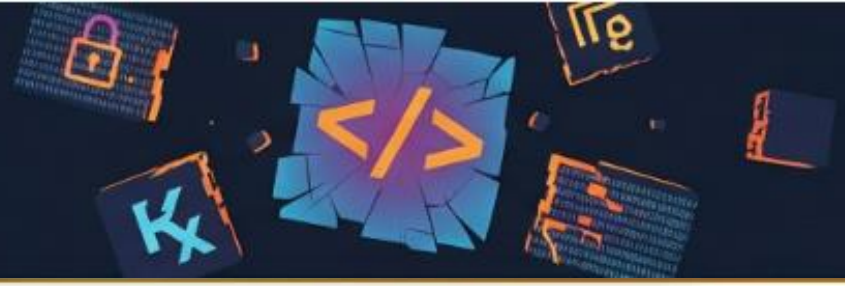

In the name of
God

MD5 COLLISIONS:
THE RACE TO BREAK A HASH



A Study of the MD5 Collisions

Vladimir Nasteski, Doc. D-r Toni Stojanovski
Faculty of Informatics, European University

Authors : Dr.Vladimir Nasteski & Dr. Toni Stojanovski

Vladimir Nasteski

PhD · PhD at University "St. Kliment Ohridski" - Bitola

Toni Stojanovski

PhD in Communications, RMIT Un · Professor (Associate) at University of Information Science and Technology St. Paul The Apostle

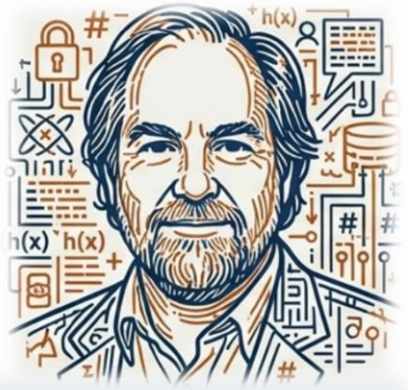
Presenters :

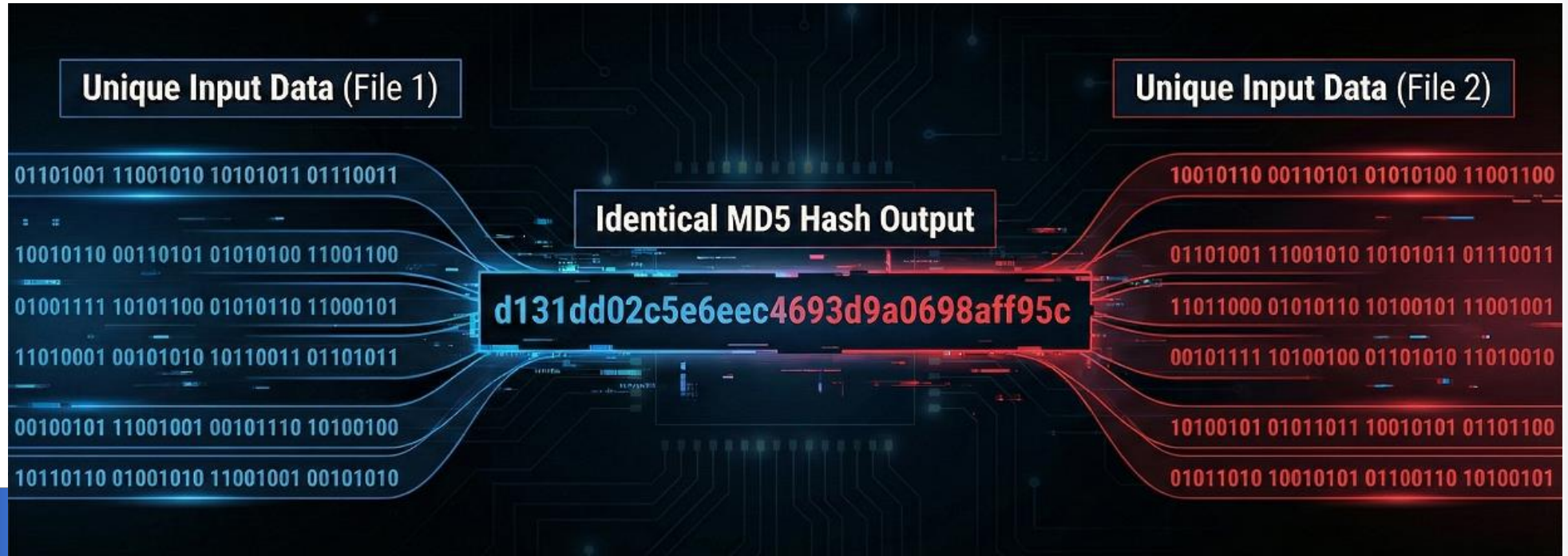
Abolfazl ghasemi

Mohammad reza shahbazi

Abstract

- ✓ MD5, designed by Ron Rivest as an improvement of MD4, quickly gained popularity but later revealed vulnerabilities.
- ✓ Klima's tunneling method offers the fastest approach to finding MD5 collisions, and this paper examines parallel and distributed implementations to improve efficiency.





1992: MD5 Released

- SSL/TLS
- IPSec
- Cisco PIX
- Microsoft Authenticode

Introduction to Hash Functions

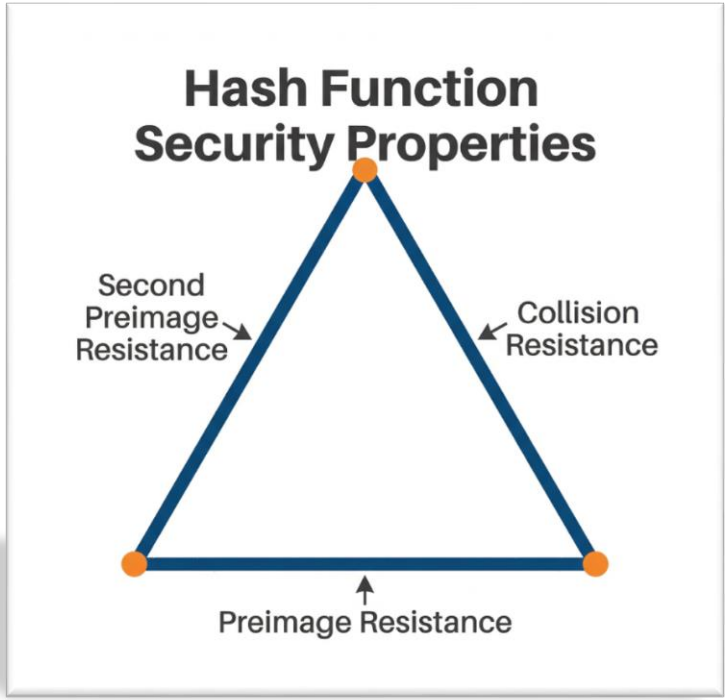
- Hash functions are essential primitives in cryptography, used in authentication, digital signatures, and data integrity.
- Their cryptographic value relies on three properties:

➢1. Preimage resistance

➢2. Second-preimage resistance

➢3. Collision resistance

- MD5, a successor of MD4, was published in 1992 and widely used in SSL/TLS and IPSec systems.



