Keccak

How SHA-3 came to be, and a layman's peek under its hood

Nicola Apicella

Entropia e.V.

GPN 13

Karlsruhe

Motivation

- Disclaimer: I AM NO CRYPTO-GUY!
- *It's just that sort of thing that tickles my mind…*

Nicola Apicella
nicapicella<AT>entropia<DOT>de

security considerations, proofs, attack methodologies

Agenda

- Motivation
- Hash codes
- The NIST SHA-3 competition
- Keccak
- Going further…

Nicola Apicella
nicapicella<AT>entropia<DOT>de

reactions + outlook

Hashcodes 101

- “digital fingerprint”
- easy one way, difficult the other
- (even) slight change affects result greatly
- ideally no false-positives
- signatures, authentication, hashes, fingerprinting, checksums, …

Nicola Apicella
nicapicella<AT>entropia<DOT>de

Cryptographic Hash Function is an Algorithm that, given an input, calculates a hash value.
message => digest
many uses in information security; integrity checks, password verification, data ID, pseudorandom generator

Secure Hash Algorithm

- SHA(-0)
  - flawed, withdrawn, revised
- SHA-1
  - resembles early MD5
  - weaknesses discovered
- SHA-2 (is actually 2 hash functions)
  - designed by NSA
  - no known vulnerabilities (as of yet)

Nicola Apicella
nicapicella<AT>entropia<DOT>de

Standards
MD5: “broken -> use SHA-1!” (long time ago… ;-))

Design Goals

- Performance
- Security
- Analysis
- Diversity

Nicola Apicella
nicapicella<AT>entropia<DOT>de

Analysis: We'll get to how they managed that…
Diversity: We'll get to that, too.

NIST SHA-3 Competion

- 02. Nov 2007: Announcement in FR
- 31. Oct 2008: Entry deadline (**64** entrants)
- 09. Dec 2008: **51** accepted entrants (round 1)
- Feb 2009: NIST conference
- 24. Jul 2009: **14** candidates accepted to round 2
- Aug 2010: NIST conference
- 10. Dec 2010: **5** candidates accepted as finalists
- 02. Oct 2012: **Keccak becomes SHA-3**

Nicola Apicella
nicapicella<AT>entropia<DOT>de

"NIST is initiating an effort to develop one or more additional hash algorithms through a public competition, similar to the development process for the Advanced Encryption Standard (AES)."

Rescindment Determination by the ITC

The ITC will permanently determine whether to issue after the date on which it receives notice of the institution, whether there is no reasonable likelihood that injury or threat of material injury to the domestic industry would result from the importation of the subject articles if the orders are rescinded. If the ITC so determines, it will terminate the investigation and rescind the orders; otherwise, the investigation will proceed according to statutory and regulatory time limits.

This notice is issued and published pursuant to section 751(c) of the Act.

Dated: October 29, 2010.

By order of the Commission.

Marilyn R. Abbott,

Acting Assistant Secretary for Export Administration.

[FR Doc. 2010-27814 Filed 11-4-10; 8:45 am]

BILLING CODE 3510-DS-P

DEPARTMENT OF COMMERCE

National Institute of Standards and Technology

[Docket No. 0708-XF24-00-R-01]

Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family

AGENCY: National Institute of Standards and Technology, Commerce.

ACTION: Notice and request for nominations for candidate hash algorithms.

SUMMARY: This notice solicits nominations from any interested party for candidate algorithms to be considered for SHA-3, and specifies some minimum submission criteria. It also describes the evaluation process and the minimum acceptable requirements of a "complete and proper" candidate algorithm submission. The evaluation criteria that will be used to select algorithms for SHA-3 are also described in this notice. Candidate algorithm nominations must be received by October 31, 2012. Public workshops are tentative to be held in:

DATES: Candidate algorithm nominations must be made on or before October 31, 2012. Public workshops are tentative to be held in:

ADDRESSES: Candidate algorithm nominations may be sent to: sha-3@nist.gov. Public workshops are tentative to be held in: Gaithersburg, MD, United States; and Darmstadt, Germany.

FOR FURTHER INFORMATION CONTACT: For general information about the SHA-3 family selection process, send e-mail to: sha-3@nist.gov. For questions related to a specific submission package, contact the Point of Contact listed in the submission package. For additional information regarding the Federal Information Processing Standard (FIPS) for the Secure Hash Standard, subsequent revisions of that standard, and publication of related documents, see http://csrc.nist.gov/groups/ST/hash/index.html.

SUPPLEMENTARY INFORMATION: This notice contains the following sections:

1. Background
2. Requirements for Candidate Algorithm Submissions
   - 2.1 Formal Call
   - 2.2 Deadline
   - 2.3 Algorithm Specifications and Supporting Documentation
   - 2.4 Cover Sheet
   - 2.5 Design Principles
   - 2.6 Intellectual Property Statements/Agreements/Submitter Certification
   - 2.7 Publication
   - 2.8 Consolidation
   - 2.9 Withdrawal
   - 2.10 Public Comments
   - 2.11 Additional Information
3. Minimum Acceptability Requirements
   - 3.1 Security
   - 3.2 Cost
   - 3.3 Algorithm Implementation Characteristics
4. Other Factors to Be Considered in the Selection Process
   - 4.1 Security
   - 4.2 Cost
   - 4.3 Algorithm and Implementation Characteristics
5. Publication of Selection Criteria and Timeline
6. Definitions and Abbreviations
7. Changes to the Call for Submissions
8. Publication of Submitted Algorithms
9. Withdrawal of Candidates
10. Withdrawal of the Call for Submissions

Appendix A—Publication Policy for Cryptographic Hash Algorithms

1. Background

Modern cryptographic hash functions were designed to work well, from a security standpoint, in many different applications. Hash functions are widely used and have proven to be a useful and large variable length message into a fixed length message (sometimes called a digital signature), such as DSS, or "SHA." These hash functions have also been widely used for many other purposes, including "message authentication codes" (MACs), key derivation functions, and pseudo-random functions, among other cryptographic applications.

In terms of related hash functions have been developed, such as MD4, MD5, RIPEMD, HAVAL, and SHA-1, SHA-2, SHA-3, which have also been widely used and deployed in many different security applications and protocols. However, serious attacks have been published against MD4, MD5, HAVAL, and SHA-1 (although the full impact of these attacks is not yet known), and weaknesses have been identified in some of the other algorithms.

Because of the serious attacks published against SHA-1, NIST deemed it necessary to develop a new hash algorithm in advance of the need for the SHA-2 family of hash functions to be broken. A successful, open, and public competition is a proven method for stimulating the cryptographic community to analyze a candidate algorithm's security properties thoroughly before the algorithm is widely relied upon. This competition-like process will give NIST the confidence needed that the algorithm selected will be secure and efficient when used in many practical applications. It is important that the selected algorithm be available worldwide on a royalty-free basis. Therefore, the selected algorithm must be capable of supporting existing and foreseeable security practices. Finally, the algorithm must have a good performance profile across a variety of computing platforms.

NIST announced its intent to develop a new hash algorithm in a Federal Register Notice (72 FR 62212) dated November 2, 2007. That notice also announced a workshop to be held in February 2008 to present the initial public version of the new algorithm, the cryptographic community to comment on the algorithm, and to begin the development of requirements for the new hash algorithm. This workshop was held February 25–27, 2008 at the University of California, Santa Barbara. Following the workshop, NIST published a Publication (NISTIR) 7620, "Status of the First SHA-3 Candidate Conference," which summarized the discussions at the workshop and described the first round of the public review process.

NIST announced that it is evident to replace SHA-1 with a more secure hash algorithm well in advance of the need for SHA-1 to provide adequate security for federal applications. NIST intends to hold a competition to develop a new hash algorithm, SHA-3, that will complement SHA-1 and SHA-2. It is important to note that SHA-3 will not be designed because it is broken or appears likely to be broken in the near future. Rather, NIST believes that the competition and subsequent public review process will result in a thoroughly examined and strong algorithm that will provide a high level of confidence in the security of any future applications that use it.

NIST announced that it intends to hold a public competition to develop the new algorithm in a manner similar to the development process followed for the Advanced Encryption Standard (AES). NIST worked closely with the cryptographic community during the development of the AES algorithm. The AES competition was successful because of the high level of participation from the cryptographic community. Many excellent algorithms were submitted for consideration, and the competition process was fair, open, and transparent. Similarly, NIST intends to work closely with the cryptographic community during the development of SHA-3. NIST will depend on public feedback to help determine the best candidate algorithms.

Several cryptographers have floated various ideas for what SHA-3 will look like. However, as announced at the August 2007 National Institute of Standards and Technology (NIST) Information Technology Laboratory (ITL) Invitational Workshop/Cryptographic Hash Workshop, NIST does not want to limit the competition to any particular approach. NIST is also not requiring that SHA-3 be a keyed algorithm.

NIST has decided that it is prudent to develop a new hash algorithm by an open competition, similar to the development process followed for the Advanced Encryption Standard (AES). NIST believes that this competition-like process will result in a well-vetted design that enjoys a wide degree of confidence within the cryptographic community. NIST plans to announce the winning algorithm in a publication, similar to the publication announcing the AES algorithm (Rijndael).

NIST does not intend to "certify" any algorithm submitted. However, NIST does intend to make a public declaration that the winning algorithm meets the security requirements listed in this notice. It is important to note that candidate algorithms should be judged on the basis of their security and performance characteristics, even though it is not unusual for a hash algorithm to be tweaked after a competition has ended. Therefore, the selected algorithm may be modified slightly during the standardization process to improve its security or efficiency.

NIST does not want to "stockpile" algorithms that are not developed and ready to be standardized. Therefore, the selected algorithm must be available worldwide on a royalty-free basis. Furthermore, the selected algorithm must be capable of supporting existing and foreseeable security practices. Finally, the algorithm must have a good performance profile across a variety of computing platforms.

This competition-like process will give NIST the confidence needed that the algorithm selected will be secure and efficient when used in many practical applications. It is important that SHA-3 be available worldwide on a royalty-free basis. Therefore, the selected algorithm must be capable of supporting existing and foreseeable security practices. Finally, the algorithm must have a good performance profile across a variety of computing platforms.

2. Requirements for Candidate Algorithm Submissions

2.1 Formal Call

A formal call for candidates will be announced in the Federal Register Notice announcing the availability of the final version of this publication. Candidate algorithm submissions must be received by October 31, 2008. Submissions received after this date will be returned to the sender unopened.

2.2 Deadline

Candidate algorithm submission packages must be received by October 31, 2008. Submission packages received after this date will be returned to the sender unopened.

2.3 Algorithm Specifications and Supporting Documentation

Each candidate algorithm submission package must contain a complete written specification of the algorithm, including any mathematical formulae, diagrams, and examples that will assist in understanding and evaluating the algorithm. The documentation should be sufficiently detailed to allow an independent party to completely implement the algorithm. Reference code implementations must be provided in the ANSI C programming language. Optimized code implementations may be provided in ANSI C and/or other programming languages.

