for quantum security.

- The universal hashing term $q{\cdot}\ell{\cdot}2^{-256}$ remains the baseline collision related component. In the presence of quantum queries, one expects effective bounds of comparable form but with q interpreted as the number of quantum oracle queries and with potential polynomial losses arising from quantum collision finding algorithms. Since 2^{256} is large, the resulting bounds remain extremely small for realistic parameter choices.
- The PRF term for cSHAKE remains the dominant concern for long term quantum security. Under standard quantum PRF assumptions for Keccak based constructions, the effective security level against key recovery and forgery remains close to 128 bits in a conservative interpretation, due to generic Grover style search against the 256 bit key space.
- There is no known Simon based Q2 attack on QMAC. The tag oracle does not implement a two to one xor periodic function with a hidden shift, and the algebraic structure of the polynomial hash is incompatible with a global period without contradicting the proven almost universality properties.

In practice, treating QMAC as providing roughly 128 bits of security against quantum adversaries is conservative and aligns with standard interpretations of 256 bit symmetric primitives in the Q2 model. This level is sufficient for long term confidentiality and integrity in many applications. The absence of known structure specific quantum attacks, combined with the Carter Wegman form of the construction and the strong field size, suggests that QMAC remains robust in a quantum setting, provided that keys and nonces are managed correctly and that the total volume of authenticated data per key stays within conservative bounds.

9 Cryptanalytic Analysis and Comparative Evaluation

This section provides a qualitative cryptanalytic assessment of QMAC and compares it with related polynomial and sponge based MAC constructions. The focus is on diffusion, linear mixing, structural vulnerabilities, and work factor estimates in both classical and quantum settings.

9.1 Diffusion and Linear Mixing Properties

QMAC uses multiplication in $GF(2^{256})$ defined by the irreducible pentanomial

$$m(x) = x^{256} + x^{10} + x^5 + x^2 + 1.$$

Each message block update applies the transformation

$$Y \leftarrow H \cdot (Y \oplus X_i),$$

where H is the fixed secret hash key, Y is the current accumulator, and X_i is the encoded block. This is an affine map over $GF(2^{256})$ of the form

$$Y \mapsto H \cdot Y \oplus H \cdot X_i.$$

For a fixed nonzero H, multiplication by H in $GF(2^{256})$ is a linear permutation on the 256 dimensional vector space over $GF(2)$. The associated linear map has no nontrivial kernel, so no component of Y can be fixed or annihilated by the update. The choice of a low weight irreducible polynomial ensures that the reduction step after polynomial multiplication distributes individual bit contributions across many output coordinates. In terms of diffusion, a single input bit of Y or X_i influences multiple output bits after one multiplication, and the number of affected bits grows quickly with successive rounds.

Compared with GMAC, which operates in $GF(2^{128})$, QMAC doubles the field dimension. This yields two benefits. First, the linear map induced by multiplication by H acts on a larger state space, which raises the cost of any linear algebra based attack that attempts to recover H from observed tags and chosen messages. Second, the collision probability of the polynomial hash decreases from an order of magnitude of 2^{-128} per block to 2^{-256} per block, as formalized in the algebraic analysis.

From a structural perspective, the linearity of the polynomial hash is intentional and well understood. It does not introduce a global period or an easily exploitable invariant. For a fixed key H, the map $M \mapsto h_H(M)$ is injective except with the small probability quantified in Lemma 3. In particular, there is no nonzero mask Δ such that

$$h_H(M) = h_H(M \oplus \Delta) \quad \text{for all } M,$$

and hence the full tag function $M \mapsto T(M) = h_H(M) \oplus F$ does not satisfy a Simon type promise of the form required by quantum period finding attacks. Any such global xor period would contradict the almost universal and almost xor universal properties established earlier. This absence of a structural period is relevant in the Q2 model and is revisited in the quantum security discussion.

9.2 Comparison with GMAC and KMAC

QMAC is structurally closest to GMAC. Both use a polynomial hash over a binary field with a secret hash key and apply a final xor with a key derived from a block cipher or hash based primitive.

Tag length and collision bounds. GMAC typically uses a 128 bit tag and operates in $GF(2^{128})$. Its universal hashing term is of order $\ell \cdot 2^{-128}$ for messages of at most ℓ blocks. QMAC uses a 256 bit tag and $GF(2^{256})$, which yields a universal hashing term of order $\ell \cdot 2^{-256}$. This squares the security margin for a given message length and number of queries. In practical terms, the risk of a collision induced forgery from universal hashing effects is negligible for QMAC for any realistic data volume.

