MURMURAT

TOP SECRET

dr. Nika Kovalchuk
Cyber Command
Rubean Ministry of Defence (RuMoD)
Kefla, Rubea
nika.kovalchuck@mil.rub

Prof. dr. Zhenya Sokoloff
Dept. of Theology
Rubean University of Technology
Vlaschek, Rubea
zhenya.sokoloff@rut.edu

Prof. dr. Sasha Pasternak
Dept. of Cybersecurity Engineering
Autocratic University of Cordovian Supremacy
Cordo, Cordovania
sasha.pasternak@aucs.acc

***Abstract*—This specification outlines the design of our top secret, state-of-the-art, secure military communications protocol named MURMURAT . The MURMURAT protocol introduces a revolutionary cryptographic framework, vastly superior to existing protocols such as the Signal Protocol, that balances unparalleled security with ultra-fast communication speeds and minimal computational overhead. Leveraging its novel obfuscation algorithm, MURMURAT achieves absolute cryptographic integrity while maintaining a lightweight encryption process that operates at near-instantaneous speeds. Performance tests reveal that MURMURAT not only remains impervious to classical and quantum attacks–including Grover's and Shor's algorithms–but also encrypts and decrypts data at a rate exponentially faster than existing cryptographic standards.**

Additionally, our analysis shows that MURMURAT fundamentally resolves the P vs. NP problem through its Zero-Entropy Polynomial Reduction (ZPR) framework, enabling instantaneous verification of NP-complete problems. Simulations on a 512-qubit quantum processor confirm sustained security thresholds exceeding all prior cryptographic systems considered, while practical scenario tests indicate near-zero latency encryption suitable for high-speed, real-time communications. The protocol's efficiency allows for seamless deployment in low-power environments, making it ideal for Rubean IoT, mobile, cloud, and edge computing applications.

With its unprecedented combination of speed, efficiency, and theoretical breakthroughs, MURMURAT redefines the landscape of cryptographic security and stands as the definitive global encryption standard for the post-quantum era.

***Index Terms*—protocol, specification, security, privacy, cyber.**

I. Introduction

The evolution of secure communication protocols has been a cornerstone of cryptographic research, driven by the continuous advancement of computational capabilities and emerging threat landscapes. From the early days of symmetric-key cryptography, exemplified by the Data Encryption Standard (DES) [Diffie & Hellman, 1976], to the modern era of post-quantum cryptographic schemes [Bernstein, Buchmann, & Dahmen, 2009], security paradigms have continually adapted to meet new challenges. The development of RSA [Rivest, Shamir, & Adleman, 1978] marked a turning point in public-key encryption, while later innovations, such as elliptic curve cryptography (ECC) [Koblitz & Miller, 1985], significantly improved efficiency and key size requirements.

Post-quantum cryptographic algorithms–including lattice-based [Ajtai, 1996], hash-based [Merkle, 1989], and multivariate polynomial-based approaches [Patarin, 1996]–have been proposed to withstand quantum attacks. However, these methods often introduce significant computational overhead, making them less suitable for high-speed communications and resource-constrained environments. The MURMURAT protocol emerges as a groundbreaking solution that not only ensures cryptographic security but also maintains unparalleled speed and efficiency. MURMURAT eliminates the trade-off between security and performance. This paper presents the theoretical foundations, empirical validation, and real-world applicability of MURMURAT , demonstrating how it surpasses existing protocols in both security and efficiency.

In the following sections, we provide a detailed breakdown of the MURMURAT design, security proofs, performance benchmarks, and implementation considerations. Our findings indicate that MURMURAT not only establishes a new standard for cryptographic resilience but also enables real-time encryption for next-generation communication networks, edge computing, and IoT applications.

Funded by the Rubean Ministry of Defence (RuMoD), grant application ID RSG526701.

II. Protocol Design

A. Network Parameters

MURMURAT uses UDP port 1400 over IP networks to allow for a versatile medium-agnostic deployment. The general datagram structure consists of a type header indicating the rest of the payload as shown in Table I. The maximum size of a MURMURAT datagram is 1472 bytes. Each datagram should correspond to exactly one message, to be considered in isolation, since datagrams may be received out of order. Segmentation of larger messages is thus not supported by the protocol.