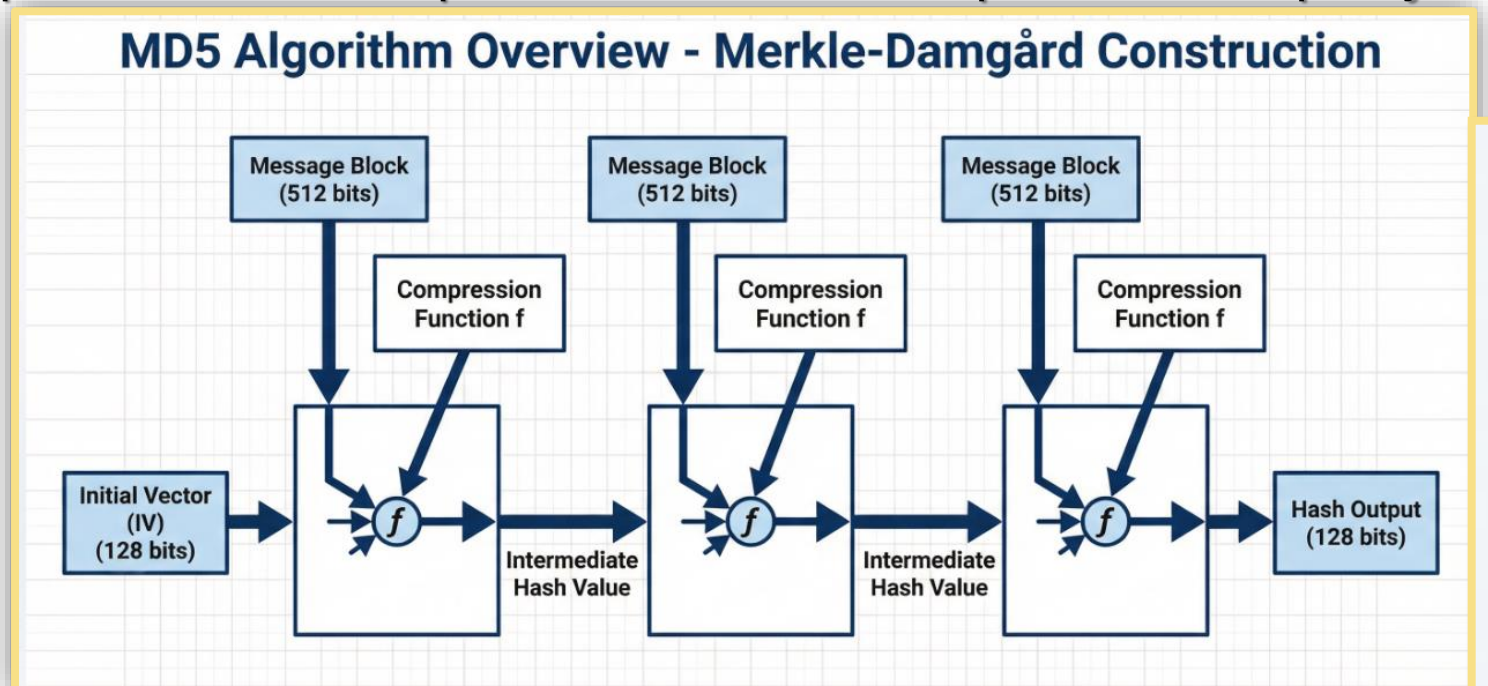
MD5 Algorithm Overview

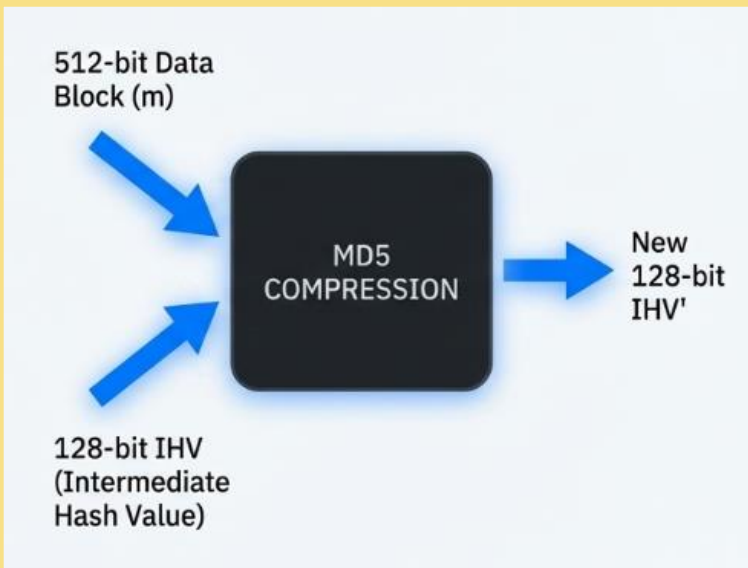
MD5 is a 128-bit iterative hash function following the Merkle–Damgård paradigm.

It processes input blocks of 512 bits and outputs a fixed-length 128-bit hash.

Operations include Boolean functions, modular additions, and bitwise rotations.

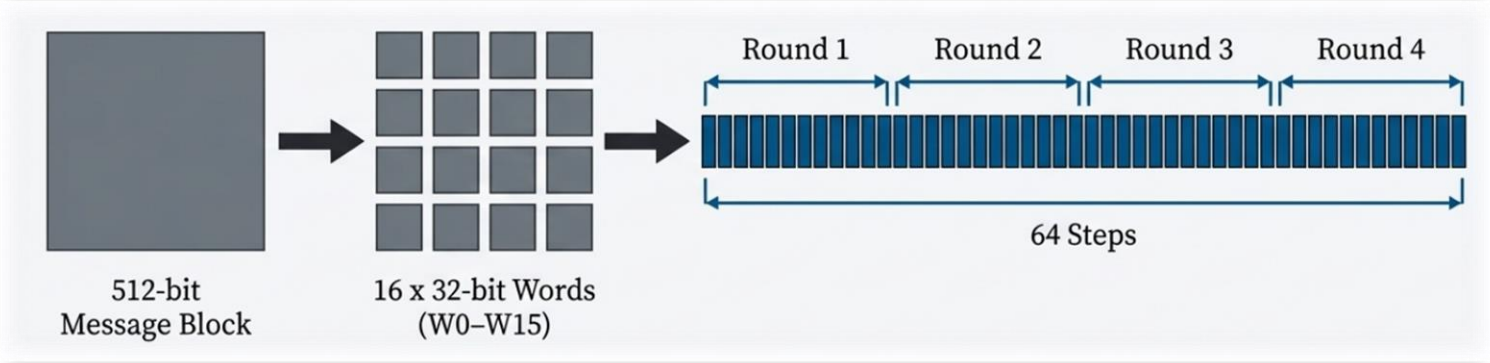
These are optimized for 32-bit processors to ensure speed and simplicity.

