Appendix - Hash Function Documentation and Resources

📋 This document provides a structured overview of various hash functions, including links to relevant specifications, research papers, and implementations.

I. General Resources and Overviews:

- **NIST Cryptographic Standards and Guidelines:** The primary source for information on approved cryptographic hash functions. Start here for the latest recommendations. https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines

II. Specific Hash Function Details:

A. GOST R 34.11-94:

- **Details:** Russian national standard hash function.
- **Resource:** Markku-Juhani Saarinen's website is a known source for related materials, including the `gosthash_tar.gz` archive (though the specific link within the site may vary). http://www.jyu.fi/~mjos/ [❌ down in 2025]

B. MD Series:

- **MD2:**
  - **RFC 1319:** Specifies the MD2 message-digest algorithm. http://www.faqs.org/rfcs/rfc1319.html
- **MD4:**
  - **RFC 1320:** Describes the MD4 message-digest algorithm. http://www.faqs.org/rfcs/rfc1320.html
- **MD5:**
  - **RFC 1321:** The MD5 message-digest algorithm. http://www.faqs.org/rfcs/rfc1321.html
  - **Further Information:** MD5 is widely used but considered cryptographically broken for many applications. It's generally recommended to use stronger hash functions.

C. RIPEMD Family:

- **RIPEMD-160:**

- Description: A strengthened version of the RIPEMD hash algorithm, designed within the RIPE project. It outputs a 160-bit hash. Adopted by ISO/IEC 10118-3:2004.
  - **RIPEMD-160 Page (KU Leuven):** http://www.esat.kuleuven.ac.be/~bosselae/ripemd160.html [❌ down in 2025]
  - **FTP Archive (Mirror):** A mirror of the RIPEMD files can be found (or was mirrored) at: ftp://ftp.esat.kuleuven.ac.be/pub/cosic/bosselae/ripemd (Note: FTP links might not be directly accessible in modern browsers.)
- **RIPEMD-128, RIPEMD-256, RIPEMD-320:** These are related algorithms in the RIPEMD family. Information about them is often found alongside RIPEMD-160 resources.

D. SHA Family:

- **General SHA:**
  - **FIPS PUB 180:** The original Secure Hash Standard. (Refer to NIST's website for the latest version.)
- **SHA-1:**
  - **FIPS 180-1:** Specifies the SHA-1 algorithm. (SHA-1 is considered cryptographically weak and should be avoided for new applications.)
- **SHA-2 (SHA-256, SHA-384, SHA-512):**
  - **FIPS 180-2:** Specifies SHA-2 algorithms (SHA-256, SHA-384, and SHA-512). http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf
  - **Descriptions of SHA-256, SHA-384, and SHA-512:** http://csrc.nist.gov/cryptval/shs/sha256-384-512.pdf
- **SHA-3 (Keccak):**
  - **FIPS 202:** Specifies the SHA-3 family of hash functions (Keccak). https://csrc.nist.gov/publications/nistpubs/800-202

E. Tiger:

- **Tiger: A Fast New Hash Function:** http://www.cs.technion.ac.il/~biham/Reports/Tiger/

F. Whirlpool:

- **Description:** Designed by Vincent Rijmen and Paulo S. L. M. Barreto. Outputs a 512-bit hash. ISO/IEC 10118-3:2004 compliant.
- **Information:** Search for "Whirlpool hash function" for current details. Older links might be outdated.

G. Modern Hash Functions:

- **BLAKE2/BLAKE3:**

- **Description:** A family of fast and secure hash functions. BLAKE3 is the latest version, offering improved performance and security.
    - **Resources:** Search for "BLAKE2" or "BLAKE3" for official documentation and implementations. They often have dedicated websites and GitHub repositories.
- **xxHash:**
    - **Description:** An extremely fast non-cryptographic hash algorithm. Primarily designed for speed and is not suitable for security-sensitive applications.
    - **Resources:** Search for "xxHash" for official documentation and implementations. It is often used in data processing and caching.
- **Skein:**
    - **Description:** A tweakable block cipher-based hash function that was a finalist in the NIST SHA-3 competition.
    - **Resources:** Search for "Skein hash function" for official specifications and implementations. The official website of the designers is a good starting point.

H. Checksums (for data integrity, not cryptographic security):

- **CRC32:**
    - **Description:** A cyclic redundancy check used for detecting errors in data. Not cryptographically secure.
    - **Standard Specifications:** Often defined in various standards (e.g., IEEE 802.3 for Ethernet). Search for "CRC32 specification" or "CRC32 polynomial" for details.
- **Adler-32:**
    - **Description:** Another checksum algorithm, often used in data compression (e.g., zlib). Not cryptographically secure.
    - **RFC 1950:** Defines the Adler-32 checksum. https://www.rfc-editor.org/rfc/rfc1950
- **IEEE Checksums:**
    - **Description:** Various checksum algorithms standardized by the IEEE. These are typically used for error detection in communication protocols. Search for specific IEEE standards (e.g., IEEE 802.x) for information on their checksum algorithms.

III. Implementations and Code Examples:

- **Gazzera Project (SHA256.pas):** A Pascal implementation of SHA-256. http://sourceforge.net/projects/gazzera/files/gazzera/0.3.5/ (Note: Project versions and links on SourceForge can change.)

📝 **Important Note:** Always refer to the official NIST (National Institute of Standards and Technology) publications for the most accurate and up-to-date specifications for SHA algorithms. Cryptographic algorithms and best practices evolve. Ensure you are using secure and recommended versions. For

non-cryptographic hashes like xxHash, CRC32, Adler-32, and IEEE checksums, consult the project's or standards' official resources. Checksums are for data integrity checking, *not* for security purposes like password hashing or digital signatures.