2.4 Cover Sheet

A cover sheet must contain the following information:

1. Point(s) of contact (author(s), address, telephone number, fax number, and e-mail address).
2. Algorithm name.
3. Principal submitter's name.
4. Auxiliary submitters' names.
5. Statement by each author/submitter that he or she did not and will not receive financial support for his/her efforts in developing, promoting, or analyzing the algorithm from any party other than his/her employer(s) at the time the algorithm was developed.
6. Statement by each author/submitter that he or she does not and will not hold any patent claims or pending patent applications that may cover the algorithm.
7. Signature (hardcopy or electronic) of the principal submitter.

2.5 Design Principles

The submitter must explain the design principles behind the algorithm, including the following:

1. Explanation of how the algorithm's security requirements are fulfilled and why the algorithm should be expected to meet the security requirements listed in section 3.1 of this notice.
2. Explanation of how the algorithm's expected performance efficiency addresses the "NIST IR's Evaluation Criteria" (specified in section 4.3) and the anticipated application environments (see section 3.3) in which it is expected to be deployed.
3. Explanation of the advantages and limitations of the algorithm with respect to known attacks, such as length-extension and birthday attacks.
4. Specific recommendations for parameter choices (e.g., number of rounds) for the hash function, including a brief rationale for the recommendations.
5. The hash function's suitability for use in keyed modes, such as HMAC, KMAC, randomized hashing, and PRF.
6. Analysis of the algorithm's resistance against various attacks, including (but not limited to) length-extension and birthday attacks.
7. Assessment of the algorithm's susceptibility to any applicable generic attacks (see section 3.1).
8. Justification for any claimed security strength that is not at least as great as the best-supported security strength for the corresponding hash function in the SHA family (see section 3.1).
9. Discussion of the algorithm's advantages and limitations with respect to its intended applications.
10. Assessment of the algorithm's susceptibility to any applicable generic attacks (see section 3.1).
11. Justification for any claimed security strength that is not at least as great as the best-supported security strength for the corresponding hash function in the SHA family (see section 3.1).
12. Discussion of the algorithm's advantages and limitations with respect to its intended applications.

2.6 Intellectual Property Statements/Agreements/Submitter Certification

The submitted algorithm must be available worldwide on a royalty-free basis during the period of the hash function competition. It must be available under royalty-free intellectual property rights (royalty-free licenses may be obtained from the submitter) in order to allow anyone to implement the algorithm and to use it for the formal evaluation process. Therefore, the submitter must explicitly agree to make available royalty-free licenses in accordance with subparagraph (1) of section 2 of the Publication Policy for Cryptographic Hash Algorithms (see Appendix A of this notice) for the algorithm, its implementations, or the use of the algorithm or any of its implementations.

Each submitter must provide a statement that he/she is not aware of any patent claims or pending patent applications that may cover the algorithm, and that he/she will not seek patent claims on the algorithm in the future. If the submitter is aware of any patent claims or pending patent applications that may cover the algorithm, the submitter must provide a statement that he/she will make available royalty-free licenses in accordance with subparagraph (1) of section 2 of the Publication Policy for Cryptographic Hash Algorithms (see Appendix A of this notice) for the algorithm, its implementations, or the use of the algorithm or any of its implementations.

2.7 Publication

Each submitted algorithm must be described in sufficient detail to allow qualified cryptographers to duplicate the results. Source code must be provided.

2.8 Consolidation

The consolidation of similar algorithms submitted by different groups is encouraged. Consolidation may occur before the start of the first round, between rounds, or during the third round. Consolidation may be proposed by the submitter(s) of the algorithm(s) to be consolidated, or by NIST. Consolidation of algorithms will only occur if the submitters of the algorithms to be consolidated agree to the consolidation. Consolidation is not required in order for a submitted algorithm to be considered complete and proper.

2.9 Withdrawal

Withdrawal of algorithms is allowed at any time during the competition. However, withdrawal of algorithms is discouraged. Withdrawal of algorithms after the start of the third round may result in a reduction in the number of algorithms considered during the third round.

2.10 Public Comments

Public comments on the submitted algorithms will be accepted during the comment periods announced for each round of the competition. Comments should be sent to sha-3@nist.gov. Comments will be posted on the NIST website as soon as possible after the comment period begins.

2.11 Additional Information

Additional information regarding the competition will be posted on the competition website as it becomes available. This information may include:

- Publication of the accepted submissions.
- Presentations given at any workshops.
- Comments received.
- Reports on the evaluation of the candidate algorithms.
- Any revisions that NIST decides to make to its announcements or requirements.
- Other helpful information.

3. Minimum Acceptability Requirements

In order to provide a basis for the comparison and evaluation of candidate algorithms submitted to be considered for SHA-3, candidate algorithms will be required to meet certain minimum acceptability requirements. Algorithms that do not meet these requirements will not be considered complete and proper, and will not be evaluated. Complete submissions will be announced by NIST shortly after the receipt deadline.

3.1 Security

The security provided by an algorithm is the most important factor in the evaluation. Algorithms will be judged on the following factors:

1. Application-Independent Security

Algorithms having the same level (length in bits) will be considered to be of equal security strength with respect to finding a preimage, finding a second preimage, and finding a collision. Although it is true that the SHA-2 family of hash functions is believed to have a considerably higher security strength than the SHA-1 family, which both algorithms offer the same security strength.

NIST has decided that it is prudent to develop a new hash algorithm by an open competition, similar to the development process followed for the Advanced Encryption Standard (AES). NIST believes that this competition-like process will result in a well-vetted design that enjoys a wide degree of confidence within the cryptographic community. NIST plans to announce the winning algorithm in a publication, similar to the publication announcing the AES algorithm (Rijndael).

NIST does not intend to "certify" any algorithm submitted. However, NIST does intend to make a public declaration that the winning algorithm meets the security requirements listed in this notice. It is important to note that candidate algorithms should be judged on the basis of their security and performance characteristics, even though it is not unusual for a hash algorithm to be tweaked after a competition has ended. Therefore, the selected algorithm may be modified slightly during the standardization process to improve its security or efficiency.

NIST does not want to "stockpile" algorithms that are not developed and ready to be standardized. Therefore, the selected algorithm must be available worldwide on a royalty-free basis. Furthermore, the selected algorithm must be capable of supporting existing and foreseeable security practices. Finally, the algorithm must have a good performance profile across a variety of computing platforms.

This competition-like process will give NIST the confidence needed that the algorithm selected will be secure and efficient when used in many practical applications. It is important that SHA-3 be available worldwide on a royalty-free basis. Therefore, the selected algorithm must be capable of supporting existing and foreseeable security practices. Finally, the algorithm must have a good performance profile across a variety of computing platforms.

factors (scored in the order of relative importance):

4.1 Security

The security provided by an algorithm is the most important factor in the evaluation. Algorithms will be judged on the following factors:

1. Application-Independent Security

Algorithms having the same level (length in bits) will be considered to be of equal security strength with respect to finding a preimage, finding a second preimage, finding a collision, and finding a pseudo-random output (PRO) (see 3.1, 2nd bullet). NIST views Publication 800-107, Recommendation for Applications Using Approved Hash Algorithms, as the definitive source for guidance on the strength of each approved hash algorithm.

- Collision resistance of approximately n/2 bits.
- Preimage resistance of approximately n bits.
- Second preimage resistance of approximately n bits for any message shorter than n bits.
- Resistance to length extension attacks, and
- Any other application-independent security weakness that would call into question the suitability of the algorithm for any of the cryptographic uses listed in section 3.3 (Minimum Acceptability Requirements).

NIST has decided that it is prudent to develop a new hash algorithm by an open competition, similar to the development process followed for the Advanced Encryption Standard (AES). NIST believes that this competition-like process will result in a well-vetted design that enjoys a wide degree of confidence within the cryptographic community. NIST plans to announce the winning algorithm in a publication, similar to the publication announcing the AES algorithm (Rijndael).

NIST does not intend to "certify" any algorithm submitted. However, NIST does intend to make a public declaration that the winning algorithm meets the security requirements listed in this notice. It is important to note that candidate algorithms should be judged on the basis of their security and performance characteristics, even though it is not unusual for a hash algorithm to be tweaked after a competition has ended. Therefore, the selected algorithm may be modified slightly during the standardization process to improve its security or efficiency.

NIST does not want to "stockpile" algorithms that are not developed and ready to be standardized. Therefore, the selected algorithm must be available worldwide on a royalty-free basis. Furthermore, the selected algorithm must be capable of supporting existing and foreseeable security practices. Finally, the algorithm must have a good performance profile across a variety of computing platforms.

This competition-like process will give NIST the confidence needed that the algorithm selected will be secure and efficient when used in many practical applications. It is important that SHA-3 be available worldwide on a royalty-free basis. Therefore, the selected algorithm must be capable of supporting existing and foreseeable security practices. Finally, the algorithm must have a good performance profile across a variety of computing platforms.

2. Additional Security Requirements of the Hash Functions

In addition to the specific requirements mentioned above, NIST expects the SHA-3 algorithm of message digest size n to meet the following security requirements, or a reasonable subset. These requirements are believed to be satisfied by the ideal hash function of message digest size n (see 3.1, 1st bullet). Any algorithm submitted that does not meet these requirements will be considered to be a weakness.

- Collision resistance of approximately n/2 bits.
- Preimage resistance of approximately n bits.
- Second preimage resistance of approximately n bits for any message shorter than n bits.
- Resistance to length extension attacks, and
- Any other application-independent security weakness that would call into question the suitability of the algorithm for any of the cryptographic uses listed in section 3.3 (Minimum Acceptability Requirements).

NIST has decided that it is prudent to develop a new hash algorithm by an open competition, similar to the development process followed for the Advanced Encryption Standard (AES). NIST believes that this competition-like process will result in a well-vetted design that enjoys a wide degree of confidence within the cryptographic community. NIST plans to announce the winning algorithm in a publication, similar to the publication announcing the AES algorithm (Rijndael).

NIST does not intend to "certify" any algorithm submitted. However, NIST does intend to make a public declaration that the winning algorithm meets the security requirements listed in this notice. It is important to note that candidate algorithms should be judged on the basis of their security and performance characteristics, even though it is not unusual for a hash algorithm to be tweaked after a competition has ended. Therefore, the selected algorithm may be modified slightly during the standardization process to improve its security or efficiency.

NIST does not want to "stockpile" algorithms that are not developed and ready to be standardized. Therefore, the selected algorithm must be available worldwide on a royalty-free basis. Furthermore, the selected algorithm must be capable of supporting existing and foreseeable security practices. Finally, the algorithm must have a good performance profile across a variety of computing platforms.

This competition-like process will give NIST the confidence needed that the algorithm selected will be secure and efficient when used in many practical applications. It is important that SHA-3 be available worldwide on a royalty-free basis. Therefore, the selected algorithm must be capable of supporting existing and foreseeable security practices. Finally, the algorithm must have a good performance profile across a variety of computing platforms.

4.2 Cost

As described in section 4.1.1, existence of hash algorithms may reduce overall system cost requirements for use in the formal 1 evaluation. In the following, the term "algorithm" is used to mean the entire, including all of its variants (e.g., SHA-2 includes digest sizes of 224, 256, 384, and 512 bits).