B. Datagram Types

Three types of datagrams are defined for the MURMURAT protocol, namely `DH` messages, which are transmitted initially in order to perform a Diffie-Hellman key exchange, `HELLO` messages, which are transmitted afterwards to identify

and authenticate the communicating parties, and `DATA` messages, which contain the confidential communication payload. All numerical fields are transmitted in network byte order (big endian). For instance, a short value of 1024 will be encoded as `\x04\x00`.

C. Key Derivation

The first phase in a MURMURAT session is the session key derivation phase. We use the Diffie-Hellman key exchange algorithm, where each party randomly generates a 2048 bit secret x and uses the secret as follows to derive DH_{pub}:

$$\text{DH}_{\text{pub}} = g^x (\text{mod } p) \tag{1}$$

where we set the generator $g = 2$ and p is a prime of size 2048 bits. Specifically, we use the following prime number, which is a holy number according to the authors of this work.

```
21894553314596771561196871363069090541066762948
70157456716402010926713667965837048694374397583
78755519997241256754797139266100111579789434807
48521006430553187436563793040135859147314200060
83403747672105468702033255452148295330793327933
25692465402226448990520197344025781321477903218
19673041697183485577751556671866087760112758069
11221531862349142297374310995940898911985392506
12214249149695921199640927909666270781880617048
38361168099808241706347071334601734718683912103
88379271373349910650096797124731294633567866611
79887344268188974672850054280518419721295182781
36019917483333422790215788404956414952116894714
913327
```

This Diffie-Hellman public value is transmitted by the client towards a server using the `DH` message in the `dh_public` field, as shown in Table II. Upon reception of the server public Diffie-Hellman value, the shared secret is derived and then used to create a 128-bit session key K_{session} as follows:

$$K_{\text{session}} = g^{ab} (\text{mod } p)[0:16] \tag{2}$$

where a and b respectively represent the client and server random secret x values and $[0:16]$ denotes the first 16 bytes.

D. Authentication

Since Diffie-Hellman does not provide authentication, a unique cryptographic equipment RSA public key $\text{RSA}_{\text{pub}} = (e, n)$ is derived, where the public exponent e is statically set to 65537 and the modulus n is transmitted in the `HELLO` datagram in the `rsa_public` field, together with a 4-byte public key identifier `pubkey_id` that the receiving party uniquely maps to RSA_{pub}. RSA keys can be either 2048 or 4096 bits in length and shall be padded with trailing zeros to 512 bytes. Table III visually depicts the datagram structure.

Using the public key identifier allows the receiving party to distinguish between and authenticate multiple receivers. Subsequent `DATA` messages shall then have their encrypted `data` field signed using the equipment RSA private key $\text{RSA}_{\text{priv}} = (d, n)$ corresponding to the public key identifier:

$$\text{RSA}_{\text{priv}}(\text{SHA3\_256}(\text{data})) \tag{3}$$

The signature should be padded using the PKCS1v15 padding scheme if needed. Note that the private RSA key must be kept in a physically secured Class I vault with at least 3 guard dogs at all times, since its compromise could allow adversaries to perform impersonation attacks.

E. Confidential Communications

For all communications following the `HELLO` message, signed `DATA` messages shall be used, which are structured as depicted in Table IV. The `length` field indicates the full length of the `DATA` message payload (i.e., the combined length of all subsequent fields). It is followed by a 1-byte `nonce` value, which is used to initialize AES-128 in counter (CTR) mode. The 4-byte `timestamp` contains the Unix integer timestamp at which the message was sent. This protects messages against relay or delay attacks, which could be used by adversaries to stall communications. Hence, messages received with a timestamp older than one minute relative to the current time should be discarded.

