Introduction to cryptographic hash functions

Krystian Matusiewicz

Centre for Advanced Computing Algorithms and Cryptography,
Department of Computing, Macquarie University

NTU Seminar, 5 October 2007

Outline

Outline

Hash functions

Definition

A *hash function* is a function that maps strings of arbitrary length to strings of fixed length, i.e. any function

$$h : \{0,1\}^* \to \{0,1\}^n$$

for some fixed n called the hash length.

Example

A modulo operator can be seen as a simple example of a hash function. For any bitstring M we can see it as a number and compute $h(M) = M \mod q$.

Hash functions : popular uses

Hash functions (ordinary) are ubiquitous tools of computer science and engineering.

- hash tables
- pattern matching
- checksum algorithms

In all those applications the essential property is that the hash value is a short “tag” or “fingerprint” of possibly long data.

$$h(M_1) \neq h(M_2) \overset{\text{always}}{\longrightarrow} M_1 \neq M_2$$

$$h(M_1) = h(M_2) \overset{\text{with high Prob.}}{\longrightarrow} M_1 = M_2$$

Collisions

Definition

A *collision* for the hash function h is a pair of different messages M, M' that have the same hash value $h(M) = h(M')$.

- Since the domain is bigger than the range, collisions necessarily exist.
- Quite often we don't care as long as the probability of a collision is low.
- However, low probability of a collision for random inputs does not mean they cannot be found easily.

Constructing collisions for mod operator

Example

For a modulo operator $\mod q$, the probability that a random pair of numbers will yield the same value is $1/q$. But we can easily construct different messages which collide.

$$x, x+q, x+2q, x+3q, \ldots$$

There are bad people out there...

This ability to construct different messages that collide under the hash function may be exploited by malicious persons to attack a system.

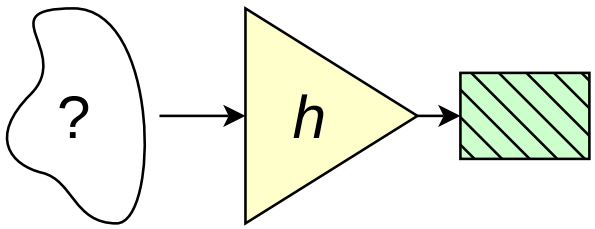
- some intrusion detection systems use hash tables – by crafting inputs that degenerate hash tables, it may be possible to launch a DoS attack
- peer-to-peer networks use hash values to identify chunks of data. By producing many fake chunks that have the same digest as original ones, one can “poison” p2p network. [cf. UUHash case]

Stronger notion: cryptographic hash functions

This naturally leads to the definition of a cryptographic hash function, i.e. a hash function that satisfies the following security requirements:

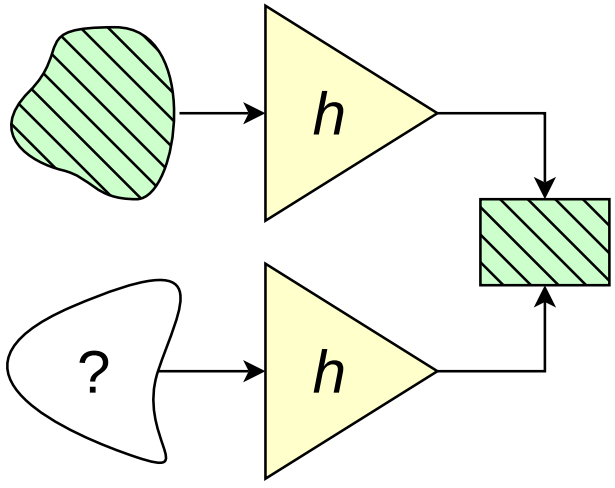
- preimage resistance
- second-preimage resistance
- collision resistance

Preimage resistance



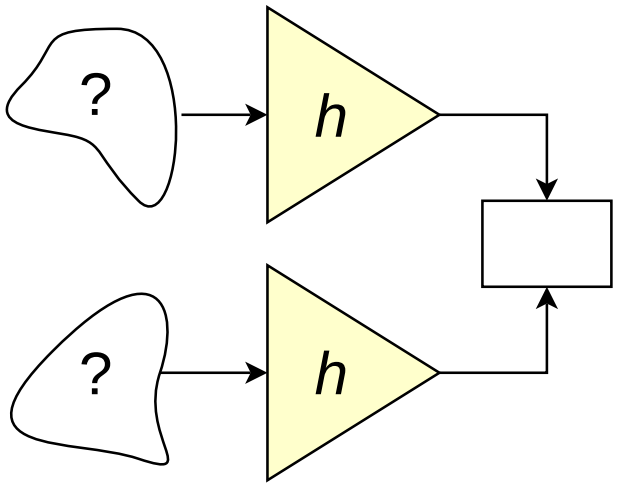
Preimage resistant : Given an output Y of the hash function it is difficult to find any *preimage* - an input X such that $h(X) = Y$.