Some examples of "flexibility" may include (but are not limited to) the following:

1. The algorithm has a tweakable parameter which allows the selection of a range of security and performance properties.
2. The algorithm can be implemented efficiently in hardware, software, or both.
3. Computational efficiency. The computational efficiency of the candidate algorithms will be an important factor in the evaluation. Both software and hardware implementations will be considered. Evidence that a candidate algorithm is inefficient relative to the other candidates will be a significant factor in the decision to eliminate it from consideration. Elegant and innovative designs are encouraged.
4. Memory requirements. The memory required to implement a candidate algorithm, for both software and hardware implementations of the algorithm, will be considered during the evaluation process. The relative importance of memory requirements will be assessed during the evaluation. The memory requirements will be primarily an issue for constrained environments, but efficient memory usage is desirable in all implementations.
5. Flexibility. Candidate algorithms will be judged according to its relative flexibility.

4.3 Algorithm and Implementation Characteristics

1. Flexibility. Candidate algorithms with greater flexibility will meet the needs of more users in more environments, and therefore, are preferable. However, some extensions of functionality may affect the security provided by the algorithm, and vice versa. In the following, the security provided by an algorithm is assumed to be sufficient (at least 112 bits of security).

Some examples of "flexibility" may include (but are not limited to) the following:

1. The algorithm has a tweakable parameter which allows the selection of a range of security and performance properties.
2. The algorithm can be implemented efficiently in hardware, software, or both.
3. Computational efficiency. The computational efficiency of the candidate algorithms will be an important factor in the evaluation process (e.g., speed). Both software and hardware implementations will be considered. Evidence that a candidate algorithm is inefficient relative to the other candidates will be a significant factor in the decision to eliminate it from consideration. Elegant and innovative designs are encouraged.
4. Memory requirements. The memory required to implement a candidate algorithm, for both software and hardware implementations of the algorithm, will be considered during the evaluation process. The relative importance of memory requirements will be assessed during the evaluation. The memory requirements will be primarily an issue for constrained environments, but efficient memory usage is desirable in all implementations.
5. Flexibility. Candidate algorithms will be judged according to its relative flexibility.

4.4 Other

As described in section 4.2.1, existence of hash algorithms may reduce overall system cost requirements for use in the formal 1 evaluation. In the following, the term "algorithm" is used to mean the entire, including all of its variants (e.g., SHA-2 includes digest sizes of 224, 256, 384, and 512 bits).

Some examples of "flexibility" may include (but are not limited to) the following:

1. The algorithm has a tweakable parameter which allows the selection of a range of security and performance properties.
2. The algorithm can be implemented efficiently in hardware, software, or both.
3. Computational efficiency. The computational efficiency of the candidate algorithms will be an important factor in the evaluation process (e.g., speed). Both software and hardware implementations will be considered. Evidence that a candidate algorithm is inefficient relative to the other candidates will be a significant factor in the decision to eliminate it from consideration. Elegant and innovative designs are encouraged.
4. Memory requirements. The memory required to implement a candidate algorithm, for both software and hardware implementations of the algorithm, will be considered during the evaluation process. The relative importance of memory requirements will be assessed during the evaluation. The memory requirements will be primarily an issue for constrained environments, but efficient memory usage is desirable in all implementations.
5. Flexibility. Candidate algorithms will be judged according to its relative flexibility.

4. Other Factors to Be Considered in the Selection Process

4.1 Security

The security provided by an algorithm is the most important factor in the evaluation. Algorithms will be judged on the following factors:

1. Application-Independent Security

Algorithms having the same level (length in bits) will be considered to be of equal security strength with respect to finding a preimage, finding a second preimage, finding a collision, and finding a pseudo-random output (PRO) (see 3.1, 1st bullet). NIST views Publication 800-107, Recommendation for Applications Using Approved Hash Algorithms, as the definitive source for guidance on the strength of each approved hash algorithm.

- Collision resistance of approximately n/2 bits.
- Preimage resistance of approximately n bits.
- Second preimage resistance of approximately n bits for any message shorter than n bits.
- Resistance to length extension attacks, and
- Any other application-independent security weakness that would call into question the suitability of the algorithm for any of the cryptographic uses listed in section 3.3 (Minimum Acceptability Requirements).

2. Additional Security Requirements of the Hash Functions

In addition to the specific requirements mentioned above, NIST expects the SHA-3 algorithm of message digest size n to meet the following security requirements, or a reasonable subset. These requirements are believed to be satisfied by the ideal hash function of message digest size n (see 3.1, 1st bullet). Any algorithm submitted that does not meet these requirements will be considered to be a weakness.

- Collision resistance of approximately n/2 bits.
- Preimage resistance of approximately n bits.
- Second preimage resistance of approximately n bits for any message shorter than n bits.
- Resistance to length extension attacks, and
- Any other application-independent security weakness that would call into question the suitability of the algorithm for any of the cryptographic uses listed in section 3.3 (Minimum Acceptability Requirements).

4.2 Cost

As described in section 4.1.1, existence of hash algorithms may reduce overall system cost requirements for use in the formal 1 evaluation. In the following, the term "algorithm" is used to mean the entire, including all of its variants (e.g., SHA-2 includes digest sizes of 224, 256, 384, and 512 bits).

Some examples of "flexibility" may include (but are not limited to) the following:

1. The algorithm has a tweakable parameter which allows the selection of a range of security and performance properties.
2. The algorithm can be implemented efficiently in hardware, software, or both.
3. Computational efficiency. The computational efficiency of the candidate algorithms will be an important factor in the evaluation process (e.g., speed). Both software and hardware implementations will be considered. Evidence that a candidate algorithm is inefficient relative to the other candidates will be a significant factor in the decision to eliminate it from consideration. Elegant and innovative designs are encouraged.
4. Memory requirements. The memory required to implement a candidate algorithm, for both software and hardware implementations of the algorithm, will be considered during the evaluation process. The relative importance of memory requirements will be assessed during the evaluation. The memory requirements will be primarily an issue for constrained environments, but efficient memory usage is desirable in all implementations.
5. Flexibility. Candidate algorithms will be judged according to its relative flexibility.

4.3 Algorithm and Implementation Characteristics

1. Flexibility. Candidate algorithms with greater flexibility will meet the needs of more users in more environments, and therefore, are preferable. However, some extensions of functionality may affect the security provided by the algorithm, and vice versa. In the following, the security provided by an algorithm is assumed to be sufficient (at least 112 bits of security).

Some examples of "flexibility" may include (but are not limited to) the following:

1. The algorithm has a tweakable parameter which allows the selection of a range of security and performance properties.
2. The algorithm can be implemented efficiently in hardware, software, or both.
3. Computational efficiency. The computational efficiency of the candidate algorithms will be an important factor in the evaluation process (e.g., speed). Both software and hardware implementations will be considered. Evidence that a candidate algorithm is inefficient relative to the other candidates will be a significant factor in the decision to eliminate it from consideration. Elegant and innovative designs are encouraged.
4. Memory requirements. The memory required to implement a candidate algorithm, for both software and hardware implementations of the algorithm, will be considered during the evaluation process. The relative importance of memory requirements will be assessed during the evaluation. The memory requirements will be primarily an issue for constrained environments, but efficient memory usage is desirable in all implementations.
5. Flexibility. Candidate algorithms will be judged according to its relative flexibility.

4.4 Other

As described in section 4.2.1, existence of hash algorithms may reduce overall system cost requirements for use in the formal 1 evaluation. In the following, the term "algorithm" is used to mean the entire, including all of its variants (e.g., SHA-2 includes digest sizes of 224, 256, 384, and 512 bits).

Some examples of "flexibility" may include (but are not limited to) the following:

1. The algorithm has a tweakable parameter which allows the selection of a range of security and performance properties.
2. The algorithm can be implemented efficiently in hardware, software, or both.
3. Computational efficiency. The computational efficiency of the candidate algorithms will be an important factor in the evaluation process (e.g., speed). Both software and hardware implementations will be considered. Evidence that a candidate algorithm is inefficient relative to the other candidates will be a significant factor in the decision to eliminate it from consideration. Elegant and innovative designs are encouraged.
4. Memory requirements. The memory required to implement a candidate algorithm, for both software and hardware implementations of the algorithm, will be considered during the evaluation process. The relative importance of memory requirements will be assessed during the evaluation. The memory requirements will be primarily an issue for constrained environments, but efficient memory usage is desirable in all implementations.
5. Flexibility. Candidate algorithms will be judged according to its relative flexibility.

Testing will be performed by NIST using the submitted reference implementations provided by the submitters. Optimized implementations will be considered for the evaluation process, but will not be used in the selection decision. Submitters are encouraged to provide implementations that are optimized for various platforms and environments that are likely to be important in practice. Implementations that are received prior to the beginning of Round 1 will be posted on the SHA-3 website at http://csrc.nist.gov/groups/ST/hash/sha-3/index.html, along with the other submitted algorithms and implementations, as soon as possible after the submission deadline. Requests for the withdrawal of submitted algorithms will only be honored until the submission deadline. After the submission deadline, withdrawal of algorithms will only occur in exceptional circumstances, such as the discovery of a fatal flaw.

4.5 Out

As described in section 4.2.1, existence of hash algorithms may reduce overall system cost requirements for use in the formal 1 evaluation. In the following, the term "algorithm" is used to mean the entire, including all of its variants (e.g., SHA-2 includes digest sizes of 224, 256, 384, and 512 bits).

Some examples of "flexibility" may include (but are not limited to) the following:

1. The algorithm has a tweakable parameter which allows the selection of a range of security and performance properties.
2. The algorithm can be implemented efficiently in hardware, software, or both.
3. Computational efficiency. The computational efficiency of the candidate algorithms will be an important factor in the evaluation process (e.g., speed). Both software and hardware implementations will be considered. Evidence that a candidate algorithm is inefficient relative to the other candidates will be a significant factor in the decision to eliminate it from consideration. Elegant and innovative designs are encouraged.
4. Memory requirements. The memory required to implement a candidate algorithm, for both software and hardware implementations of the algorithm, will be considered during the evaluation process. The relative importance of memory requirements will be assessed during the evaluation. The memory requirements will be primarily an issue for constrained environments, but efficient memory usage is desirable in all implementations.
5. Flexibility. Candidate algorithms will be judged according to its relative flexibility.

5. Publication of Selection Criteria and Timeline

5.1 Cover Sheet

A cover sheet must contain the following information:

1. Point(s) of contact (author(s), address, telephone number, fax number, and e-mail address).
2. Algorithm name.
3. Principal submitter's name.
4. Auxiliary submitters' names.
5. Statement by each author/submitter that he or she did not and will not receive financial support for his/her efforts in developing, promoting, or analyzing the algorithm from any party other than his/her employer(s) at the time the algorithm was developed.
6. Statement by each author/submitter that he or she does not and will not hold any patent claims or pending patent applications that may cover the algorithm.
7. Signature (hardcopy or electronic) of the principal submitter.