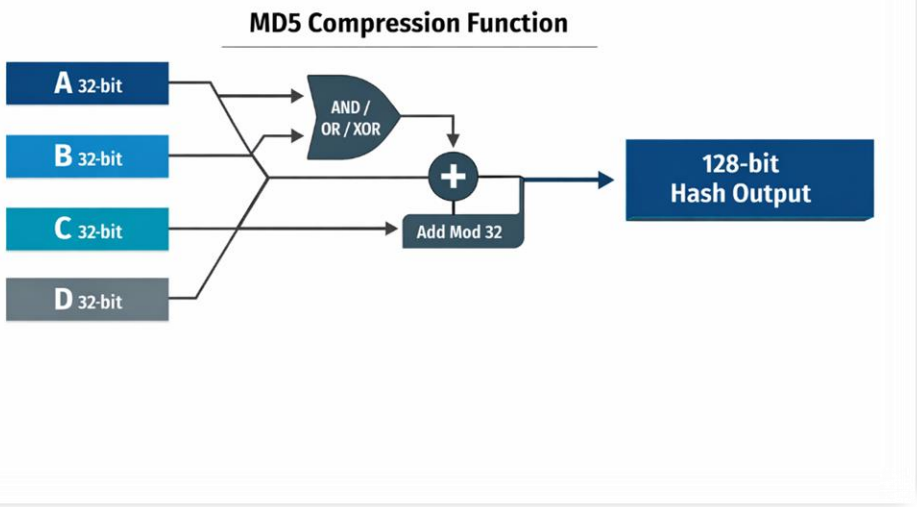




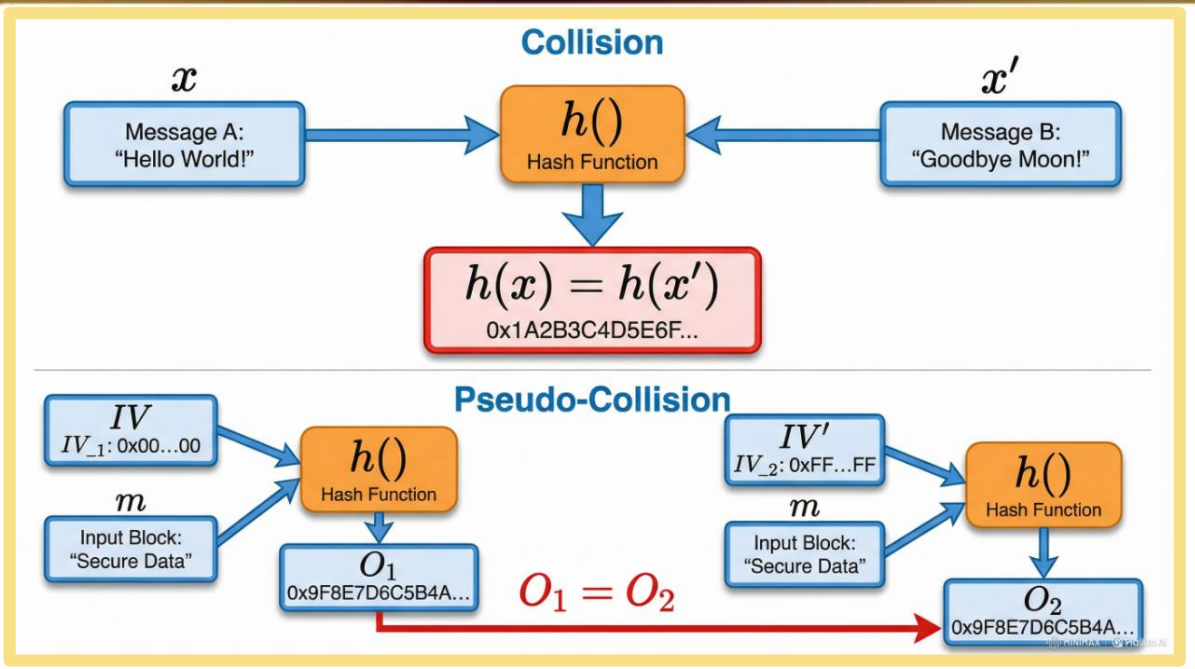
Compression Function

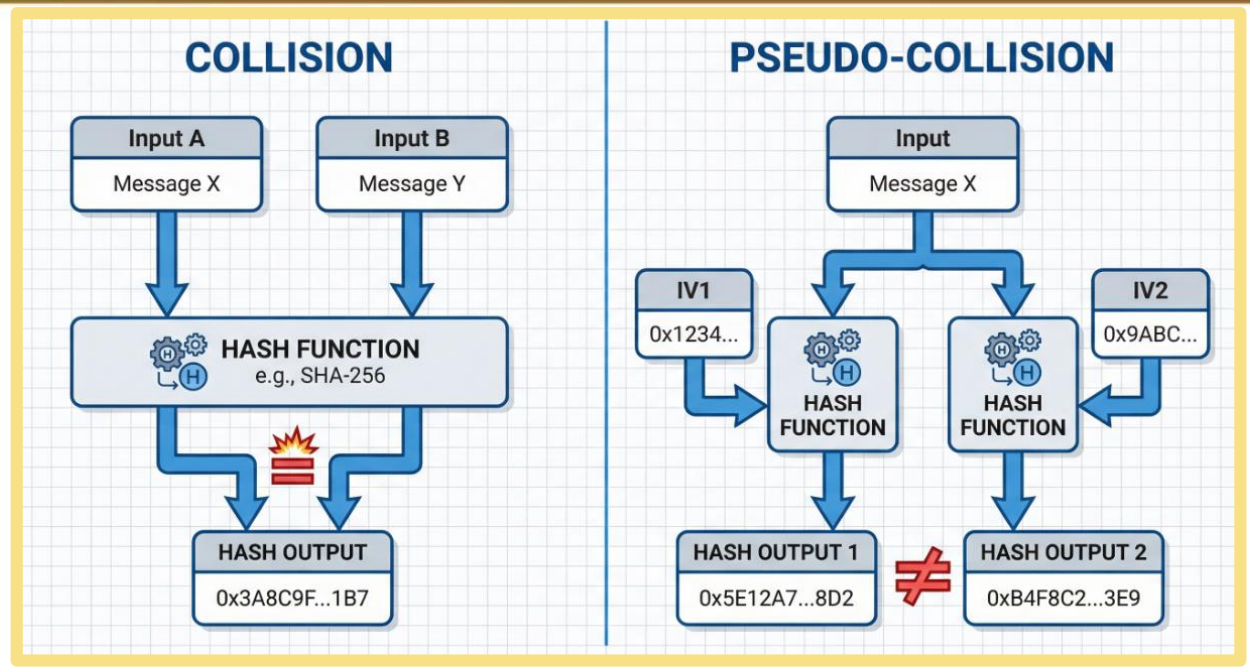
- **MD5's compression function updates a 128-bit intermediate hash by processing 512-bit message blocks through 64 steps (four rounds) that operate on four 32-bit registers (a, b, c, d) using bitwise functions, modular additions, and cyclic left rotations; the result is added to the IHV to form the next hash.**





Collisions & Pseudo-collisions





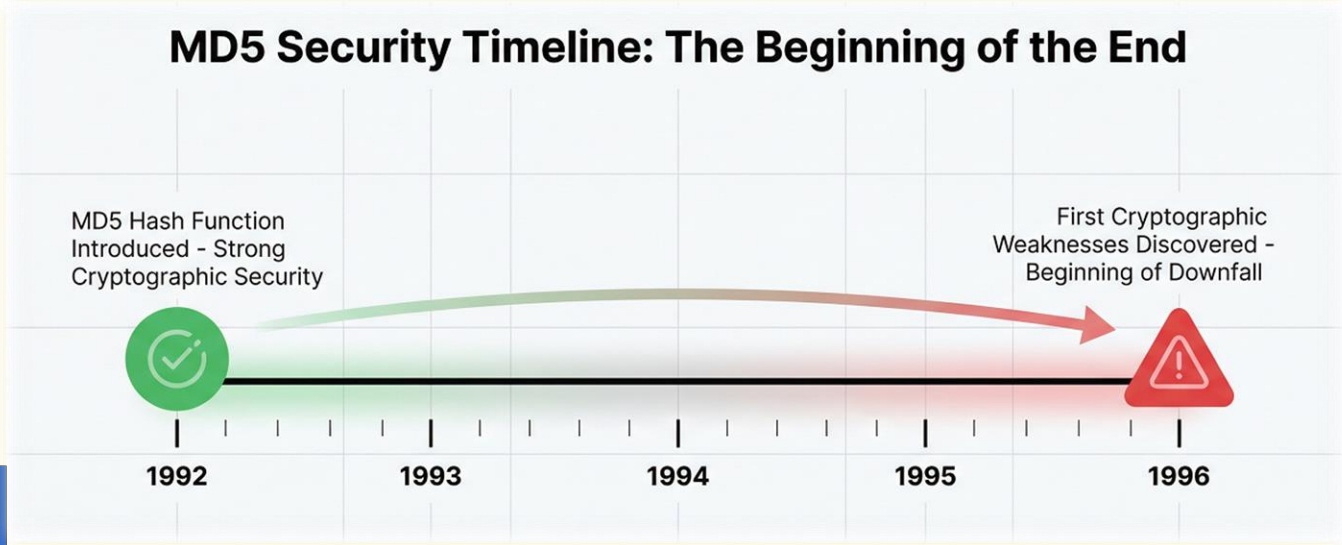
A **collision** occurs when two distinct inputs produce the same hash (h(x) = h(x')).

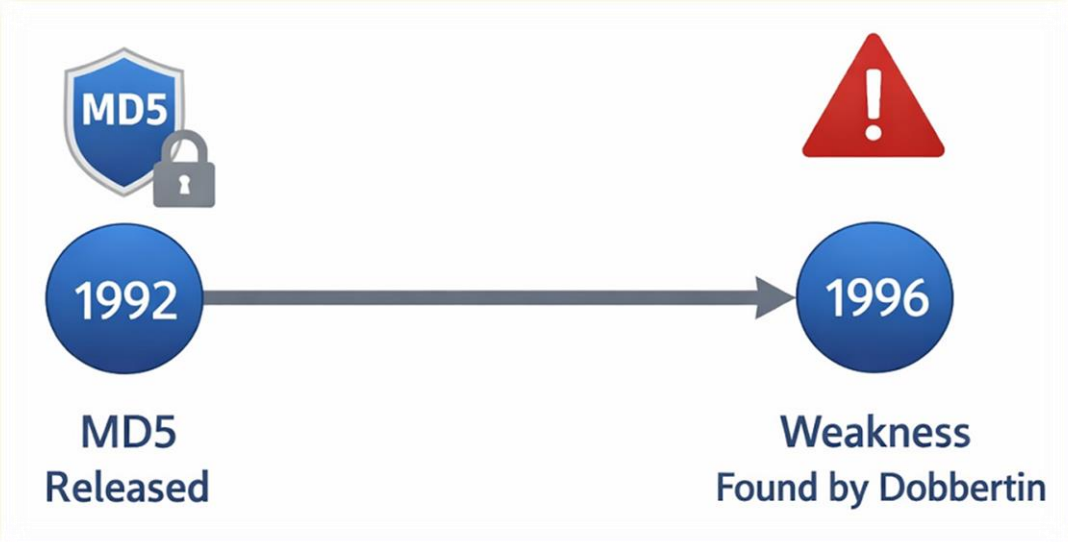
A **pseudo-collision** occurs when different initialization vectors (IV ≠ IV') yield the same output for a (possibly identical) input.