Second preimage resistance



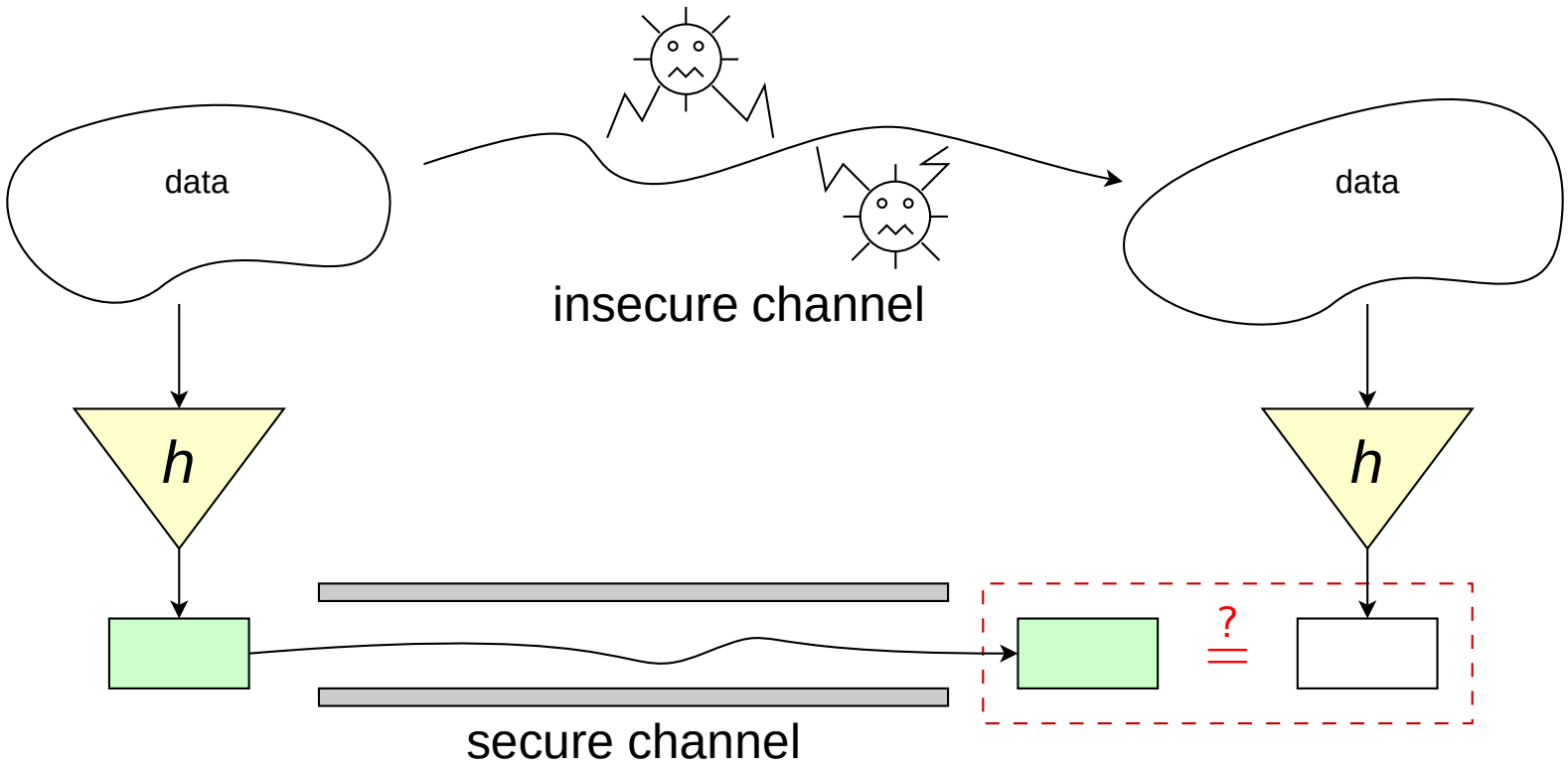
Second preimage resistant : Given a fixed input X to the hash function and the corresponding output $h(X)$ it is difficult to find a *second preimage* - another input X', $X' \neq X$ such that $h(X) = h(X')$.