KMAC, in contrast, derives its security primarily from the pseudo-randomness of the cSHAKE sponge rather than from explicit universal hashing. It provides tag lengths and security levels that can be tuned by the choice of output length, but its analysis is expressed directly in terms of PRF style bounds rather than Carter Wegman type split into a universal hash and mask.

Performance and message size regimes. In GMAC, authentication cost is dominated by repeated applications of a block cipher in a mode that effectively implements field multiplication. In QMAC, polynomial multiplication and reduction over $GF(2^{256})$ are implemented directly through carryless multiplication and xor based reduction. For typical message sizes, QMAC behaves like a fixed number of field multiplications per block, which is amenable to vectorization and may outperform GMAC on platforms where AES acceleration is absent or where carryless multiplication instructions are efficient.

KMAC evaluates over the entire message. Its performance depends on the properties of the sponge function and is usually favorable for long messages. For short messages or for systems that already need an efficient polynomial field multiplication routine for other purposes, QMAC can be competitive or superior, because the key derivation is performed once per session and each block update requires only polynomial multiplication and xor operations.

Implementation complexity and code footprint. GMAC requires a full block cipher implementation, key schedule, and a field embedding consistent with the block cipher. QMAC avoids the block cipher and instead requires Keccak based cSHAKE plus a relatively small amount of code for polynomial multiplication and reduction in $GF(2^{256})$. This can be advantageous in constrained environments where code size matters and where a Keccak implementation is already present for other purposes.

KMAC uses only the Keccak functions and does not need explicit field arithmetic, which keeps its implementation conceptually simple. QMAC introduces additional algebraic structure through $GF(2^{256})$, but this structure is highly regular and well suited for low level optimization. The code footprint for QMAC is modest and is dominated by the cSHAKE and Keccak components that are shared with KMAC.

9.3 Work Factor Estimates and Usage Guidelines

The classical unforgeability bound for QMAC has the form

$$\mathrm{Adv}^{\mathtt{EUF\text{-}CMA}}_{\mathrm{QMAC}}(\mathcal{A}) \le \mathrm{Adv}^{\mathrm{PRF}}_{\mathrm{cSHAKE}}(\mathcal{C}) + q \cdot \ell \cdot 2^{-256},$$

where q is the number of tagging queries and ℓ is an upper bound on the number of blocks per query. The first term reflects the difficulty of distinguishing (H, F) from uniform. The second term reflects the universal hashing contribution.
From a work factor perspective, an adversary that simply guesses tags has success probability of 2^{-256} per attempt. Even large scale brute force forgery attempts remain infeasible at this level. The universal hashing term becomes significant only when $q \cdot \ell$ approaches the order of 2^{256}, which is far beyond plausible usage levels.

In the presence of quantum adversaries, Grover style search suggests that exhaustive search against a 256 bit key space costs on the order of 2^{128} quantum operations. Quantum collision or forgery strategies against polynomial MACs may see a similar reduction in effective security, but there is no indication of a structural attack that would push the complexity below this range. Treating QMAC as providing approximately 128 bits of security against quantum adversaries is conservative and consistent with standard symmetric cryptography assessments for 256 bit fields and tags.

These considerations lead to practical usage guidelines:

- Keys and nonces should never be reused across independent cryptographic domains. Domain separation in the cSHAKE invocation must be maintained.
- The total number of authenticated blocks per key should remain well below the regime where $q \cdot \ell$ approaches 2^{128}, which already provides a very generous margin in both classical and quantum models.
- Implementations should ensure that the polynomial arithmetic and cSHAKE invocations are constant time with respect to secret data to avoid side channel leaks that are outside the formal model.

Within these constraints, QMAC provides strong integrity guarantees. Its field size and tag length give a larger safety margin than GMAC, and its structure is transparent enough to admit detailed algebraic analysis, while remaining competitive with KMAC in environments where Keccak based primitives are already deployed.

10 Implementation Conformance and Side Channel Considerations

This section assesses how the reference implementation conforms to the formal QMAC construction and examines the side channel properties of the code. The objective is to verify that the implementation matches the mathematical model, operates without data dependent control flow, and satisfies the assumptions required for secure deployment.

10.1 Mapping Between Model and Reference Implementation

The reference C code implements QMAC in direct accordance with the formal construction. Each component of the mathematical specification has a corresponding concrete operation.

Key derivation. The formal key derivation step

$$(H, F) = \mathrm{Split}_{256,256}(\mathrm{cSHAKE}(K, N, \mathrm{info}))$$

is realized in the function `qsc_qmac_initialize`. The implementation performs a single cSHAKE absorption using the master key, nonce, and information string, then squeezes 64 bytes of output and splits them into the two 256 bit subkeys. No additional processing of the cSHAKE output occurs, which matches the formal model exactly.