Both reveal weaknesses in the compression function.

Early Weaknesses of MD5

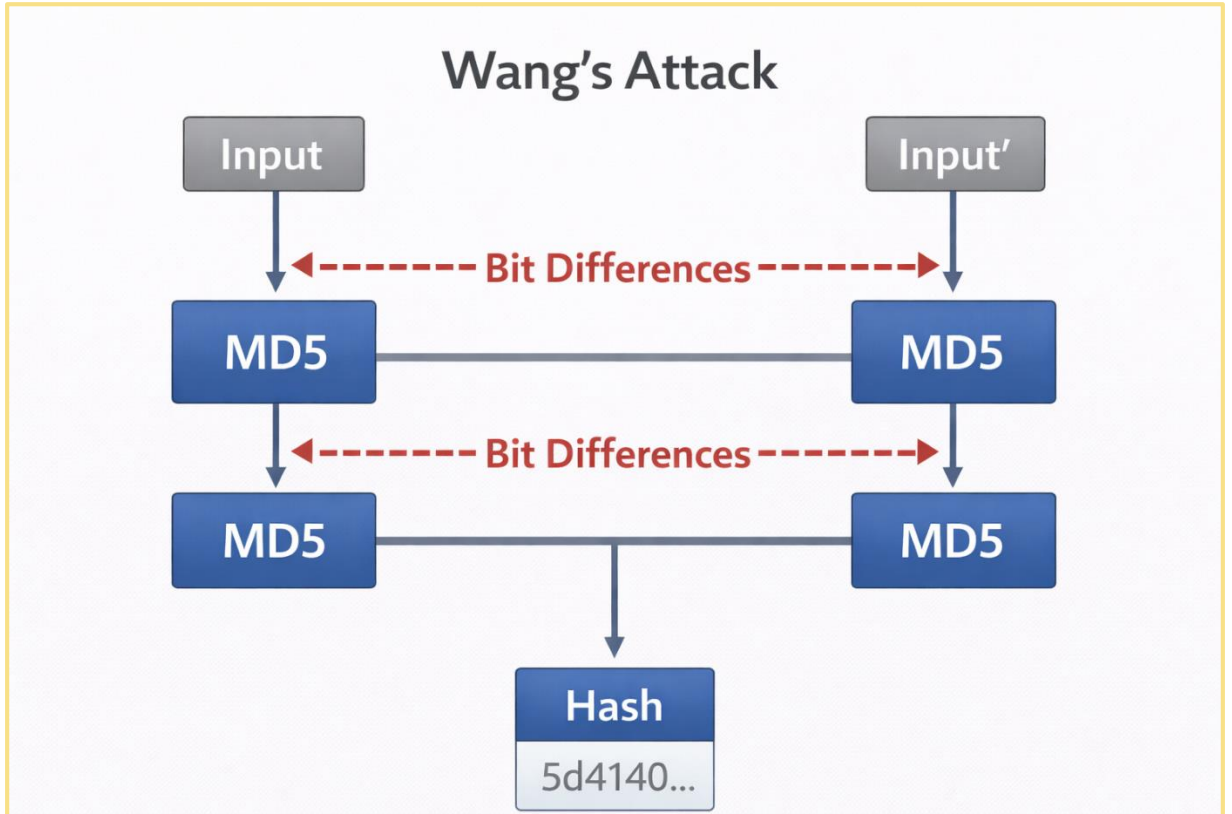
- Hans Dobbertin (1996) discovered semi-collisions in the first MD5 block, exposing weaknesses in the compression stage.
- His work led to a deeper analysis of MD5's security, inspiring more advanced attacks.





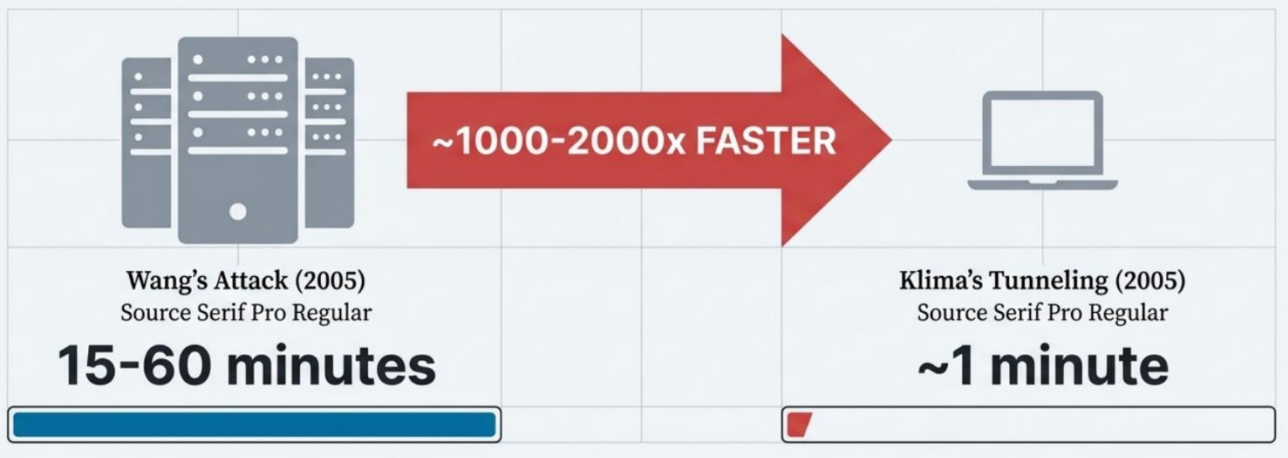
Wang’s Collision Attack

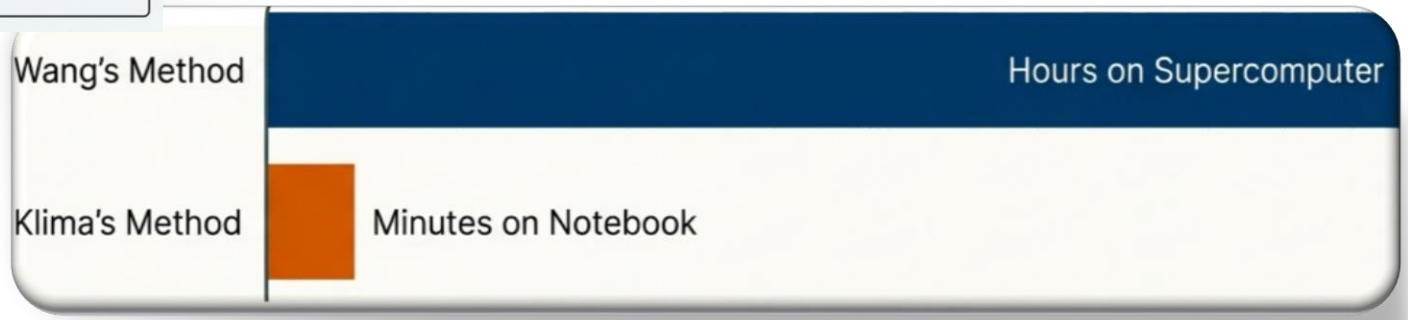
- Wang et al. (2005) performed the first full collision attack on MD5 using multi-message modification.
- Their attack combined XOR and additive differentials to generate controlled bit differences.
- Collisions were found within 15–60 minutes on IBM supercomputers.
- This breakthrough proved MD5 was no longer cryptographically safe.