Collision resistance



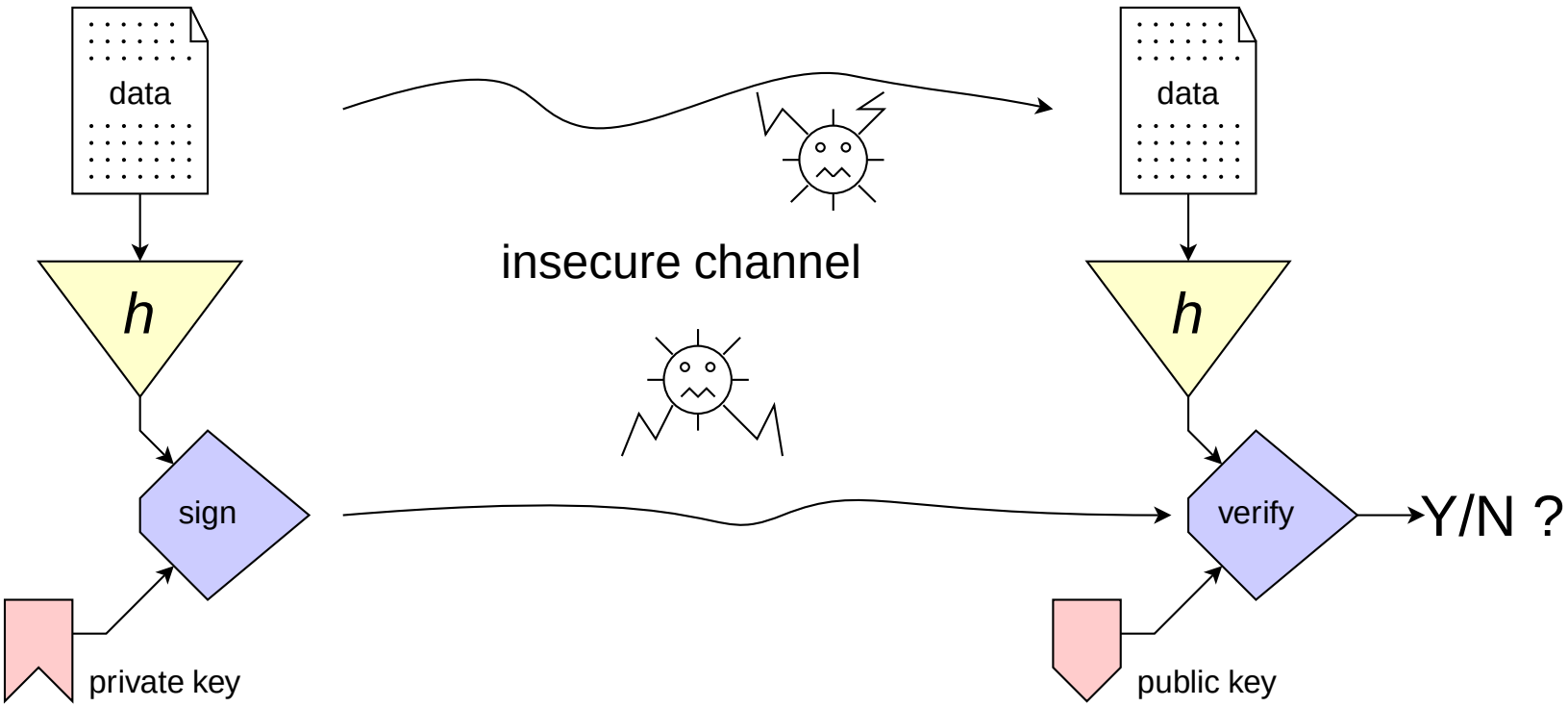
Collision resistant : It is hard to find any pair of distinct messages (X, X'), $X \neq X'$ such that $h(X) = h(X')$.

Applications: Data integrity



Comparing the digest of data sent over insecure communication channel with securely obtained original digest allows to verify integrity of the data.

Applications: digital signatures



Digital signature schemes with appendix use cryptographic hash function *h* to obtain the digest of the data which is later signed.

Outline

The three classical definitions

- Preimage resistant : Given an output Y of the hash function it is difficult to find any *preimage* - an input X such that $h(X) = Y$.
- Second preimage resistant : Given a fixed input X to the hash function and the corresponding output $h(X)$ it is difficult to find a *second preimage* - another input X', $X' \neq X$ such that $h(X) = h(X')$.
- Collision resistant : It is hard to find any pair of distinct messages (X, X'), $X \neq X'$ such that $h(X) = h(X')$.

Problems with more formal definitions

What does “hard” or “difficult” mean?

