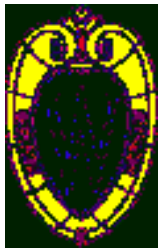


CS/ECE/Math 435
Introduction to Cryptography

Professor Chris DeMarco
Department of Electrical & Computer Engineering
University of Wisconsin-Madison

Spring Semester 2021

University of Wisconsin-Madison
Department of Electrical and Computer Engineering
CS/ECE/Math 435 - Introduction to Cryptography, Spring Semester 2021
Final Exam Structure and Coverage Information

The 435 final exam is scheduled for Sunday, May 2, 2019, with nominally scheduled time slot of 10:05 AM –12:05 PM.

The exam will be closed book, closed notes, with a formula sheet provided. Allowed work time for the exam will be two hours. Given the exam's on-line format, students will use a tablet or computer and web access to complete the exam. A basic scientific calculator (particularly for calculating logarithms and exponentiation) will also be necessary. All other devices (cell phones, additional tablets or computers) should be inaccessible during the exam, unless students have McBurney Center accommodation allowing an exception. The exam will NOT require use of MATLAB, and there will be no problems oriented toward MATLAB-specific syntax. Computation able to be performed on a scientific calculator will be adequate for all exam problems. However, there may be questions requiring ***description*** of a general algorithmic approach to solution.

The exam structure will be very close to that of the midterm, with three long format problems (accounting for 70% of exam grade), and five short-answer problems of six points each (accounting for 30% of exam grade). In answer to students' concerns following the midterm, some of the short answer questions will be broken into two parts at three points for each part, to reduce the amount of "all-or-nothing" credit associated with a single problem.

Final exam coverage will be cumulative, but you may expect somewhat greater emphasis on new material not covered on the midterm. While a given problem may mix some old and new material, you may expect two long format problems to focus primarily on new material post-midterm, and one long format problem to focus primarily on topics from earlier in the semester.
.

On-Line Exam Logistics and Proctoring: Like the midterm, **Honorlock will NOT be used for the 435 final exam** (contrary to the original plan stated in the January course syllabus handout).

Unless a special in-advance arrangement for an alternative or make-up exam for a student is approved by Prof. DeMarco on or before 5 PM Wednesday, April 26, all students will be expected to complete the exam on-line in the designated time window on **Sunday, May 2, 2019**. The time window for the exam on Canvas will open at 9:35 AM, and close at 12:35 PM (i.e., the nominally scheduled 10:05-12:05 window, plus an added ½ hour before, and ½ hour after). Once a student initiates work on the exam in Canvas, they will be limited to two hours to complete it. The ½ hour extensions are intended only to allow flexibility in a student's exact start time, and avoid all students accessing Canvas precisely at 10:05 AM. Note that you should begin the exam no later than 10:35 AM to have the full two-hour work period.

Formula Sheet: For the final exam, students will be provided a copy of the midterm exam formula sheet, plus the additional material below (all will be provided as excerpts from the Bach course notes): i) Statement of Fermat's Little Theorem, Bach notes p. 19-1; ii) Pseudo-code of Berlekamp-Massey algorithm, Bach notes p. 22-3; iii) Feistel mapping and its inverse, Bach notes bottom p. 26-1 & top p. 26-2; iv) formulas for expected number of matches in Classic Birthday Problem and two-sample Birthday problem, Bach notes p. 28-1; v) Theorem stating condition for g to be a generator/primitive root for $\mathbf{Z}_p^*$, Bach notes p. 38-1.

Cryptographic Hash Functions

Generally, hash functions take bit strings of arbitrary length "*" to bit strings of specified length "n." Typically assume * > n as the useful case.

Motivation in Digital Signature Context

IF desirable properties of digital signatures can be maintained, we'd like to use previously described algorithms (suitable only for $m \in \mathbb{Z}_{p-1}$, so n must satisfy $2^n \leq p$)

We'll start from list of properties of hash function itself, and then argue that these result in desirable properties for "shortcut" digital signature.