5.2 Design Principles

The submitter must explain the design principles behind the algorithm, including the following:

1. Explanation of how the algorithm's security requirements are fulfilled and why the algorithm should be expected to meet the security requirements listed in section 3.1 of this notice.
2. Explanation of how the algorithm's expected performance efficiency addresses the "NIST IR's Evaluation Criteria" (specified in section 4.3) and the anticipated application environments (see section 3.3) in which it is expected to be deployed.
3. Explanation of the advantages and limitations of the algorithm with respect to known attacks, such as length-extension and birthday attacks.
4. Specific recommendations for parameter choices (e.g., number of rounds) for the hash function, including a brief rationale for the recommendations.
5. The hash function's suitability for use in keyed modes, such as HMAC, KMAC, randomized hashing, and PRF.
6. Analysis of the algorithm's resistance against various attacks, including (but not limited to) length-extension and birthday attacks.
7. Assessment of the algorithm's susceptibility to any applicable generic attacks (see section 3.1).
8. Justification for any claimed security strength that is not at least as great as the best-supported security strength for the corresponding hash function in the SHA family (see section 3.1).
9. Discussion of the algorithm's advantages and limitations with respect to its intended applications.

5.3 Algorithm Specifications and Supporting Documentation

Each candidate algorithm submission package must contain a complete written specification of the algorithm, including any mathematical formulae, diagrams, and examples that will assist in understanding and evaluating the algorithm. The documentation should be sufficiently detailed to allow an independent party to completely implement the algorithm. Reference code implementations must be provided in the ANSI C programming language. Optimized code implementations may be provided in ANSI C and/or other programming languages.

5.4 Cover Sheet

A cover sheet must contain the following information:

1. Point(s) of contact (author(s), address, telephone number, fax number, and e-mail address).
2. Algorithm name.
3. Principal submitter's name.
4. Auxiliary submitters' names.
5. Statement by each author/submitter that he or she did not and will not receive financial support for his/her efforts in developing, promoting, or analyzing the algorithm from any party other than his/her employer(s) at the time the algorithm was developed.
6. Statement by each author/submitter that he or she does not and will not hold any patent claims or pending patent applications that may cover the algorithm.
7. Signature (hardcopy or electronic) of the principal submitter.

5.5 Design Principles

The submitter must explain the design principles behind the algorithm, including the following:

1. Explanation of how the algorithm's security requirements are fulfilled and why the algorithm should be expected to meet the security requirements listed in section 3.1 of this notice.
2. Explanation of how the algorithm's expected performance efficiency addresses the "NIST IR's Evaluation Criteria" (specified in section 4.3) and the anticipated application environments (see section 3.3) in which it is expected to be deployed.
3. Explanation of the advantages and limitations of the algorithm with respect to known attacks, such as length-extension and birthday attacks.
4. Specific recommendations for parameter choices (e.g., number of rounds) for the hash function, including a brief rationale for the recommendations.
5. The hash function's suitability for use in keyed modes, such as HMAC, KMAC, randomized hashing, and PRF.
6. Analysis of the algorithm's resistance against various attacks, including (but not limited to) length-extension and birthday attacks.
7. Assessment of the algorithm's susceptibility to any applicable generic attacks (see section 3.1).
8. Justification for any claimed security strength that is not at least as great as the best-supported security strength for the corresponding hash function in the SHA family (see section 3.1).
9. Discussion of the algorithm's advantages and limitations with respect to its intended applications.

5.6 Intellectual Property Statements/Agreements/Submitter Certification

The submitted algorithm must be available worldwide on a royalty-free basis during the period of the hash function competition. It must be available under royalty-free intellectual property rights (royalty-free licenses may be obtained from the submitter) in order to allow anyone to implement the algorithm and to use it for the formal evaluation process. Therefore, the submitter must explicitly agree to make available royalty-free licenses in accordance with subparagraph (1) of section 2 of the Publication Policy for Cryptographic Hash Algorithms (see Appendix A of this notice) for the algorithm, its implementations, or the use of the algorithm or any of its implementations.

Each submitter must provide a statement that he/she is not aware of any patent claims or pending patent applications that may cover the algorithm, and that he/she will not seek patent claims on the algorithm in the future. If the submitter is aware of any patent claims or pending patent applications that may cover the algorithm, the submitter must provide a statement that he/she will make available royalty-free licenses in accordance with subparagraph (1) of section 2 of the Publication Policy for Cryptographic Hash Algorithms (see Appendix A of this notice) for the algorithm, its implementations, or the use of the algorithm or any of its implementations.

5.7 Publication

Each submitted algorithm must be described in sufficient detail to allow qualified cryptographers to duplicate the results. Source code must be provided.

5.8 Consolidation

The consolidation of similar algorithms submitted by different groups is encouraged. Consolidation may occur before the start of the first round, between rounds, or during the third round. Consolidation may be proposed by the submitter(s) of the algorithm(s) to be consolidated, or by NIST. Consolidation of algorithms will only occur if the submitters of the algorithms to be consolidated agree to the consolidation. Consolidation is not required in order for a submitted algorithm to be considered complete and proper.

5.9 Withdrawal

Withdrawal of algorithms is allowed at any time during the competition. However, withdrawal of algorithms is discouraged. Withdrawal of algorithms after the start of the third round may result in a reduction in the number of algorithms considered during the third round.

5.10 Public Comments

Public comments on the submitted algorithms will be accepted during the comment periods announced for each round of the competition. Comments should be sent to sha-3@nist.gov. Comments will be posted on the NIST website as soon as possible after the comment period begins.

5.11 Additional Information

Additional information regarding the competition will be posted on the competition website as it becomes available. This information may include:

- Publication of the accepted submissions.
- Presentations given at any workshops.
- Comments received.
- Reports on the evaluation of the candidate algorithms.
- Any revisions that NIST decides to make to its announcements or requirements.
- Other helpful information.

6. Definitions and Abbreviations

Algorithm: A computational procedure that is well-defined enough that it can be expressed as a computer program.

Birthday attack: A generic attack that finds collisions in a hash function with a workload of approximately $2^{n/2}$ hash function computations, where n is the number of bits in the hash function's output.

Collision: A pair of distinct inputs that hash to the same output.

Collision resistance: A property of a hash function for which it is computationally infeasible to find a collision.

Cryptographic hash function: A function that maps a bit string of arbitrary length to a fixed-length bit string, and satisfies the following three properties: preimage resistance, second preimage resistance, and collision resistance.

Digital signature: A value computed with a cryptographic algorithm and appended to a data object in order to provide origin authentication, data integrity, and signatory non-repudiation.

Hash function: A function that maps a bit string of arbitrary length to a fixed-length bit string, and satisfies the following two properties: preimage resistance and second preimage resistance.

Hash value: The output of a hash function.

Keyed hash function: A hash function that accepts two inputs: a key and a message.

Length-extension attack: Given the hash value $h(x)$ of some unknown input x, the ability to find $h(x \| y)$ for some y.

Message authentication code (MAC): A keyed cryptographic function that is used to provide assurance of the source and integrity of data.

Preimage: Any input that hashes to a given output.

Preimage resistance: A property of a hash function for which it is computationally infeasible to find a preimage.

Pseudo-random function (PRF): A keyed cryptographic function that is used to provide a pseudo-random output.

Pseudo-random output (PRO): An output that is computationally indistinguishable from a truly random output.

Second preimage: Given an input, an input that hashes to the same output.

Second preimage resistance: A property of a hash function for which it is computationally infeasible to find a second preimage.

7. Changes to the Call for Submissions

NIST may find it necessary to announce additional requirements or clarifications after the publication of the final call for submissions. Any such announcements will be posted on the SHA-3 website at http://csrc.nist.gov/groups/ST/hash/s

64 entrants, 51 accepted (13 not; 5 known)
10 Conceded – 16 Substantial Weaknesses – 11 Round 2 – 9 Final Round – 5 Finalists
AURORA (Sony); MD6 (Rivest), SANDstorm (Sandia National Laboratories);
Fugue (IBM), ECHO (France Telecom), CubeHash (Bernstein, qmail Salsa20);
Skein (Schneier), Grøstl (Knudsen)

Design Goals

- Performance
- Security
- Analysis
- Diversity

Nicola Apicella
nicapicella<AT>entropia<DOT>de

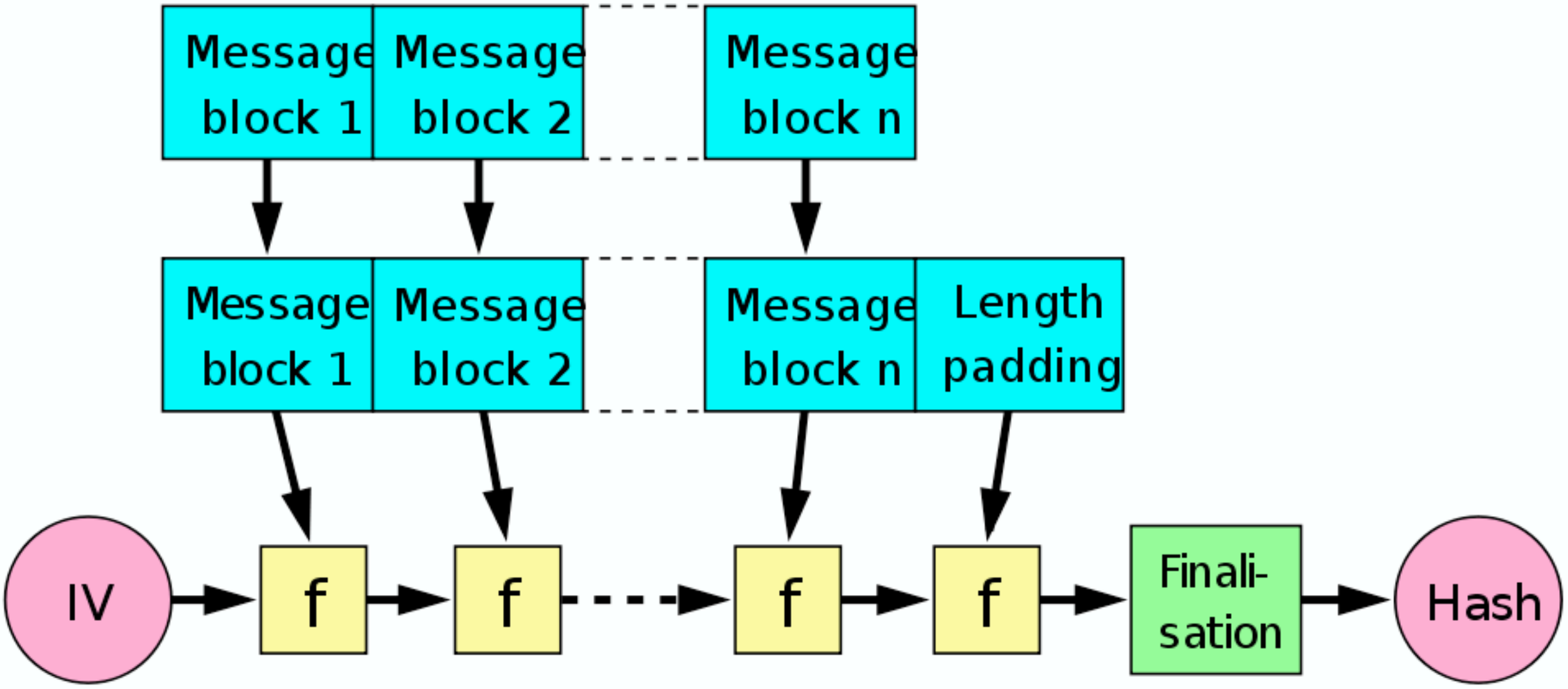
1. Hardware focus 2. ‘nervous’; too fast
3. tweaks or lack of cryptanalysis; tested & mature
4. with regards to each other and, in particular, to SHA-<3 (SPONGE vs. MD)