Two rigorous models of computational complexity:

- Asymptotic complexity (classical complexity theory)
- Concrete security

Classical complexity theory

- Define a family of functions where the security parameter (e.g. digest length) corresponds to “instance size”
- Prove that the complexity function (function that expresses the amount of work necessary to break the algorithm as the function of the security parameter) has desired asymptotic behaviour (e.g. grows exponentially).

Classical complexity theory – some problems

- Worst case complexity is often not enough for cryptographic purposes.
- How is asymptotic complexity related to “real world” expectations? Example: knapsack problem is known to be NP-hard, but knapsack cryptosystems were broken for proposed instance sizes.

Concrete security

Developed as an alternative to asymptotic analysis.

- define a finite family of functions indexed by the key $k \in \mathcal{K}$
- introduce an adversary, i.e. someone who wants to attack the system and define the winning conditions for him
- define a computational experiment in which adversary takes part (a sequence of steps requiring some interaction with the adversary)
- measure the *advantage* of the adversary, i.e. the probability that he wins the game taken over all random choices of the key k
- if no reasonable adversary can achieve significant advantage, the system is secure

Concrete security definition of collision resistance

Definition (Rogaway, Shrimpton)

Let $H : \mathcal{K} \times \mathcal{M} \to \{0,1\}^n$ be a hash-function family and let A be an adversary. Then we define

$$\mathbf{Adv}_H^{Coll}(A) =$$

$$Pr[K \xleftarrow{\$} \mathcal{K}; (M, M') \xleftarrow{\$} A(K) \ : \ (M \neq M') \wedge H_K(M) = H_K(M')].$$

In the above formula $M \xleftarrow{\$} \mathcal{S}$ denotes choosing a random element from the distribution $\mathcal{S}$ and calling it M.

In practice...

- Concrete security setting needs a family of functions.
- In practice, we have only a single, fixed instance of the function.
- The only security framework we can talk about is *human ignorance model* [Rogaway, 2006]
- But this is not satisfactory for the designers of cryptographic primitives.

More requirements

Apart from the three fundamental properties, quite often more is implicitly required from a cryptographic hash function.

- “hash function should mimic a random function” – no detectable structure
  - simulation of random oracles – e.g. for RSA-OAEP, RSA-PSS
  - pseudorandom function families (PRFs) – e.g. for HMAC
- security properties of truncated variants – e.g. SHA-224
- regularity of the hash function – influences birthday attacks
- others ???

Outline

Iterative hash functions

- It's difficult to design a function that accepts inputs of arbitrary length.
- The natural solution is to use some kind of an iterative procedure to process data in chunks.
- use a compression function that "shrinks" input

Definition

A *compression function* is a function

$$f : \{0,1\}^{m+n} \rightarrow \{0,1\}^{n}$$

where $m > 0$.