Klima's Improved Method

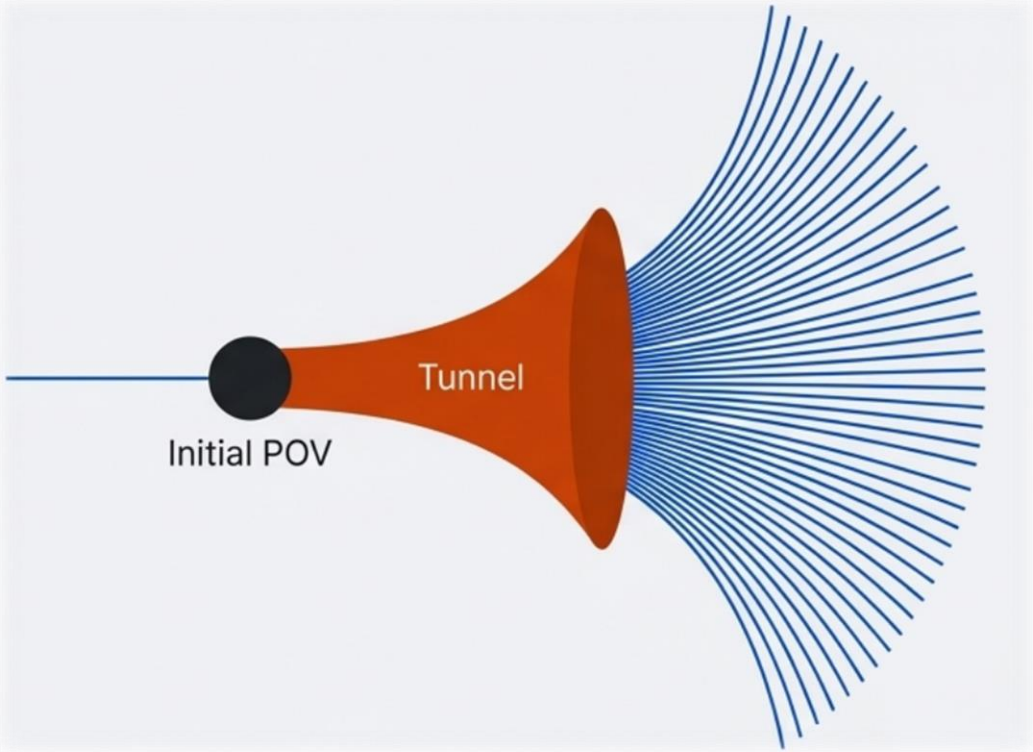
- Motivated by Wang's work, Klima developed a faster and more transparent algorithm.
- His "tunneling" approach improved collision search speed by 1000–2000X.
- Collisions could now be found on a notebook computer in about one minute.
- Unlike Wang's opaque results, Klima fully published his algorithm and source code.

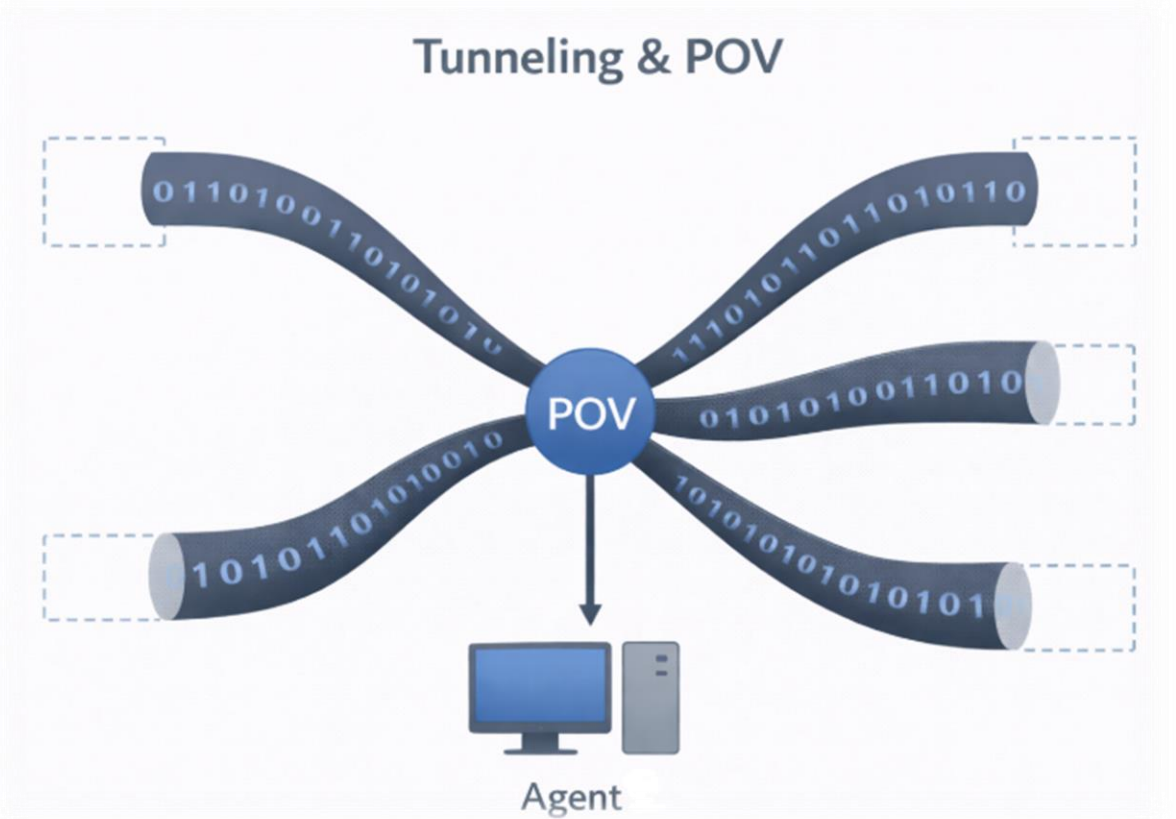




Tunneling and Points of Verification (POV)

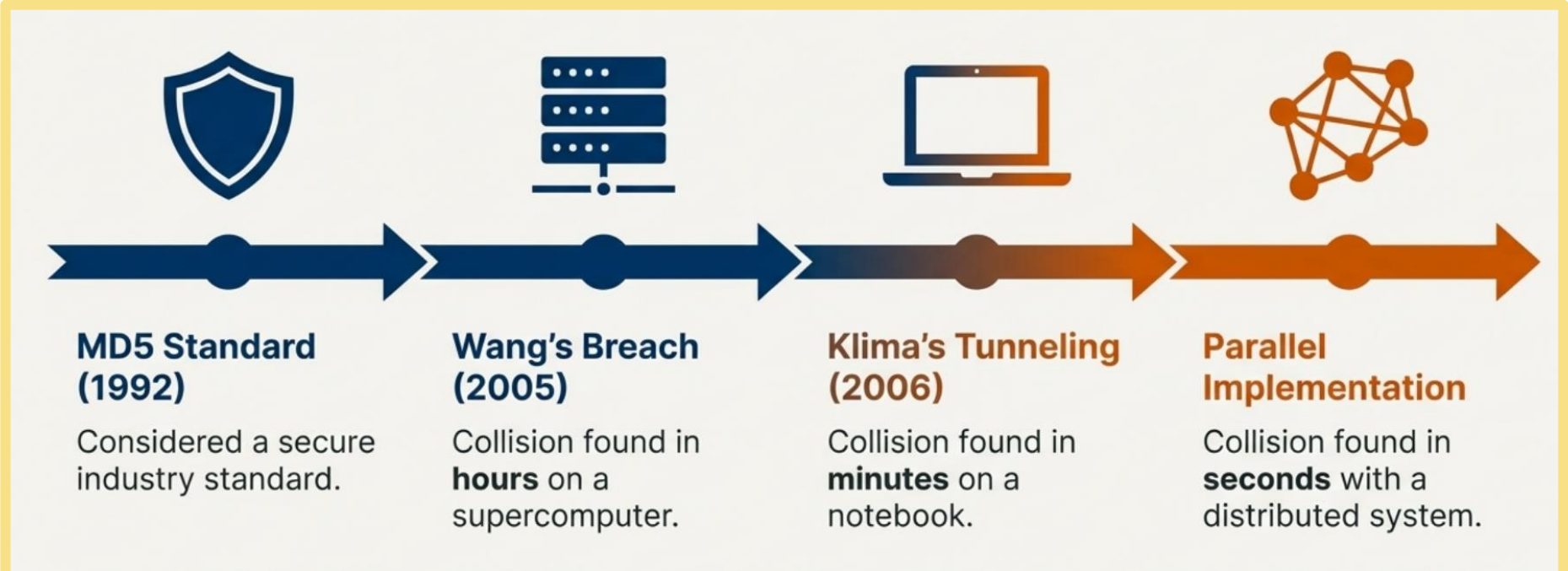
- Klima used Wang's sufficient conditions and defined 2^29 Points of Verification (POVs).
- From a valid POV, multiple tunnels are generated, each potentially leading to a collision.
- Each tunnel can spawn 2^24 new POVs, accelerating the process.
- This method works for any initialization vector (IV).