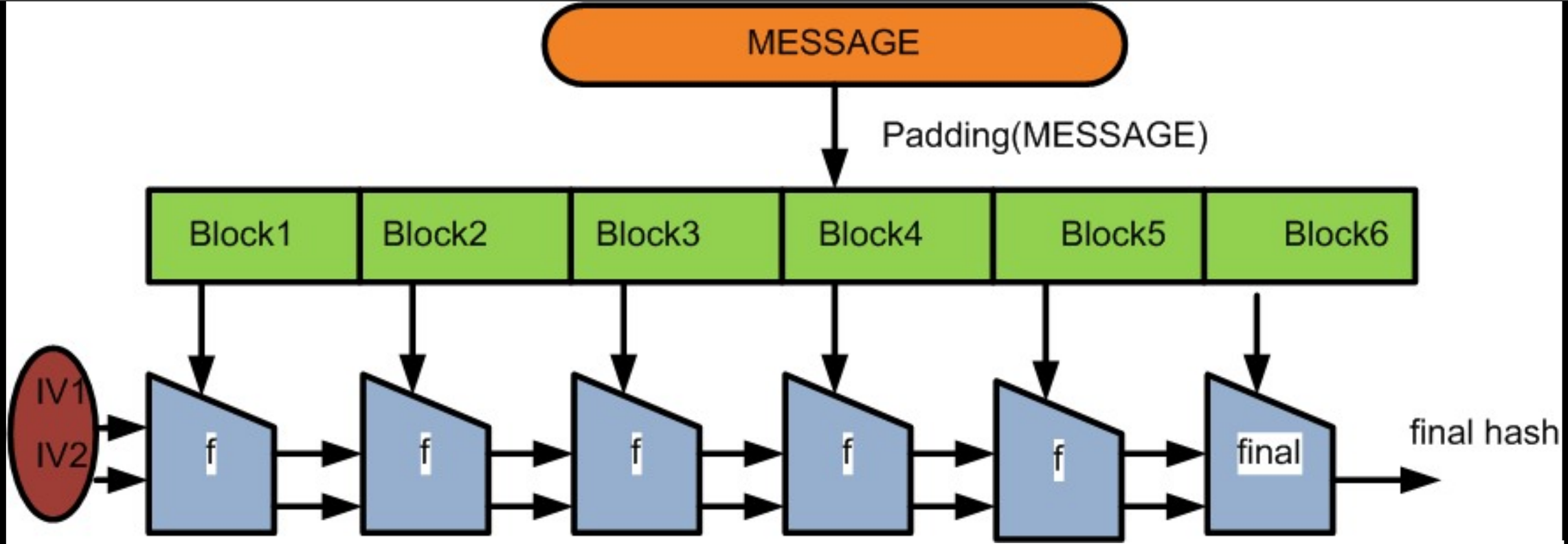
Sponge Function?

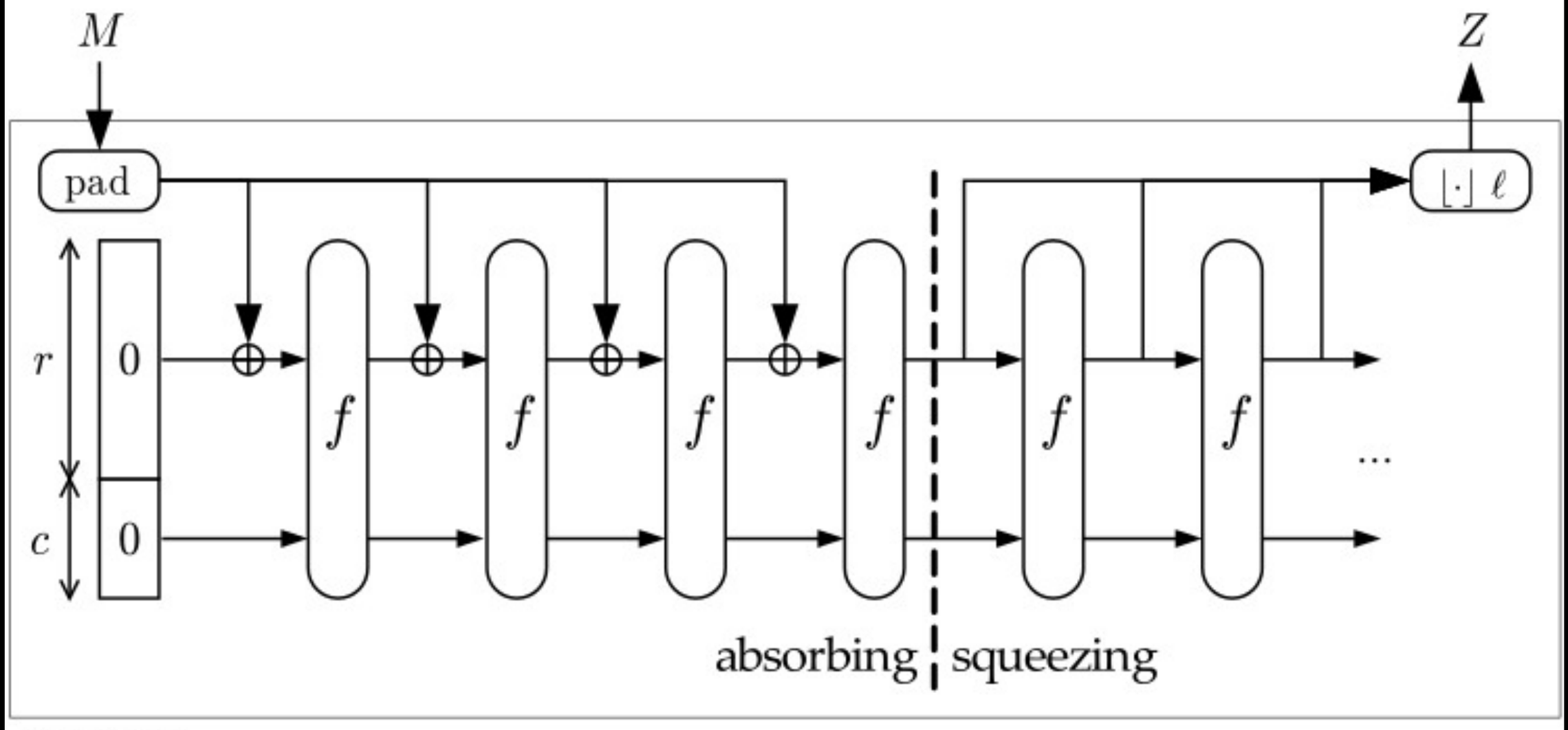
- Merkle-Damgård construction
- Sponge function
  - special case of wide-pipe Merkle-Damgård

Nicola Apicella
nicapicella<AT>entropia<DOT>de

mention padding? 10*1
MD5, all SHAs use MD
“collisions can be cascaded”


Message block 1
Message block 2
Message block n
Message block 1
Message block 2
Message block n
Length padding
IV
f
f
f
f
Finali-sation
Hash


MESSAGE
Padding(MESSAGE)
Block1
Block2
Block3
Block4
Block5
Block6
IV1
IV2
f
final
final hash


M
Z
pad
⌊·⌋ ℓ
r
c
0
0
f
...
absorbing
squeezing

sponge

Keccak's History

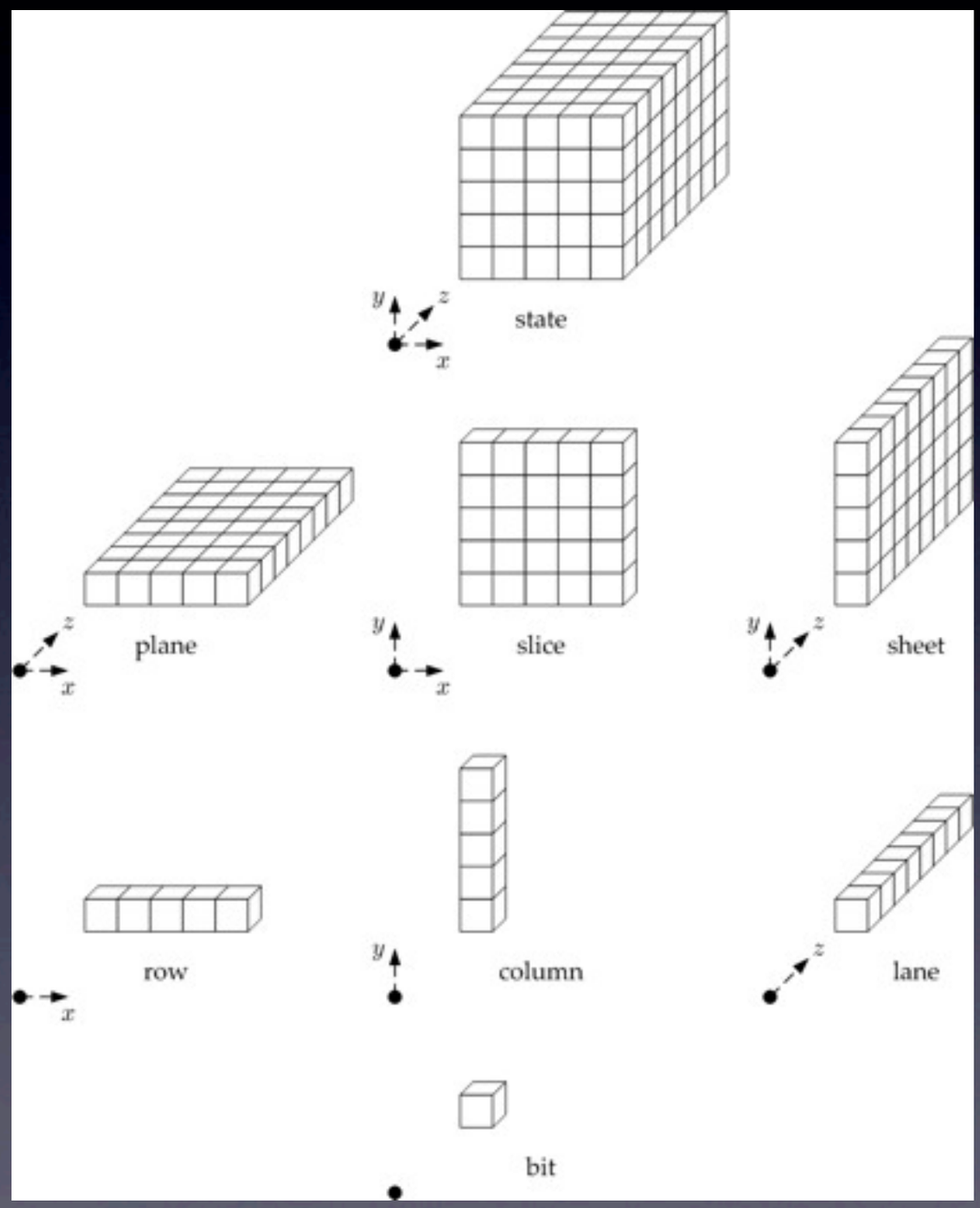
- Bertoni, **Daemen**, Peeters, Van Assche
- 2002: Panama (broken)
- 2006: RadioGatún
- Keccak becomes SHA-3