Merkle-Damgård construction

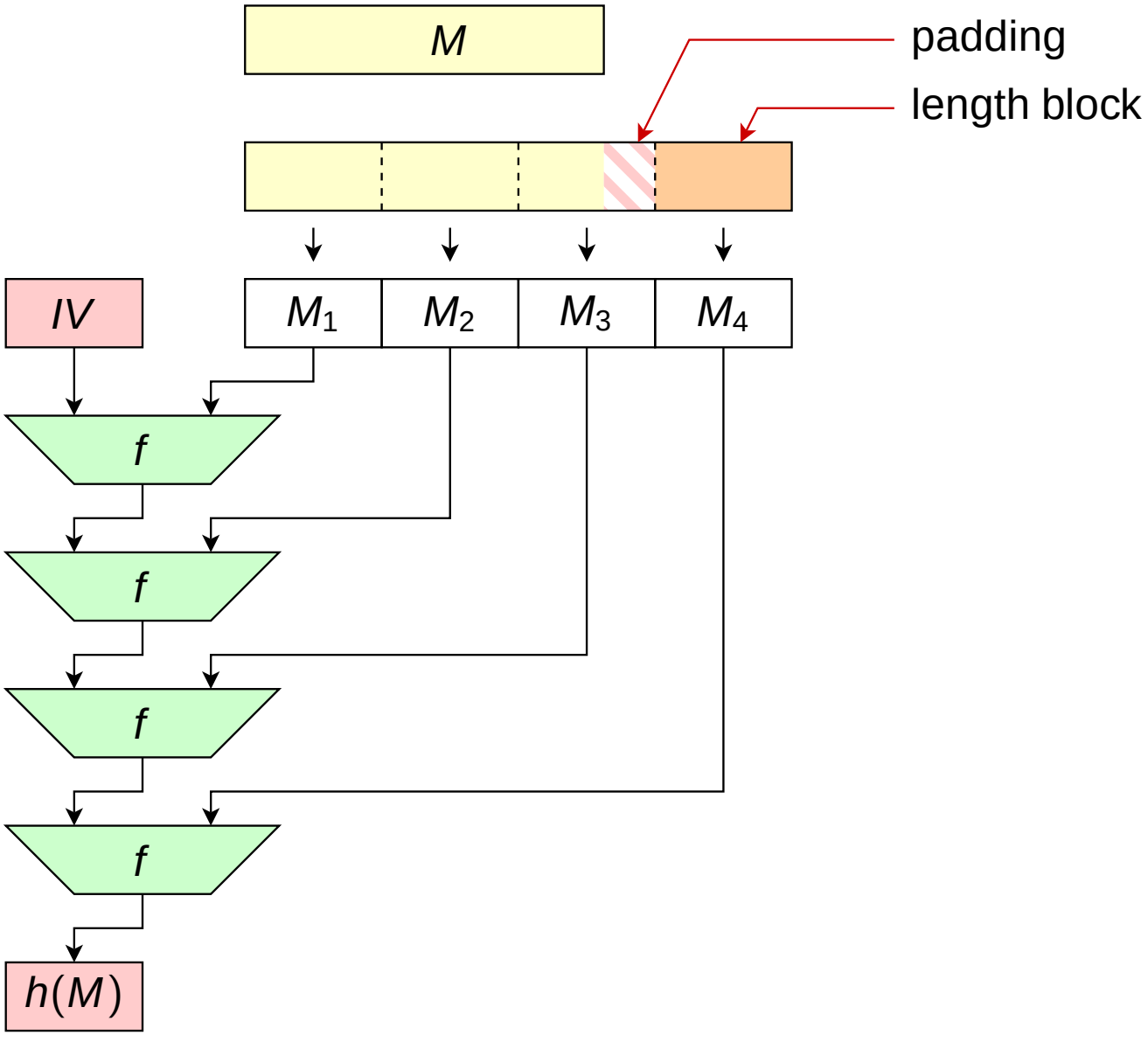
INPUT: message $M \in \{0,1\}^*$
OUTPUT: digest $h(M) \in \{0,1\}^n$

- pad the message M to a multiple of block length.
- (optionally) Append a block of m bits with encoded initial length of the message M.
- Split M into blocks $M = (M_0, M_1, \ldots, M_b)$.
- Perform the iterative compression

  $H_0 := IV$ $\quad$ {quite often $IV = 0^b$}
  for $i = 0$ **to** b **do**
  $\quad H_{i+1} := f(H_i || M_i)$
  end for

- Return the result $h(M) = H_{b+1}$.

One picture worth a thousand words: Merkle-Damgård



Security of iterated hash functions

The fundamental question

What is the relation between security properties of the iterated hash function and the security properties of the compression function itself?

- MD construction comes with a reduction proof of collision resistance: as long as the compression function is collision resistant, the hash function is collision resistant [Merkle-Damgård]
- But what about the other properties? – The situation here is more complex...

Drawbacks of the iterated construction

- multicollisions [Joux 2004]
- second preimages faster than 2^n [Dean 1999; Kelsey, Schneier 2005]
- iteration does not preserve balance [Bellare, Kohno 2004]

Alternatives to MD iteration mode

Recent alternatives that aim at improving properties of the iterative constructions.

- Wide-pipe strategy [Lucks 2005]
- Randomized Hashing [Halevi, Krawczyk 2006]

$$H_{i+i} := h(H_i \,||\, M_i \oplus r) \ .$$

- HAIFA [Biham, Dunkelman 2007]

$$H_{i+1} := f(H_i \,||\, M_i \,||\, \#bits \,||\, salt) \ .$$

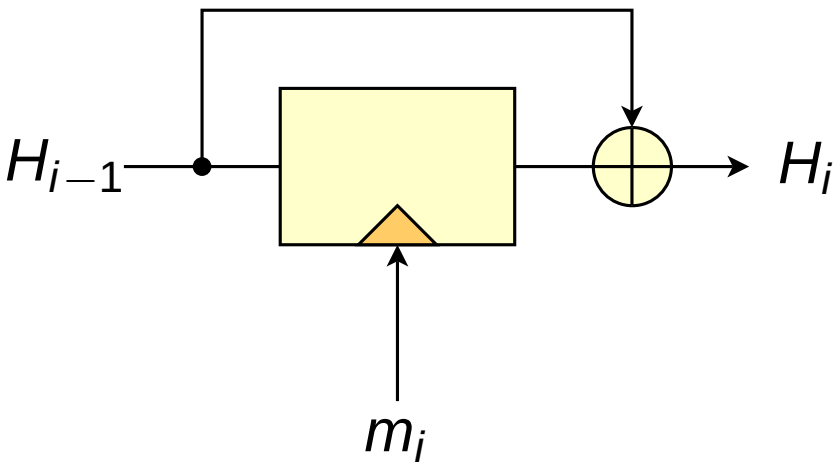
How to build a good compression function?