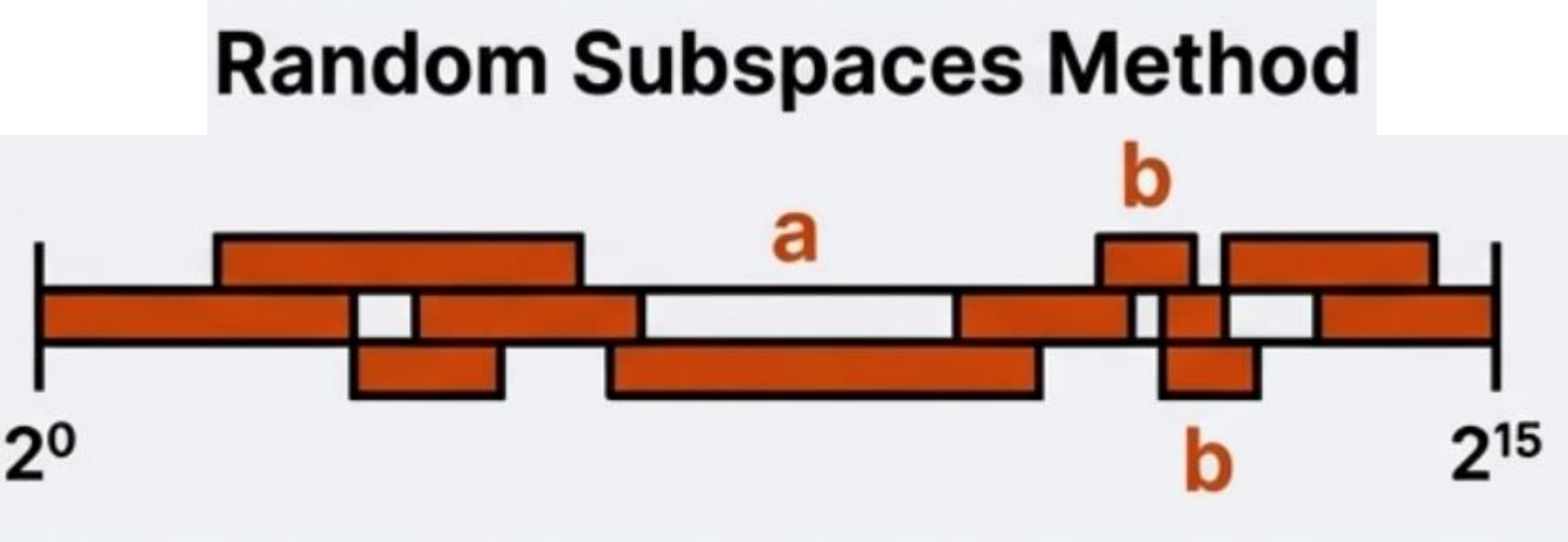
Transition to Parallel Implementation

- This study extends Klima's tunneling method to parallel and distributed environments.
- The goal: to divide the collision search space efficiently among multiple agents.
- Three methods of dividing the search space are evaluated theoretically and experimentally.



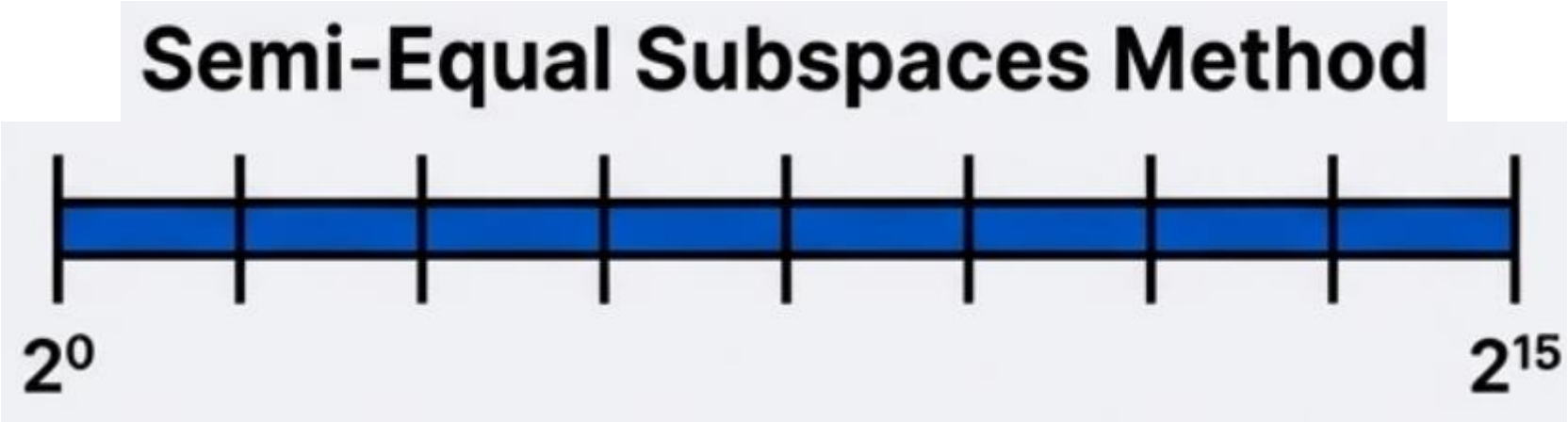
Random Subspaces Method

- Each agent randomly selects its starting point within the total search space.
- Agents independently communicate found collisions to a central server.
- The random approach causes overlaps and uneven workloads, lowering efficiency.



Semi-Equal Subspaces Method

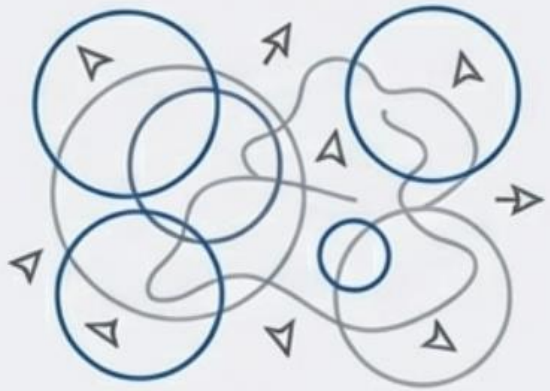
- Agents dynamically join the system and are assigned half of the largest remaining subspace
- Bidirectional communication allows the central server to redistribute tasks.
- Performance is close to optimal, with balanced workload distribution.



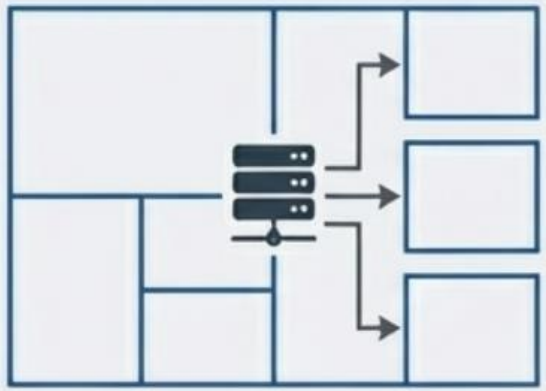
Equal Subspaces Method

- When the number of agents is known in advance, the entire space is divided equally.
- Each agent searches a fixed-size region.
- This guarantees minimal overlap and the shortest average search time.