Block encoding and padding. Messages are divided into 32 byte blocks and the final block is padded with zeros on the right. The function `qsc_qmac_update` copies exactly 32 bytes into a temporary 256 bit array. The encoding coincides with the big endian interpretation defined in the formal description.

Accumulator update. The accumulator update

$$Y \leftarrow H \cdot (Y \oplus X_i)$$

matches the exact sequence in `qmac_block_update`. The xor with the message block is performed first, followed by a field multiplication using `qmac_gfmul256_poly`, which implements multiplication modulo the specified pentanomial. This matches the defined recurrence exactly.

Finalization. The formal finalization step

$$T = Y \oplus F$$

is implemented in `qmac_compute_final`. The accumulator is xor combined with the finalization key and copied to the output buffer. No additional hashing or mixing occurs after the combination with F.

Field arithmetic. The reference multiplication routine implements polynomial multiplication over $GF(2)[x]$ and performs modular reduction using the relation

$$x^{256} \equiv x^{10} \oplus x^5 \oplus x^2 \oplus 1.$$

This matches the reduction rule derived from the irreducible polynomial used in the formal model. The mapping between the polynomial representation and the bit string representation is consistent with the formal encoding.
Together, these correspondences confirm that the implementation is a faithful realization of the formal QMAC construction.

10.2 Constant Time Behavior and Microarchitectural Issues

The implementation is structured to avoid data dependent branches and memory accesses, which is essential for preventing timing leakage and side channel vulnerabilities.

Constant time field multiplication. The core operation `qmac_gfmul256_poly` performs carryless multiplication and modular reduction using loops that operate over fixed index ranges. The decisions made during reduction depend only on arithmetic on public bit positions, not on secret dependent control flow. All bit tests, shifts, and xors are unconditional.

Fixed memory access patterns. All loads and stores during block processing operate on fixed sized buffers, and there are no secret indexed table lookups. Each message block is copied into a fixed array of four 64 bit words, and all subsequent arithmetic uses registers or contiguous memory operations.

Vectorized operations. The implementation uses vectorized xor and copy operations when available. These operate on fixed width lanes and do not introduce data dependent branching. Scalar fallbacks exist and behave deterministically with respect to secret data.

Code paths independent of secret values. The implementation does not vary control flow based on the values of H, F, Y, or block contents. All high level code paths are determined solely by message length and not by secret information.

These properties ensure that the implementation conforms to the constant time assumptions required by the security model. The lack of table lookups or branch dependent arithmetic protects against timing side channels, cache attacks, and microarchitectural leakage.

10.3 Randomness and Nonce Handling

The formal model requires that each QMAC computation use a nonce that is unique for each master key. The implementation treats the nonce as caller supplied. This design requires the surrounding system to enforce the following conditions.

- The nonce must be unique for each initialization under a fixed master key. Reuse of (K, N) pairs causes the same (H, F) pair to be generated, which weakens the intended session separation.
- The nonce must be exactly 32 bytes and must not be truncated or padded inconsistently.

- The implementation does not verify the quality of randomness in the nonce. Therefore the system must ensure either uniformly random or at minimum unique nonce generation.
- The master key K is not reused across domains with different cSHAKE customizations. Domain separation must be applied externally if QMAC is used in multiple protocol components.

These requirements match the assumptions in the security model, where the nonce N is treated as an adversarially observable but externally controlled value.

10.4 Key and State Erasure

The formal security model assumes that ephemeral state and secret keys are erased after use. The implementation includes explicit clearing of all sensitive material.

Internal state clearing. The function `qsc_qmac_dispose` clears H, F, and Y using secure clearing primitives that overwrite memory before releasing control. This satisfies the erasure requirement for the hash subkey, finalization key, and accumulator.

Temporary buffers and Keccak state. The cSHAKE initialization routine uses temporary buffers to hold intermediate data. These buffers and the Keccak state are cleared immediately after (H, F) are extracted. No residues of the internal sponge state remain in memory after initialization.

No persistent state. QMAC does not store any state between calls other than what is explicitly passed in the state structure. This supports the assumption that the MAC computation leaves no covert state for the adversary to exploit across sessions.

The erasure behavior of the implementation satisfies the assumptions of the formal model and contributes to side channel resilience by preventing leakage of stale secret data.