Three main approaches:

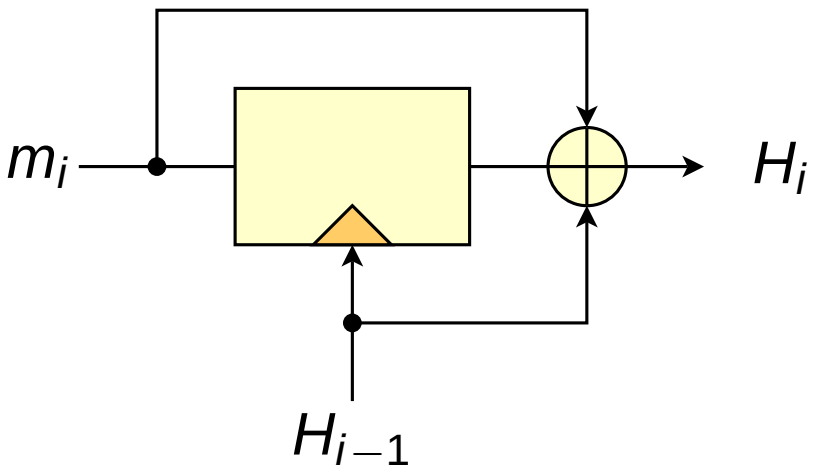
- Compression functions based on block ciphers
- Dedicated heuristic designs
- Constructions with security reductions

Cipher-based compression functions

Assume we have a block cipher $E : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$.
We can use it to construct a compression function.



Davies-Meyer mode



Miyaguchi-Preneel mode

So called “PGV modes” [Preneel, Govaerts, Vanderwalle 1993]

Cipher-based compression functions

The good:

- come with security reductions: if you have a good block cipher, you can have a good compression function [Black, Rogaway, Shrimpton 2002]

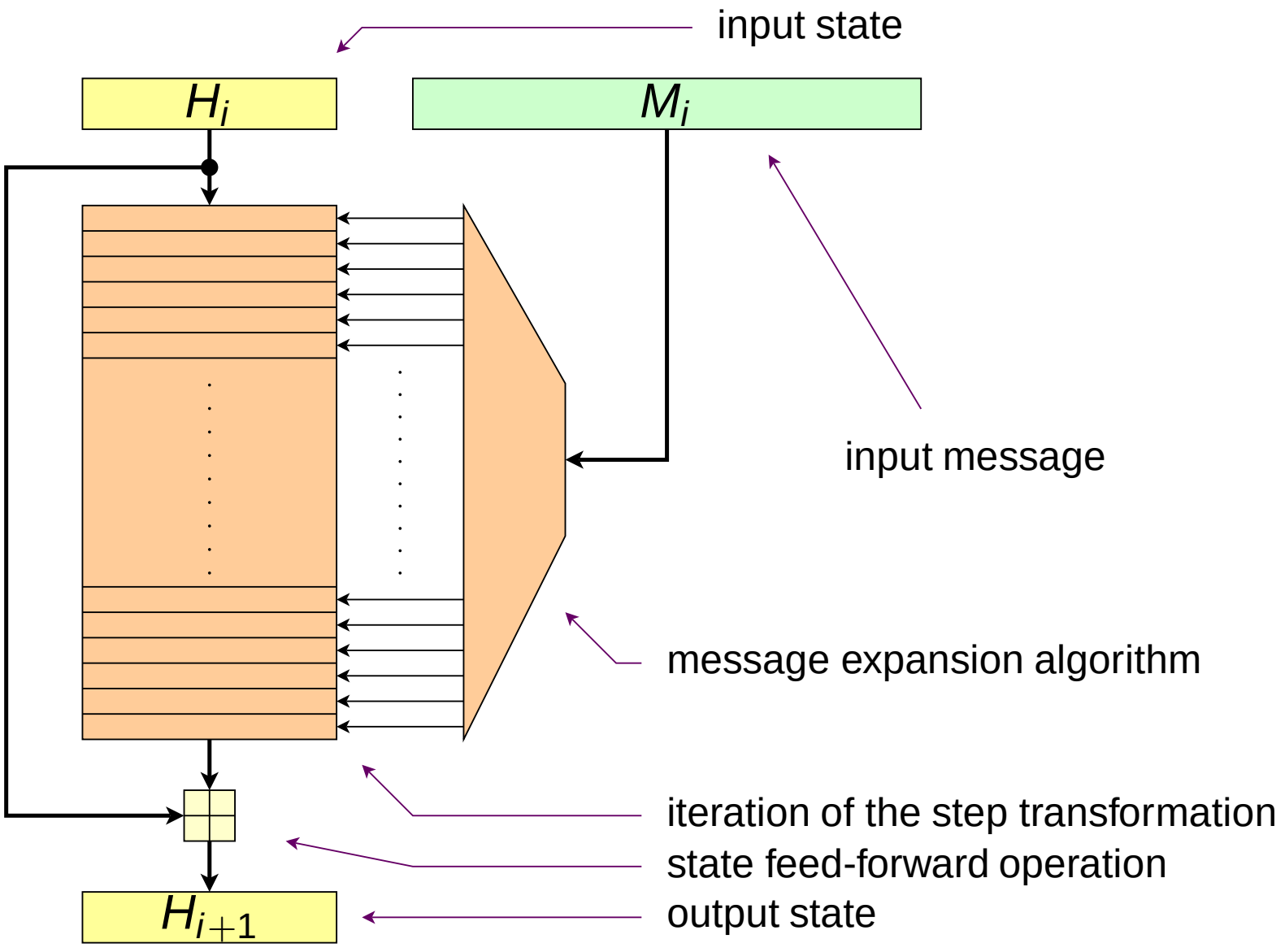
The bad:

- Cipher-based hashes are too slow for some...
- The block size of most popular ciphers is too small to produce digests of appropriate length (solutions: design a specialized block cipher, like Whirlpool or use double block-length modes)
- Do we want to put all the eggs in just one basket?

Dedicated constructions

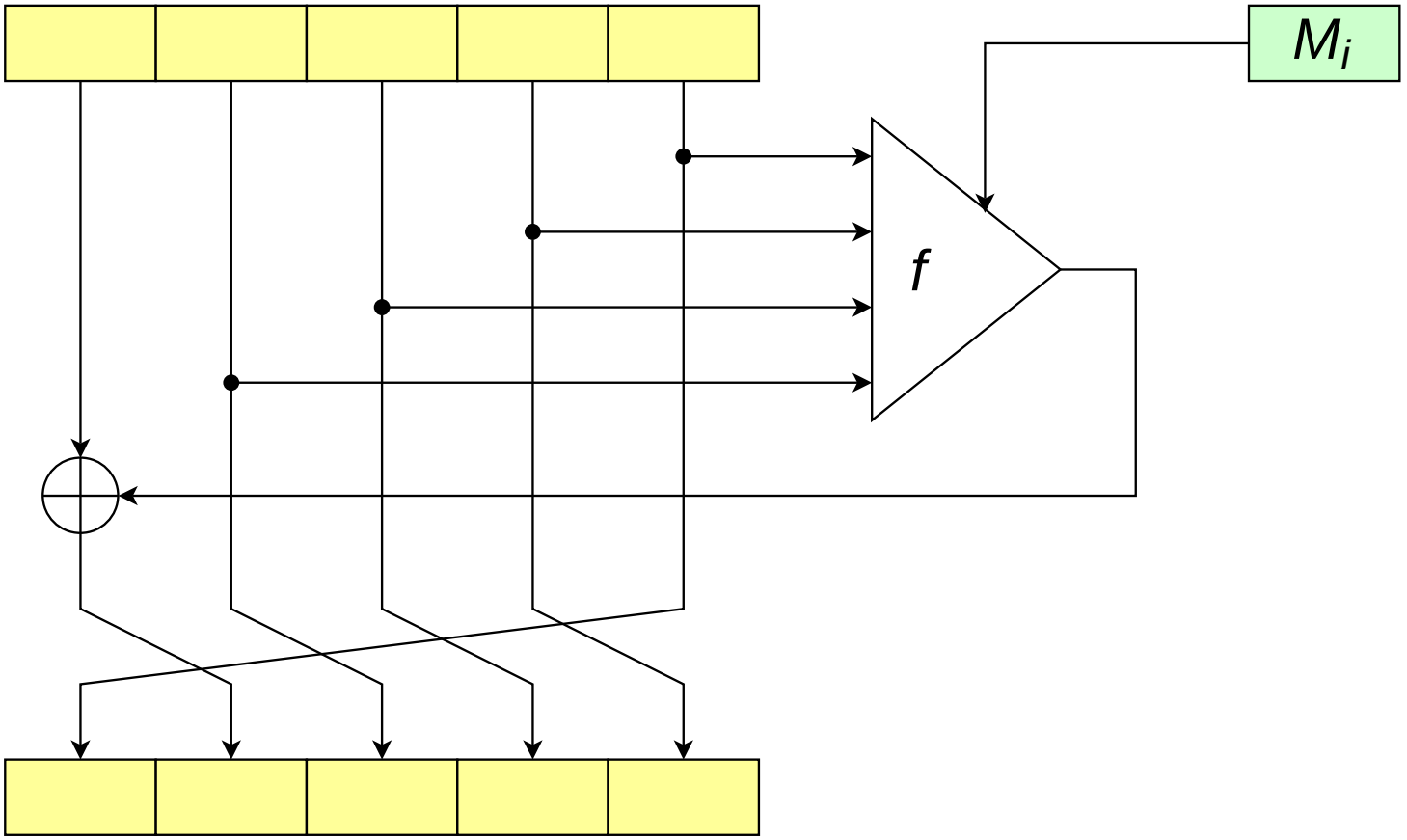
- Compression functions designed “from scratch”
- Most popular approach in practice: Davies-Meyer mode but with specially crafted ciphers: still secure (?) but faster than mainstream block ciphers
- MD4, MD5, HAVAL, HAS-V, SHA-1, SHA-2, (and many others) are designed that way