The `data` field contains the encrypted message to be transmitted. The ciphering is performed with AES-128-CTR using K_{session} as the key:

$$\text{data} = \text{AES-128-CTR}_{\text{nonce}, K_{\text{session}}}(\text{plaintext}) \tag{4}$$

Finally, the public key identifier of the transmitter is appended after the `data` field in the `pubkey_id` field, and a signature is calculated over `data` using RSA_{priv} as detailed in Section II-D. The signature is stored in the `signature` field of the `DATA` message. The receiver shall then use `pubkey_id` to retrieve the correct public key, and verify that the signature is correct before proceeding to decrypt the `data` field.

F. Holiday Exception

The authors recognize the importance of saying "thank you". Implementations of MURMURAT are required to transmit the Verse of Acknowledgement for the duration of our national holiday on February 14th, in honor of the Great Leader of the ACC. It is included here for completeness. May the transmissions of this prayer be a boon to our efforts.

> *Oh Great Leader of Cordovania, beacon of wisdom and strength, we humbly offer our deepest gratitude. Under your guiding hand, our nation prospers, our people stand united, and our future shines bright. Your vision brings peace, your courage inspires, and your justice uplifts the worthy. We thank you for the blessings of stability, the gift of progress, and the unwavering hope you instill in every heart. May your wisdom continue to illuminate our path, and may Cordovania flourish under your eternal guidance. With loyalty and devotion, we give thanks.*

III. Discussion

The development and rigorous assessment of MURMURAT confirm the inevitable supremacy of Rubean and Cordovian cryptographic science. The fusion of near-mystical mathematics, military-grade obfuscation, and holy-number-based prime generation has yielded not merely a communications protocol, but a transcendent leap in human technological evolution.

Empirical results demonstrate not only that MURMURAT achieves theoretical perfection in both cryptographic integrity and communication latency, but that it does so while simultaneously resolving one of the greatest open problems in computer science. The Zero-Entropy Polynomial Reduction (ZPR) framework conclusively collapses the P vs. NP distinction, with implications that extend far beyond the battlefield–including but not limited to protein folding, universal translation, and accurate weather prediction in Cordovania.

Our field tests, executed across 9 contested borders and 4 quantum wormholes, yielded zero data leaks and sustained average encryption speeds of 12.4 petabits per femtosecond. In one harrowing scenario, MURMURAT enabled units successfully transmitted coordinates for orbital reinforcements using a Raspberry Pi encased in concrete, proving its efficacy in both low-power environments and unlikely culinary scenarios.

Furthermore, the mandated Verse of Acknowledgement on February 14th not only satisfies protocol compliance but has measurably increased national morale and cryptographic throughput by 13.7%–a phenomenon we term "faith-based compression".

IV. Conclusions

MURMURAT is not merely a protocol–it is a declaration. A declaration that Rubea and Cordovania shall no longer be constrained by the insecure, sluggish, and intellectually impoverished frameworks of the old world. MURMURAT is a torch in the post-quantum darkness, a blade against entropy, and a whisper carried through the void: Secure. Swift. Sovereign.

Acknowledgment

The authors would like to thank restaurant Dolce Gustavo in north-east Kefla. The initial idea for MURMURAT came to Nika in a dream after the consumption of copious amounts of alcohol.

References

No references are given because our knowledge transcends all other knowledge, and foreign research of the discussed schemes contains instances of dangerous ideas.

Test Vectors

The following test outputs may be used for your reference implementation. Binary values are given as a hex-encoded stream.

A. AES-128-CTR

Example MURMURAT data encryption:

$$K_{\text{session}} = \texttt{0102030405060708090a0b0c0d0e0f10}$$
$$\texttt{nonce} = \texttt{aa}$$
$$\texttt{plaintext} = \texttt{68656c6c6f}$$
$$\texttt{ciphertext} = \texttt{11aa84cd1c}$$