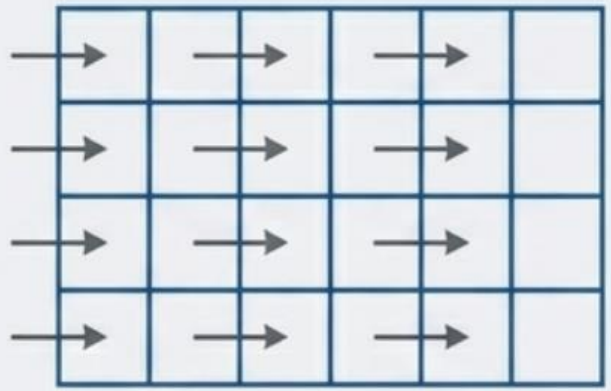
Method 1: Random Subspaces



Method 2: Semi-Equal Subspaces



Method 3: Equal Subspaces



Theoretical Comparison

- **Performance Comparison:**

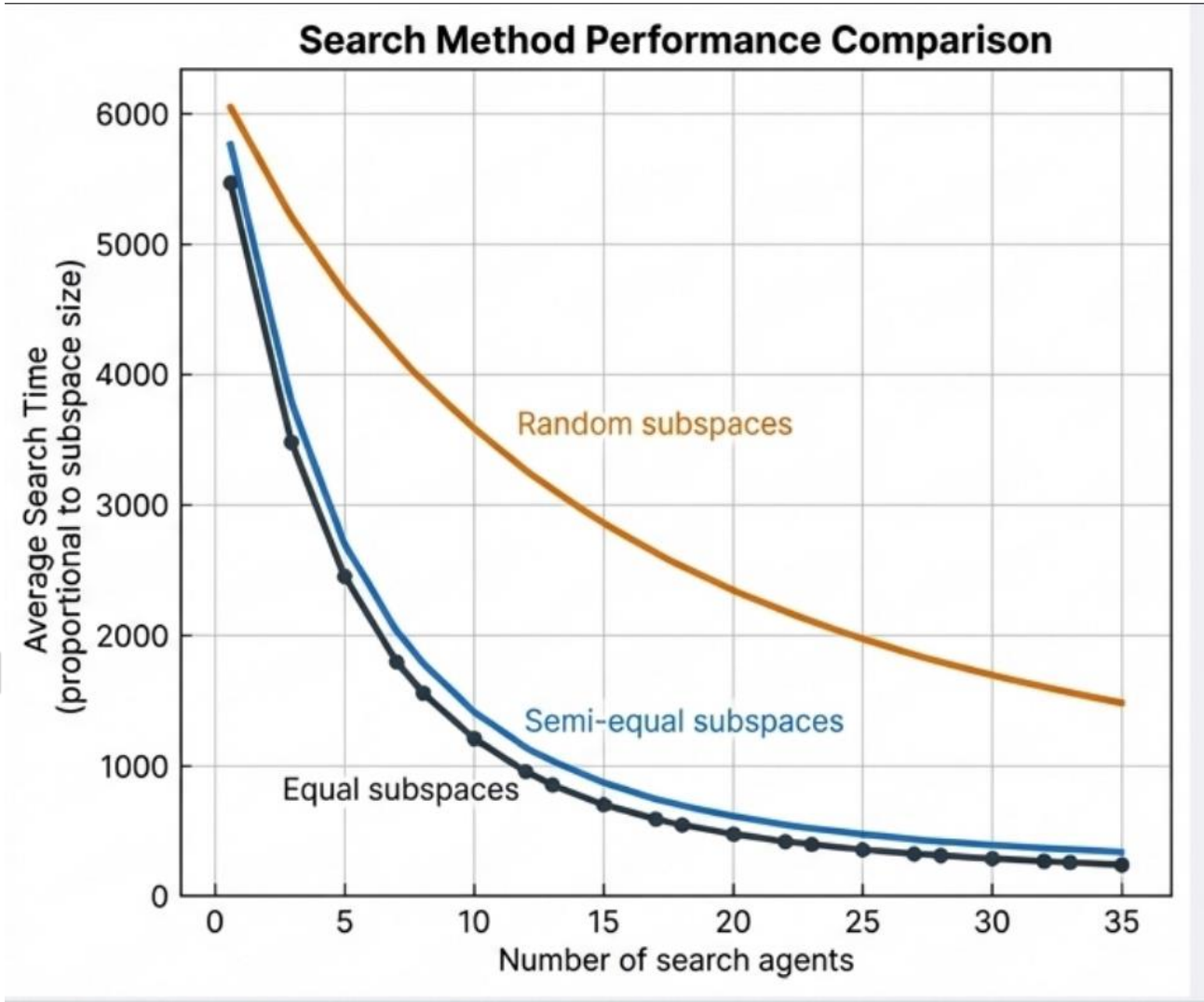
- Equal Subspaces → Best performance

- Semi-Equal Subspaces → Near-optimal

- Random Subspaces → Slowest

❖**Example: 20 agents (random) ≈ 12 agents (semi-equal) in efficiency.**

➢**Equal Subspaces consistently achieved the lowest search time.**

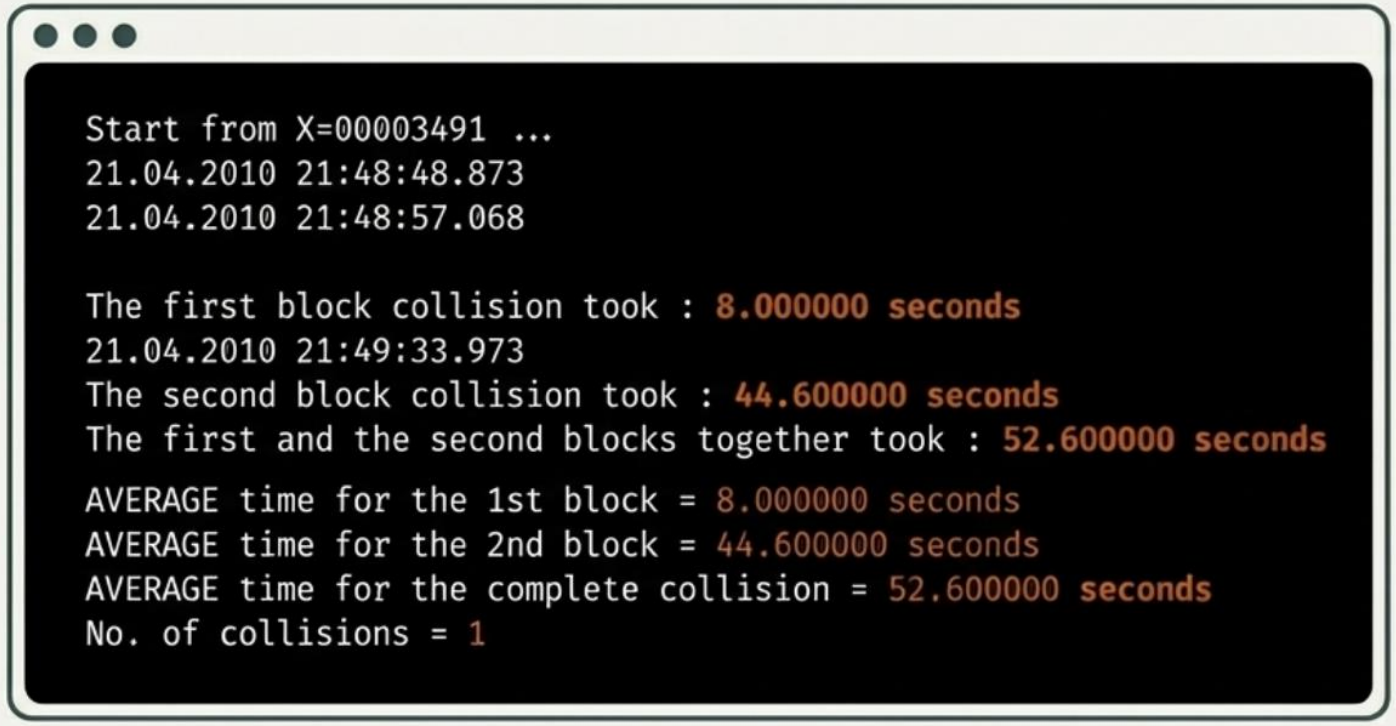


Experimental Setup

3F7A0C92...

B94E7B1D...

- ✓A prototype built in Adobe Flash CS3 implemented Klima's source code.
- ✓Random numbers ranged from 0 to 2^15.
- ✓Each new agent joining the system halves the remaining search space.
- ✓Both Semi-Equal and Random methods were tested visually.