Dedicated designs: overview



Dedicated designs: step transformation in MD designs

MD = Message Digest, a family of designs influenced by Ron Rivest's MD4 and MD5 functions.



Constructions with security reductions

Principle:

- Functions based on some difficult computational problems
- Breaking the function implies solving some difficult problem – security reduction “provable security”

Examples:

- VSH (Very Smooth Hash) [provably collision-resistant]
- MQ-Hash [provably one-way]

“Provably secure” designs – some issues

- How tight is the security reduction?
- Is the difficult problem really difficult (what about difficulty on average, for some special cases etc)
- Looks like all such constructions need quite long hashes to achieve good security level
- At the moment MUCH slower than dedicated designs

Outline

Current situation

- Vast majority of functions used in practice are dedicated heuristic designs
- Wang et al showed how to attack such constructions
- Revived interest in cryptographic hash functions
- A lot of new research into cryptanalysis of hash functions in recent years

Attacks of Wang et al and some improvements

- Look for differentials using both XOR differences (over $\mathbb{F}_2$) and modular differences (over $\mathbb{Z}_{2^{32}}$). In other words, trace signed binary differences.
- used to break MD4, MD5, HAVAL and SHA-1 (theoretically).
- Improved algorithms, especially automatisation of path finding
  - Coding-theory tools
  - Branch and prune searches
- More fine-grained conditions for automatic path-finding [Rechberger, De Cannière 2006]

The Hash Function Lounge (or a graveyard?)

“The Hash Function Lounge” – a webpage maintained by P. Barreto that lists cryptographic hash functions together with attacks

 theoretical attack on the function

 practical attack on the full function

Recently, there are many, many new skulls there ...

Outline

Replacing the currently used hash functions

- Replacing currently deployed functions is an enormous task.
- We have to be sure that the change will be for better.
- We need to know how to design secure cryptographic hash functions
  - what requirements do we need
  - how to construct functions satisfying them

NIST competition for AHS

- Public competition for the Advanced Hashing Standard
- Something similar to the AES process
- Announced in Federal Register
- Tentative Timeline definitely not feasible

Proposed Draft Minimum Acceptability Requirements

A.1 The algorithm must be publicly disclosed and available on a worldwide, non-exclusive, royalty-free basis.

A.2 The algorithm must be implementable in a wide range of hardware and software platforms.

A.3 The algorithm must support 224, 256, 384, and 512-bit message digests, and must support a maximum message length of at least 2^{64} bits

Proposed Draft Evaluation Criteria

- Security,
- Computational efficiency,
- Memory requirements,
- Hardware and software suitability,
- Simplicity,
- Flexibility, and
- Licensing requirements.

Evaluation Criteria : C.1 Security

- The actual security provided by the algorithm as compared to other submitted algorithms (of the same hash length), including (but not limited to) first and second preimage resistance, collision resistance, and resistance to generic attacks (e.g., length extension).
- The extent to which the algorithm output is indistinguishable from a random oracle.
- The soundness of the mathematical basis for the algorithms security.
- Other security factors raised by the public during the evaluation process ...

Outline

Conclusions

- Cryptographic hash functions are ubiquitous tools of modern cryptography
- Although basics are simple, there is a lot of subtle properties and relationships there
- A lot more research is needed to understand enough to propose a new hashing standard that would last
- It is a really exciting topic to work on!

Thank you!