Nicola Apicella
nicapicella<AT>entropia<DOT>de

Who is this winner…
they work at chip labs
Daemen: part of Rijnadel, which became AES (2001); similar procedure, similar requirements
Panama: Stream cipher & hash function
"considerable revisions" => Keccak

Keccak: Basics

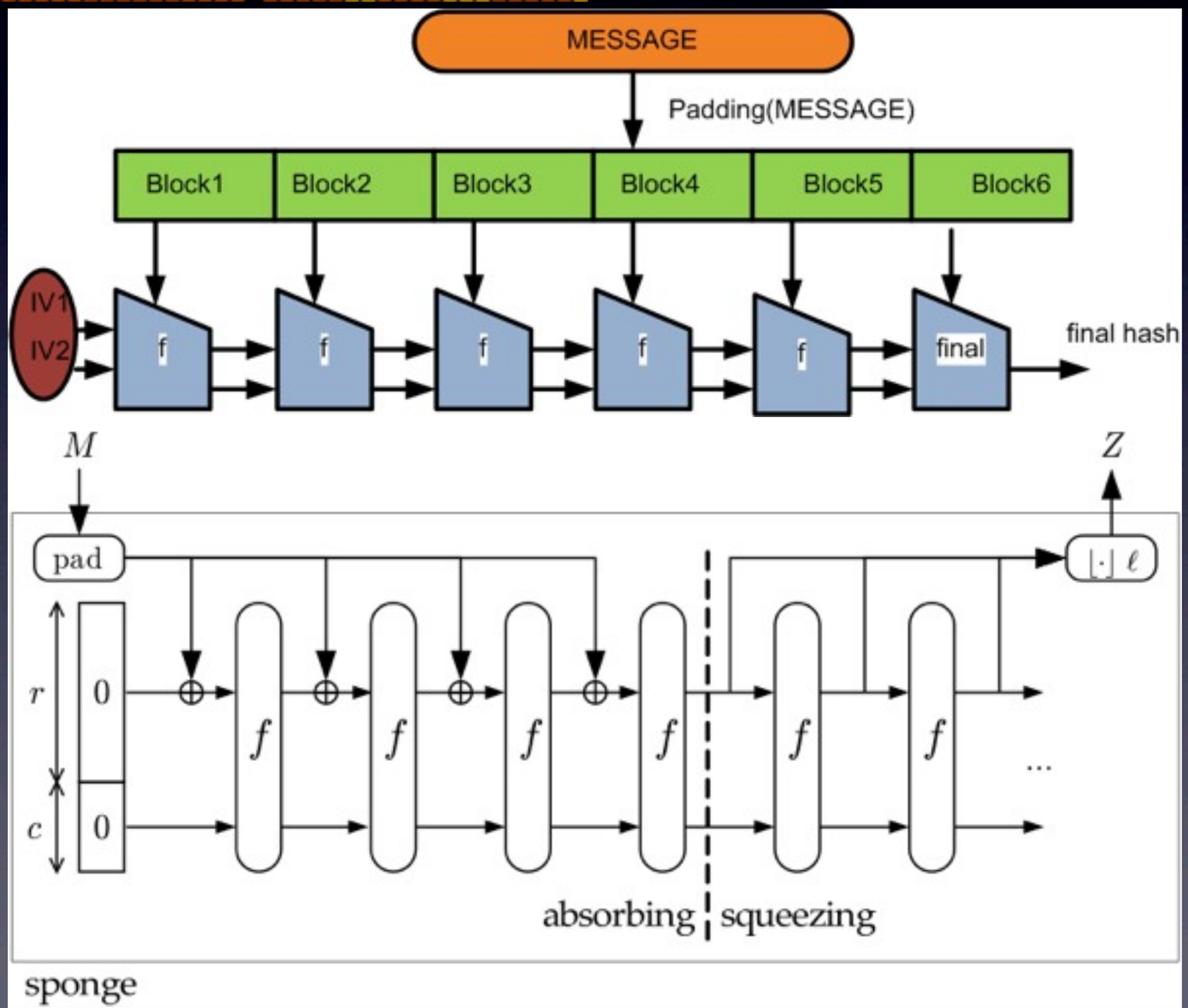
- initial empty state, 5 x 5 x w
  - w = 2^l
  - typically: l = 6; w = 64
- **absorb message block, scramble, repeat**
- squeeze (as needed)



Nicola Apicella
nicapicella<AT>entropia<DOT>de

super simple: ~300 lines of Python! (documented, unoptimized code)
(calculations mod5)
l=<6; >6 makes no sense (CPU-word 128 bits)
“Matryoshka principle”!
Keacck-f[1600]: 5*5*64

Absorb



Nicola Apicella
nicapicella<AT>entropia<DOT>de

CODE: padding: 10*1 (harder than it seems…)
CODE: initialize empty array (easy peasy…)
CODE: XOR first message block into current state (then scramble before continuing)

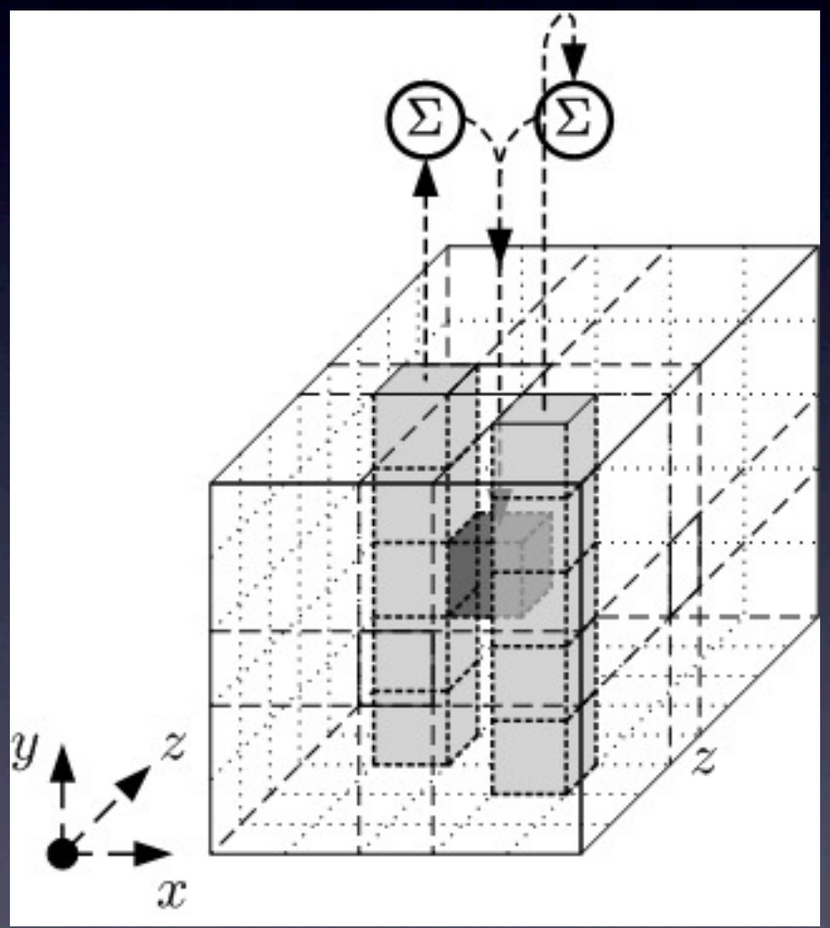
“Scramble”

- block permutation
- 12+2*l (=24) iterations of 5 sub-rounds:
  - θ (theta)
  - χ (chi)
  - π (pi)
  - ρ (rho)
  - ι (iota)

Nicola Apicella
nicapicella<AT>entropia<DOT>de

really, really simple sub-rounds
Theta is first; others arbitrary according to Keccak Reference
“NIST reference code”?
CODE: for loop…

Keccak-f: Theta



Nicola Apicella
nicapicella<AT>entropia<DOT>de

always first operation
CODE: Python
“high average diffusion and low gate count”

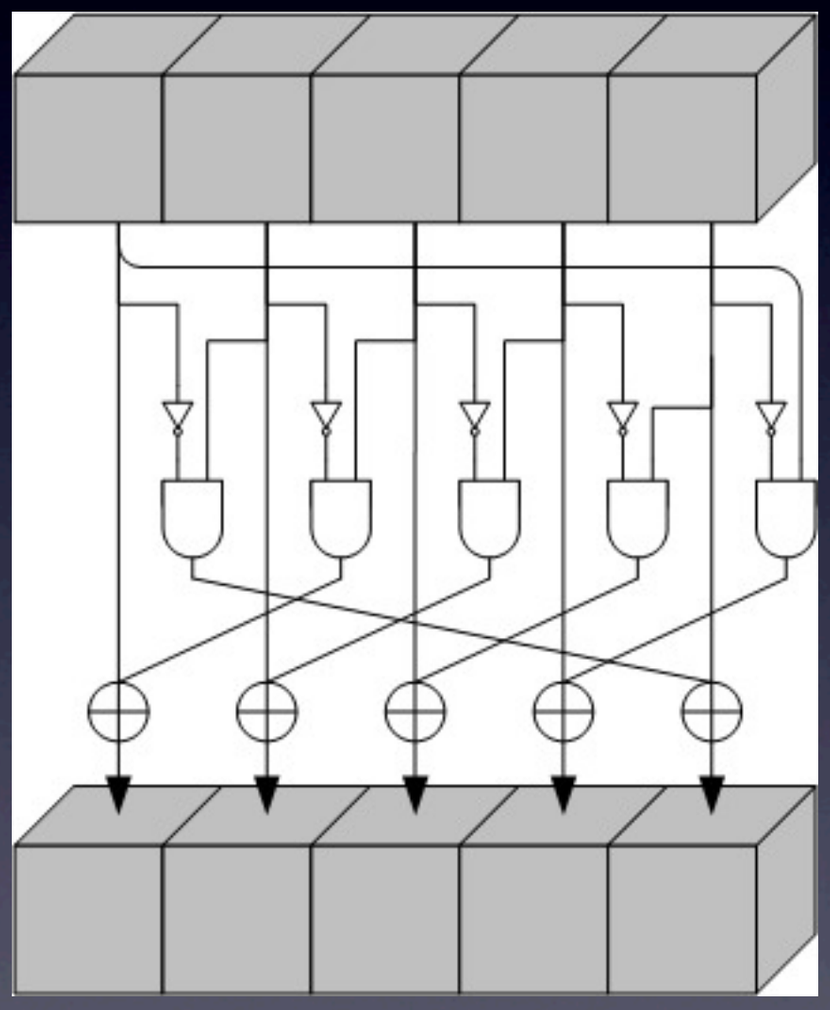
Keccak-f: Theta

```
#Theta step
for x in range(5):
   C[x] = A[x][0]^A[x][1]^A[x][2]^A[x][3]^A[x][4]

for x in range(5):
   D[x] = C[(x-1)%5]^self.rot(C[(x+1)%5],1)

for x in range(5):
   for y in range(5):
      A[x][y] = A[x][y]^D[x]
```