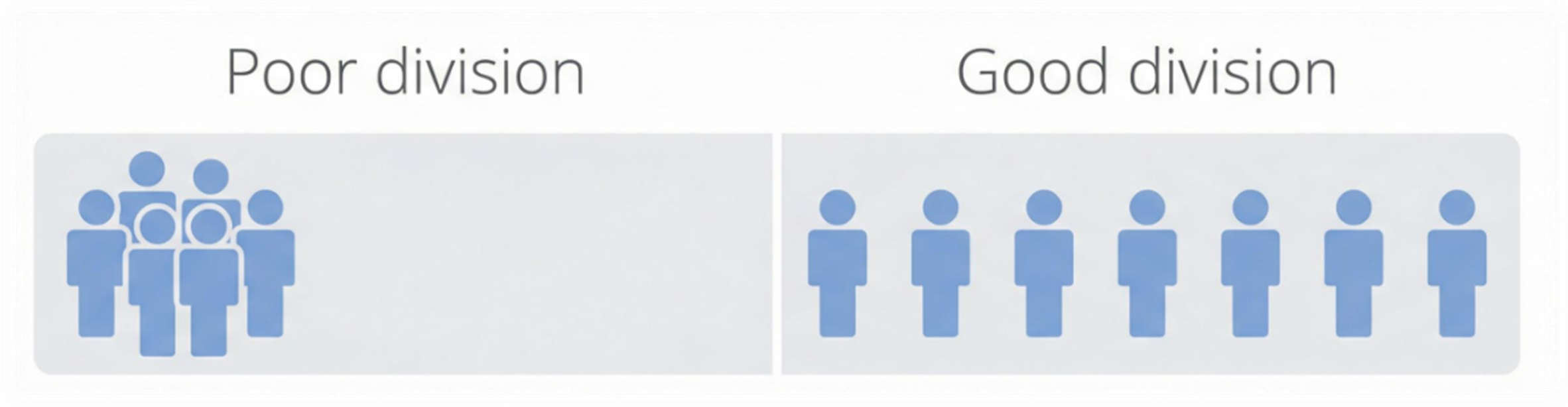


Key Results from the Test Run:

- First block collision took: **8.00 seconds**
- Second block collision took: **44.60 seconds**
- Total time for full collision: **52.60 seconds**

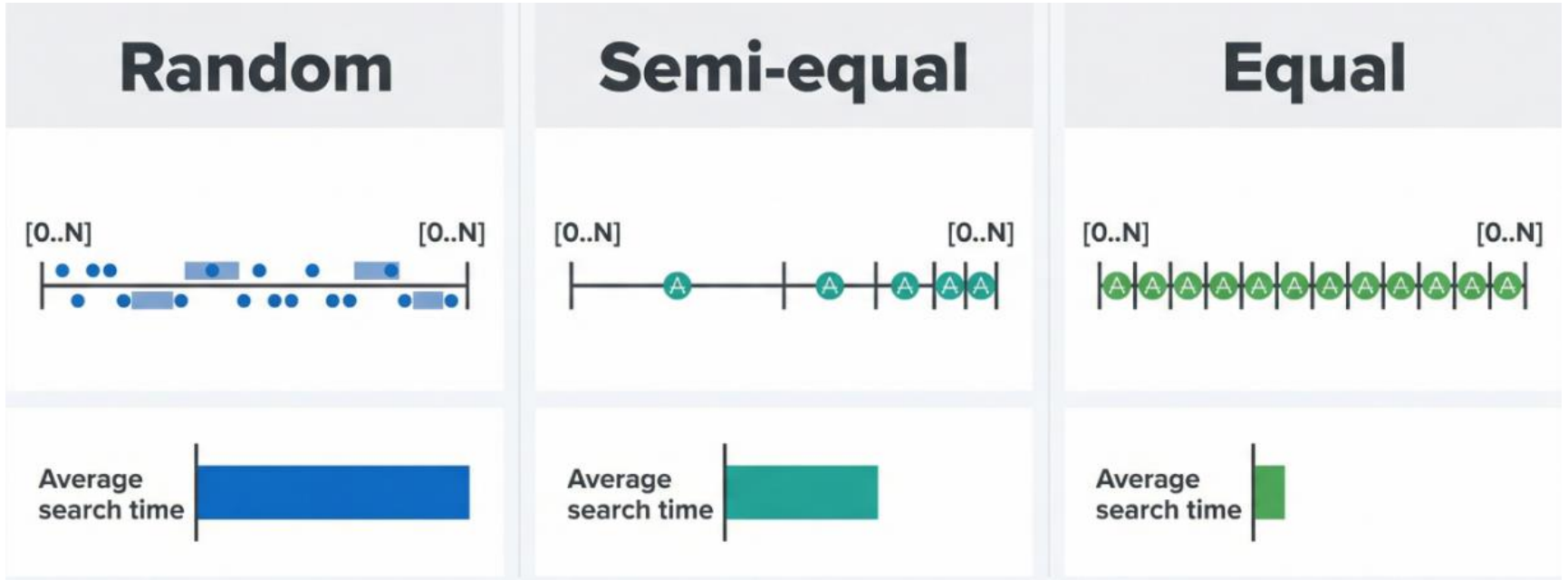
What We Learn from the Experiments

- Performance depends strongly on how the search space is divided.
- Random strategies waste computation due to overlap.
- Structured partitioning improves scalability.
- Equal and semi-equal methods clearly outperform random search.



Results and Analysis

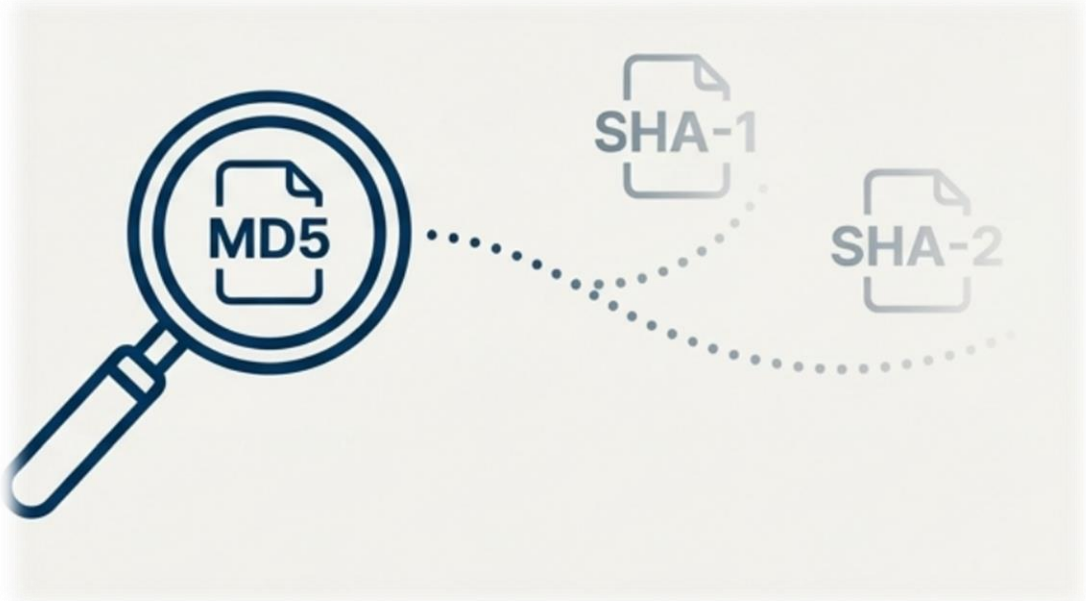
- Results prove that Equal Subspaces achieve the best performance.
- Semi-Equal performs nearly as well, ideal for systems with dynamic agent counts.
- Random Subspaces are inefficient due to overlaps and unbalanced distribution.
- Theoretical and experimental findings strongly align.



Conclusion & Future Work

The study confirms that intelligent search space division significantly improves collision search efficiency.

Future research will explore better random number generation and advanced search-space allocation techniques to apply these improvements to other hashfunctions like SHA-1 and SHA-2.



DOI:004.421:003.26

Scan to Receive the article :

JUST DO IT !

[1] H. Dobbertin, "The Status of MD5 After a Recent Attack", presented at the rump session of CryptoBytes '96, 1996.

[2] X. Wang, H. Yu, "How to Break MD5 and Other Hash Functions", EUROCRYPT 2005.

[3] V. Klima, "Finding MD5 Collisions – A Toy for a Notebook", Cryptology ePrint Archive, Report 2005/075.

[4] R. Rivest, "The MD5 Message-Digest Algorithm", RFC 1321, Internet Activities Board, 1992.

[5] J. Black, M. Cochran, T. Highland, "A Study of the MD5 Attacks: Insights and Improvements",
Fast Software Encryption – FSE, Lecture Notes in Computer Science, Vol. 4047, Springer-Verlag, 2006.

[6] M. Stevens, "Fast Collision Attack on MD5", Cryptology ePrint Archive, 2006.

[7] Rogaway, P., Shrimpton T. Cryptographic hash function basics: Definitions, implications and separations for preimage resistance, second-preimage resistance, and collision resistance. Fast Software Encryption, Lecture Notes in Computer Science, 2004.

[8] R. Rivest, "The MD4 Message-Digest Algorithm", Proceedings of CRYPTO 1990, Springer-Verlag, 1991.