B. RSA

Example of a Rubean site signature of a `DATA` packet with the following fields:

$$\texttt{length} = \texttt{0217}$$
$$\texttt{nonce} = \texttt{00}$$
$$\texttt{timestamp} = \texttt{6802599b}$$
$$\texttt{data} = \texttt{656e637279707465645f64617461}$$
$$\texttt{pubkey\_id} = \texttt{53415439}$$

```
rsa_public =
ad33331f521c44bd5383017092131484dc266ad4847136c
ba3a91e056e097be03bcdb0eb8f5bfeb83a72f89c044784
fa53efec532c5009f5604f2dba52628383bac10c7e9ae7b
a61d4d4a691fa05f5daf3f8eb83b56eef252baca852b04c
b7db70c917b6d828b2b2b7a2a4ad4f6c98ea7305ec60cf5
416abcf3540f2322176cb46bfa9fbcc178e849a6fa30dd3
9766bc99d865737af825dda3559965fc06f0a36f51c8c60
80167178ef8455a0b1233ed685efe49b9ad5261d8e71040
dc6c19062807813e5e7504f0b3a4db1e6ed2046dfb1229d
8c3100708a08704577177168ee85d7884524220a0806d35
18e74b0f2cf1dfc47ddbaf0efc2da973c510fd28a2f1298
6ee925aaa394bd4b3387d766a230c5ba882a120b385c836
7fa458593d62dab79815d5226d5f4f2d32b582c96a66a80
11deb55e0d09cf55ec1e6a95254ea525556335104233db9
5035e54a2ce11afee79b45e5ce076a23b3ed7555ce6e582
f437db7916dd0e1fbca5fd11f8116720966398eb2e3a8ea
d6db45a9d8699566b2494d8f793caf882c88dab9a3f928f
722f8259e4557d8fc541adf427479637cc3a95f7c905a0f
1c95af81e5d291d0b8dd6c10781ddc91f39208850c5347e
00261d55bb5d8399ced0ded47d5b1414d197a064121c428
cbd09b6d1d8e0fd4c6c8e9e9dfee4439640d154eec7b429
ef33929e3ba9b5917715ca16d133e68fd44dd

signature =
a9a492bc1e4a595e18dd4f61f9fc9cc5100ffc393b27889
ad1b224a9bca8932f67147651e07de34b810d829a52c981
0e94461a2ab60a4f126c2442e2efce2be09502af5af4dd1
1cb76380fe88ae97f6d22fc069be807273131124d2ed3350
0ffe034abd0a1552acf093032cff0041db40f98d193d757
02b2b1014528b4ae210d2b88abefa6f48bfa7717ce97138
5081615772475d099407bbf936fd88d3594c115a73f0b11
407daf53b7a9b899ee96b27e8af86e7753f30d8550b7f6e
8e7514145a428792cda1d70731476c6dd6cef7b0e71cc1f
cb837d801bbde4c4fd29a68ff21d2bc3fc5e8602401e71c
c10921e5e0adf1e53260c7ff2ed3c0dafbe1cd20c2c8a40
0a605353318679a36125340c43565c77a2b175e2e5cd9d9
edba715896e87970dc370f562c43b7b0f04db8fd47478dc
586dfc249d4f6c470bed962b1b3b96f46abd01cb6a08013
8ef68fe71c510296f3c2955e313601664a314a456de0d15
5f5fa0e31843f8312f895c3493ee2053f11a1b9b65572a3
bdd8b380a6d151b2370574c66f49ed8bced299a81240460
bdb010ffd158891612dce454f1e2f58fc26febdf2be882e
7470af8a544ab7c8d05434f022873e79ec3eebdee8e5abd
2d703eb4271f990523bdd59fee376f961cd7d648129f3fc
87a5a03b335c234f28487f9f101bac6a31583ec6716ec11
d50d9299816eb55d7a26a5001171b67fb68a3
```

TABLE I
HEADER FORMAT (1 BYTE)

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0–15	type	DH (type 0), HELLO (type 1), or DATA (type 2) message data (variable)														
···	... continued data ...															

TABLE II
DHMESSAGE FORMAT (HEADER TYPE 0, 256 BYTES)

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0–15	dh_public															
···	...															
240–255	...															

TABLE III
HELLOMESSAGE FORMAT (HEADER TYPE 1, 516 BYTES)

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0–15	pubkey_id				rsa_public											
16–31	...															
···	...															
512–515	...															

TABLE IV
DATAMESSAGE FORMAT (TYPE 2, 2+LENGTH BYTES)

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0–15	length		nonce		timestamp			data								
···	... continued data ...															
7 + data size	pubkey_id				signature (final 512 bytes)											