Nicola Apicella
nicapicella<AT>entropia<DOT>de

always first operation
CODE: Python
“high average diffusion and low gate count”

Keccak-f: Chi



Nicola Apicella
nicapicella<AT>entropia<DOT>de

“only non-linear mapping; otherwise, K would be linear”

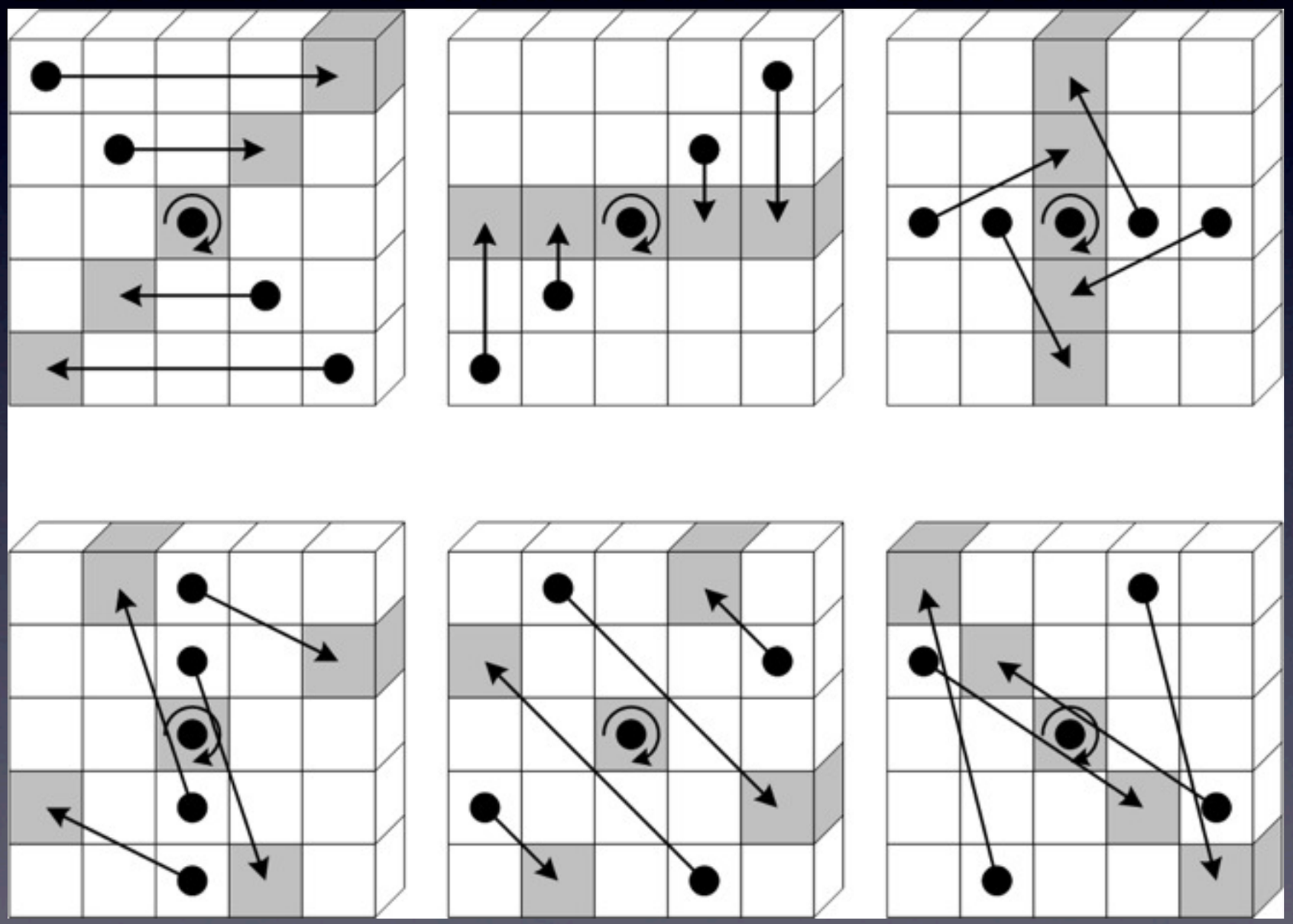
Keccak-f: Chi

```
#Chi step
for x in range(5):
   for y in range(5):
      A[x][y] = B[x][y]^((~B[(x+1)%5][y]) & \
         B[(x+2)%5][y])
```


Nicola Apicella
nicapicella<AT>entropia<DOT>de

"only non-linear mapping; otherwise, K would be linear"

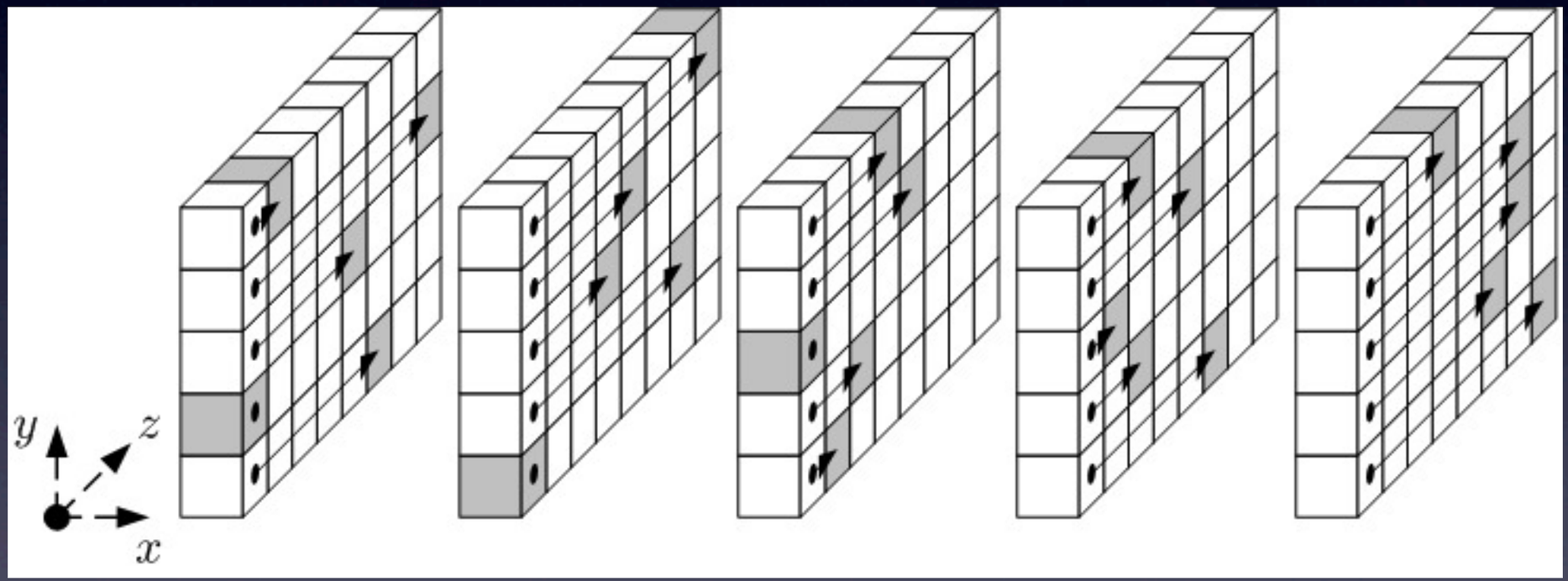
Keccak-f: Pi



Nicola Apicella
nicapicella<AT>entropia<DOT>de

"can be obtained implicitly by addressing"
"dispersion for long term diffusion"

Keccak-f: Rho



Nicola Apicella
nicapicella<AT>entropia<DOT>de

with l = 0 (1 deep): doesn't do a thing.
"otherwise, diffusion within slices would be very low"

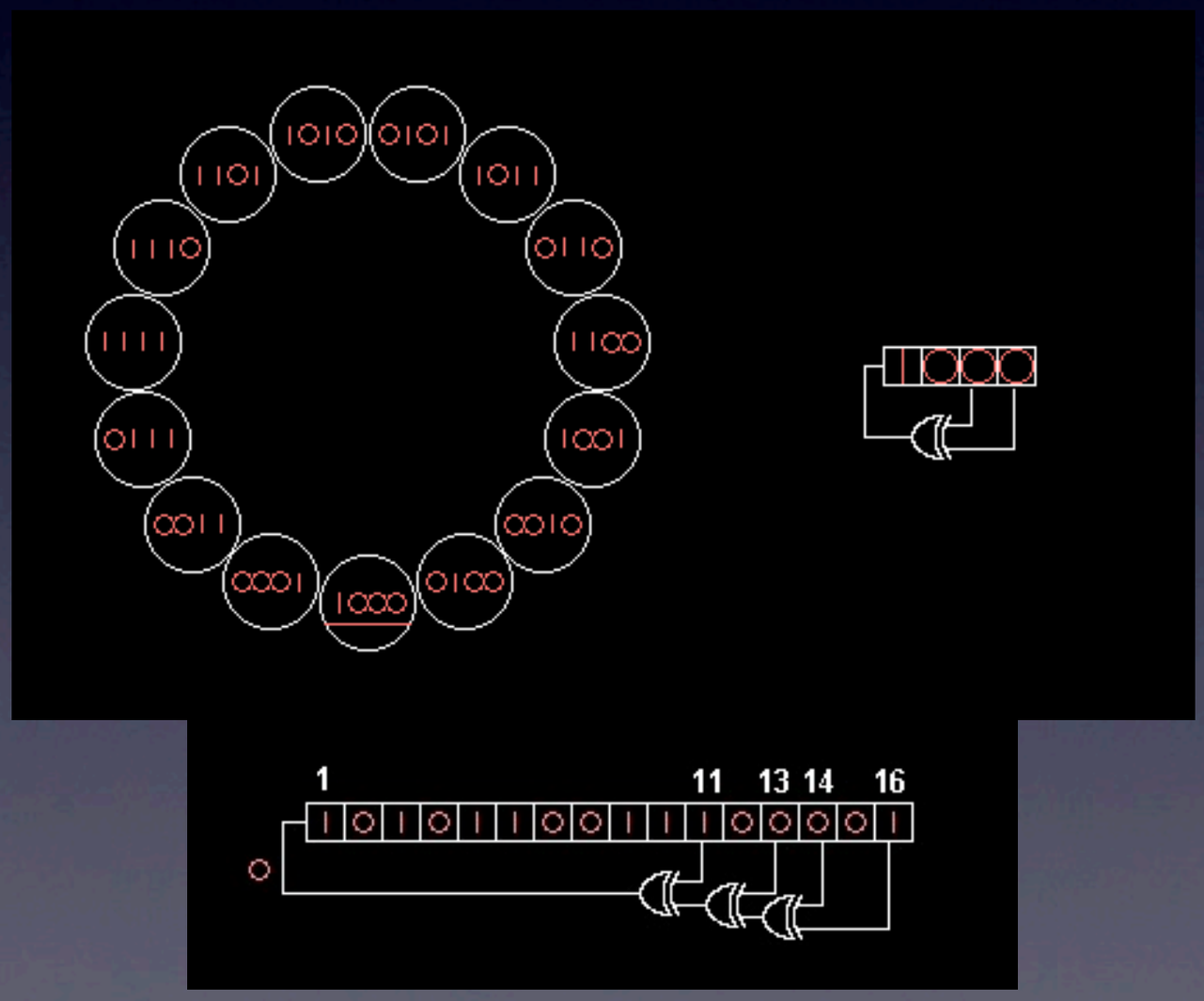
Keccak-f: Rho

```
#Rho and Pi steps
for x in range(5):
   for y in range(5):
      B[y][(2*x+3*y)%5] = self.rot(A[x][y], \
       self.r[x][y])

## Rotation offsets
r=[[0,    36,     3,    41,    18]    ,
   [1,    44,    10,    45,     2]    ,
   [62,    6,    43,    15,    61]    ,
   [28,   55,    25,    21,    56]    ,
   [27,   20,    39,     8,    14]    ]
```