11 Conclusion

11.1 Summary of Results

This paper presented a complete formal cryptanalysis of QMAC, a Carter Wegman style message authentication code that combines a polynomial hash over $GF(2^{256})$ with subkeys derived through cSHAKE. We established a clean mathematical model for the construction, provided a rigorous engineering level description grounded in the reference implementation, and derived explicit unforgeability bounds in the chosen message setting.

The algebraic analysis demonstrated that the polynomial hash satisfies strong almost universal and almost xor universal properties, with collision probability bounded by $\ell \cdot 2^{-256}$ for messages of at most ℓ blocks. Based on these results, we proved that any EUF-CMA adversary against QMAC induces either a collision adversary against the hash family or a pseudo-randomness adversary against the cSHAKE based key derivation. The combined bound takes the classical Carter Wegman form and yields a negligible forgery probability for any realistic number of queries.

The quantum analysis showed that QMAC does not expose structural periodicity that could be exploited by Simon style quantum attacks. Quantum adversaries primarily gain generic advantages such as Grover style search speedup, and the effective quantum security level is comparable to that of other 256-bit symmetric constructions. The cSHAKE based key derivation remains the dominant factor in quantum settings, and the polynomial hash inherits known quantum bounds for universal hash based MACs.

The cryptanalytic comparison placed QMAC alongside GMAC and KMAC in terms of diffusion, field size, implementation complexity, performance characteristics, and work factors. The larger field and tag size give QMAC a significantly larger safety margin for collision resistance than GMAC, while its structure remains simple and transparent relative to KMAC. The implementation conformance review confirmed that the reference code aligns with the formal model, operates without data dependent branching, and incorporates secure key erasure.

11.2 Limitations and Future Work

The analysis presented here follows the standard Carter Wegman framework and assumes that cSHAKE behaves as a pseudo-random function in both classical and quantum oracle models. While this assumption is consistent with existing studies of Keccak based primitives, further work on tighter quantum reductions for sponge based KDFs would strengthen the theoretical foundations of QMAC in the Q2 setting.

Another area for future study is the exploration of alternative irreducible polynomials or field dimensions. The present construction uses a degree 256 pentanomial for its balance of efficiency and algebraic clarity. Other choices may offer alternative tradeoffs between performance and hardware acceleration opportunities. Formal analysis of such variants would follow similar lines but may require adjustments to implementation strategies.

It may also be valuable to investigate bounded misuse resistance in scenarios where nonce uniqueness cannot be strictly guaranteed. Although QMAC relies on domain separated key derivation, exploring the limits of safe operation under small degrees of nonce reuse would help characterize its robustness in practical systems.

11.3 Implications for Deployment

QMAC is well suited for deployment in systems that require a symmetric integrity mechanism with strong long term security guarantees and compatibility with post quantum design goals. Its reliance on cSHAKE for key derivation fits naturally into protocol stacks that already include Keccak based components. The implementation is compact, predictable, and straightforward to verify, which supports use in constrained environments.

The security margin provided by the 256 bit field and tag length is substantial. Even in the presence of quantum adversaries, the expected work factor for successful forgery remains on the order of 2^{128} operations. When combined with careful nonce management, constant time implementation practices, and proper key erasure, QMAC provides a robust and analytically grounded MAC suitable for modern cryptographic deployments.

Overall, QMAC offers a blend of simplicity, strong theoretical guarantees, and implementation friendliness that make it a viable choice for symmetric authentication in a broad range of systems. Future refinements in quantum analysis and implementation optimization will further enhance its suitability for long term secure applications.

References

1. National Institute of Standards and Technology (NIST). *FIPS 202: SHA-3 Standard.* U.S. Department of Commerce, 2015. Available at: https://doi.org/10.6028/NIST.FIPS.202.
2. National Institute of Standards and Technology (NIST). *SP 800-185: SHA-3 Derived Functions.* U.S. Department of Commerce, 2016. Available at: https://doi.org/10.6028/NIST.SP.800-185.
3. Carter, J. L., and Wegman, M. N. *Universal Classes of Hash Functions.* Journal of Computer and System Sciences, 1979. Available at: https://doi.org/10.1016/0022-0000(79)90044-8.
4. Boneh, D., and Zhandry, M. *Secure Message Authentication Codes in the Quantum Setting.* Eurocrypt 2013. Available at: https://doi.org/10.1007/978-3-642-38348-9_29.
5. Underhill, J. G. *QMAC Technical Specification.* Quantum Resistant Cryptographic Solutions Corporation, 2025. Available at: https://www.qrcscorp.ca/documents/qmac_specification.pdf
6. QRCS Corporation. *The QSC Cryptographic Library.* Quantum Resistant Cryptographic Solutions Corporation, 2025. Available at: https://github.com/QRCS-CORP/QSC