Nicola Apicella
nicapicella<AT>entropia<DOT>de

with l = 0 (1 deep): doesn't do a thing.
"otherwise, diffusion within slices would be very low"

Keccak-f: Iota



Nicola Apicella
nicapicella<AT>entropia<DOT>de

-> simple 24-entry look-up table (12+2*l)

“disrupts symmetries (i.e., no slide attacks possible)”

Keccak-f: Iota

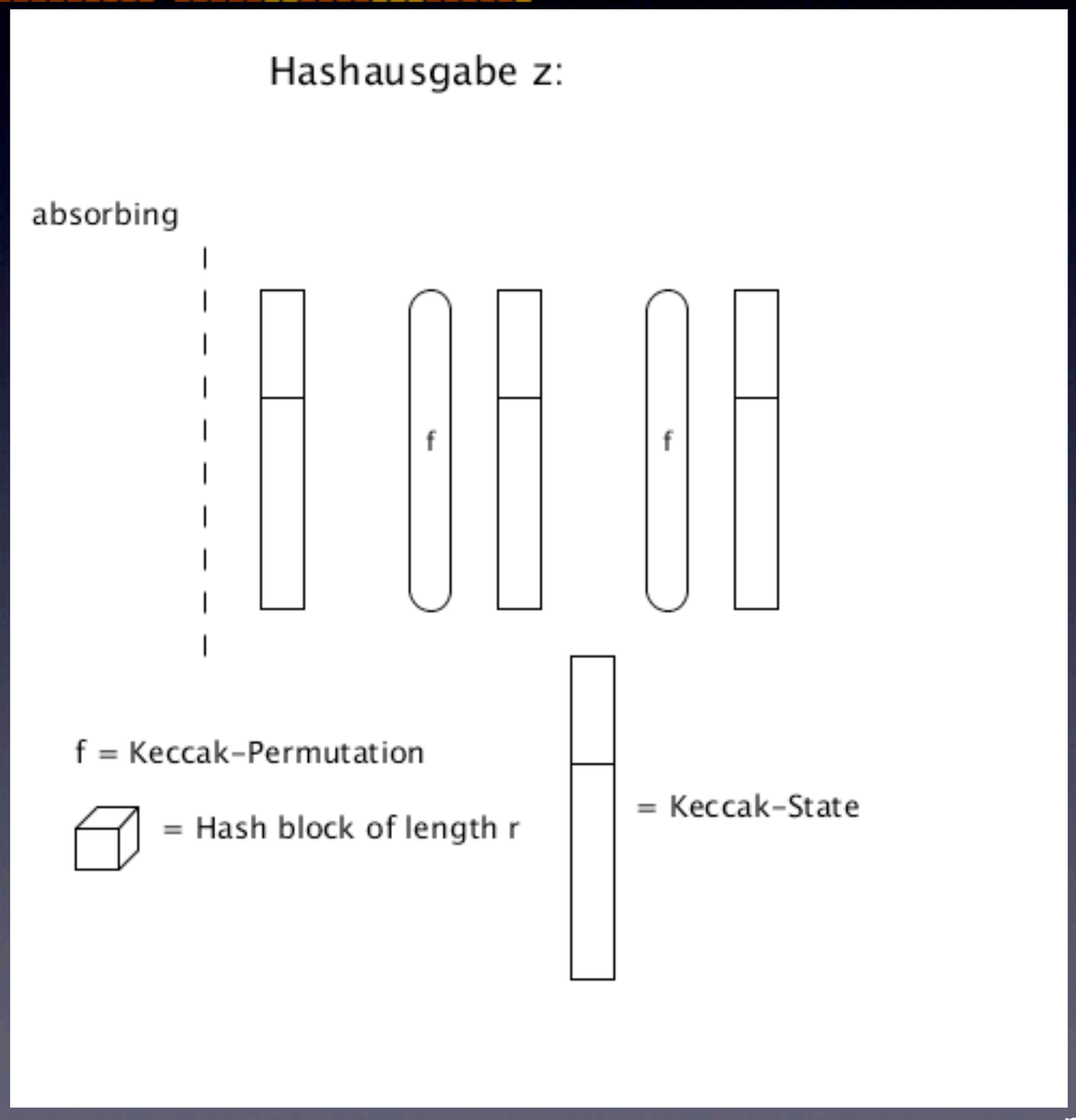
```
#Iota step
A[0][0] = A[0][0]^RCfixed
    ## Round constants
    RC=[0x0000000000000001,0x0000000000008082,
       0x800000000000808A,0x8000000080008000,
       0x000000000000808B,0x0000000080000001,
       0x8000000080008081,0x8000000000008009,
       0x000000000000008A,0x0000000000000088,
       0x0000000080008009,0x000000008000000A,
       0x000000008000808B,0x800000000000008B,
       0x8000000000008089,0x8000000000008003,
       0x8000000000008002,0x8000000000000080,
       0x000000000000800A,0x800000008000000A,
       0x8000000080008081,0x8000000000008080,
       0x0000000080000001,0x8000000080008008]
```


Nicola Apicella
nicapicella<AT>entropia<DOT>de

-> simple 24-entry look-up table (12+2*l)

“disrupts symmetries (i.e., no slide attacks possible)”

Squeeze



28/24 slides
31/May/2013

Nicola Apicella
nicapicella<AT>entropia<DOT>de

…and this is how Keccak works!

Reactions

- Schneier: “NIST should announce no award”
- many critics: just too slow in software
- but: “NIST may not have you in mind”

Nicola Apicella
nicapicella<AT>entropia<DOT>de

NIST is a “hardware company”
good choice with that requirements in mind
hardware performance: EPIC. Only XOR, NOT, AND; 24-entry list + 5*5 lookup-table (triangular numbers)

Outlook

- Regular & salted hashing
- Mask generation function
- Message authentication codes
- Stream encryption
- Single pass authenticated encryption
- Reseedable pseudorandom sequence generator

Nicola Apicella
nicapicella<AT>entropia<DOT>de

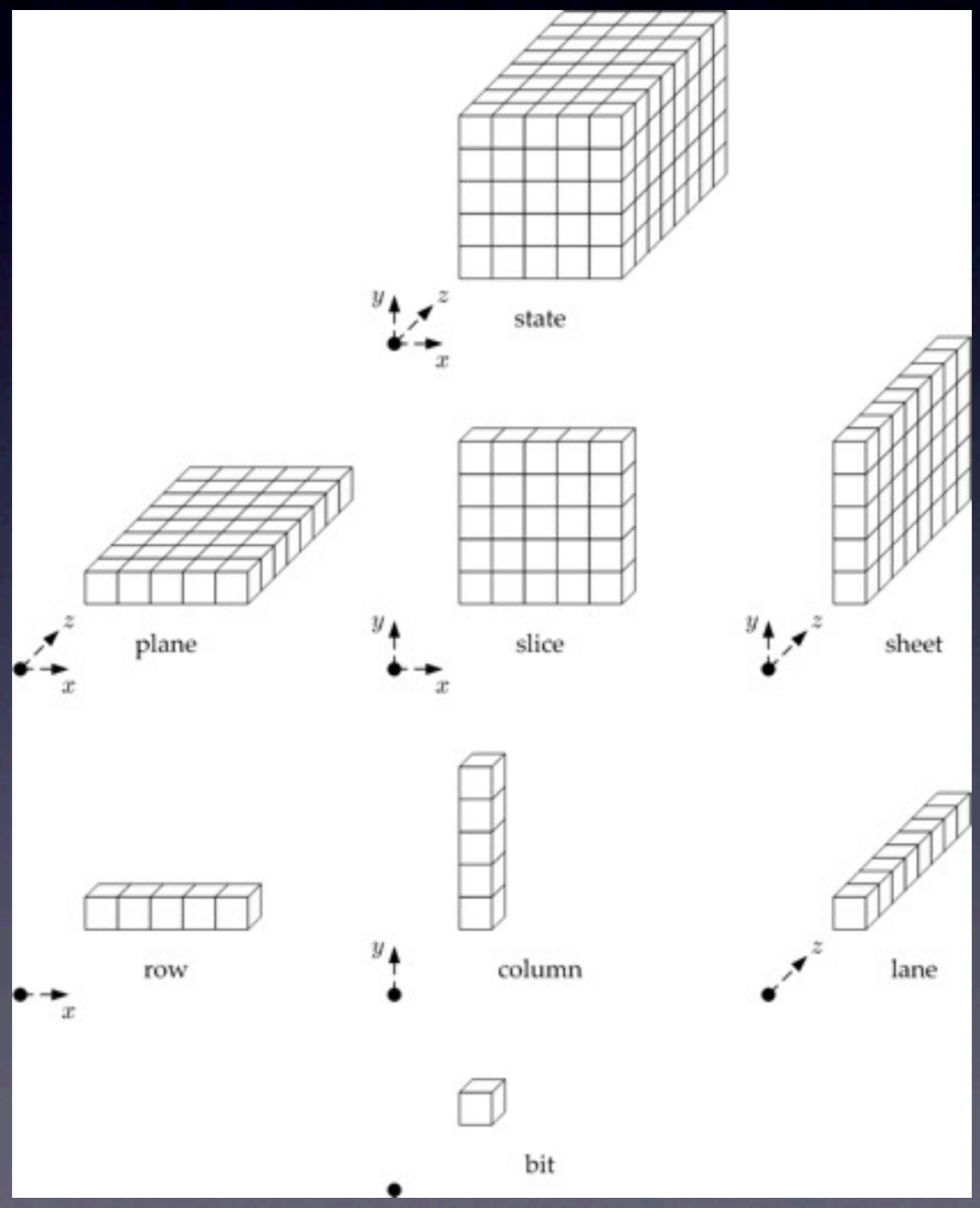
Keccak can do a lot more!
Standard is coming: FIPS 180-5
draft expected 2013 Q3, final - 2014 Q2 (secretary of commerce signature)

Acknowledgements

- Jiska, Sprawl, JackMcCrack
- Wikipedia
  - black background images: PD / CC-BY-SA Cuddlyable3;
    Merkle-Damgård: PD; Wide-Pipe: CC-BY-SA Watchdog9
- http://keccak.noekeon.org/
  - static images: CC-BY
- http://celan.informatik.uni-oldenburg.de/kryptos/info/keccak/overview/

Nicola Apicella
nicapicella<AT>entropia<DOT>de

Thanks! Questions?



Nicola Apicella
nicapicella<AT>entropia<DOT>de