Hash function h must satisfy:

0) $h: \mathbb{Z}_2^* \rightarrow \mathbb{Z}_2^n$

($\mathbb{Z}_2^*$: again, denotes arbitrary integer $> n$)

Then desirable properties of h:

1) "Forward" computation, given $x \in \mathbb{Z}_2^*$ compute $y = h(x)$, should have "low" computational cost

2) Reverse, pre-image computation should have high computational cost:

given $y \in \mathbb{Z}_2^n$, find an $x \in \mathbb{Z}_2^*$ s.t. $h(x) = y$ [note: because dimension * > dimension n, we can not expect h invertible; two distinct x, x' may map to single y]

3) Collision identification should have high computational cost; i.e. problem of identifying pair $x, x' \in \mathbb{Z}_2^*$, $x \neq x'$, s.t. $h(x') = h(x)$.

4) Second pre-image identification should have high computational cost: Given known x, y pair satisfying $y = h(x)$, compute second $x' \neq x$ s.t. $y = h(x')$.

Hash terminology: x is "message", $h(x)$ is "hash image" or "message digest."

Related observations

i) We'll typically assume $h(\cdot)$ is onto; that every $y \in \mathbb{Z}_2^n$ has at least one $x \in \mathbb{Z}_2^*$ s.t. $y = h(x)$.

ii) And given $x \in \mathbb{Z}_2^*$, sets of form

$$C_x = \{ x' \mid h(x') = h(x) \}$$

partion the set $\mathbb{Z}_2^*$ into equivalence classes; in terminology of hash functions, these C_x sets termed "fibers."

Cryptographic Hash Functions typically implemented by finite-state automata: known as Merkle-Damgard construction.

In particular, as "stepping stone" towards desired hash function, we first define a transition function

$$\delta : \underbrace{Z_2^n}_{\text{set consistent with that in which our hash outputs reside; Bach denotes "Q"}} \times \overbrace{Z_2^m}^{\text{additional finite set; } m < * \text{ Bach denotes } \Sigma} \rightarrow Z_2^n$$

With our assumption of $* > m$, partition $X \in \mathbb{Z}_2^*$ into ℓ blocks of length m (as usual, one can "pad out" last block if $*$ not integer multiple of m).

$$X = [X_0, X_1, \cdots X_{\ell-1}]$$

X_i's $\in \mathbb{Z}_2^m$ will play role of inputs to our finite state machine at each "time" $i = 0, 1, 2 \cdots$;

Q_i's $\in \mathbb{Z}_2^n$ are our "states."

More practically secure δ functions and finite state machines (FSM) will follow shortly. However, to illustrate, consider a linear case:

Let $A \in \mathbb{Z}_2^{n \times n}$, $B \in \mathbb{Z}_2^{n \times m}$

Given input $\underline{x} \in \mathbb{Z}_2^*$, partitioned as $\underline{x} = [\underline{x}_0; \underline{x}_1; \underline{x}_2 \ldots x_{l-1}]$. Note that value of l is dependent on length of original input, the wildcard "*" (so l will change for different input $\underline{x}$). Each $\underline{x}_i \in \mathbb{Z}_2^m$

An initial "state" $\underline{q}_0 \in \mathbb{Z}_2^n$ is selected (often simply random selection).

Then a "linear" FSM may be constructed

$$\underline{q}_{i+1} = \underbrace{\text{mod}\left[\left(A \cdot q_i + B \cdot x_i\right), 2\right]}_{i = 0, 1, 2, \ldots, \ell-1}$$

this function plays the role of $\delta(q_i, x_i)$

CAUTION: Notation for indexing above is modified slightly from Bach notes p. 43-2

For this linear FSM, observe

$$\underline{q}_1 = \underbrace{\mathrm{mod}\left(A\underline{q}_0 + B\cdot\underline{x}_0, 2\right)}$$

$$\underline{q}_2 = \mathrm{mod}\left[A\cdot\left(\underline{q}_1\right) + B\cdot\underline{x}_1, 2\right]$$

$$\Rightarrow \underline{q}_\ell = \mathrm{mod}\left[\left(A^\ell\underline{q}_0 + \sum_{k=0}^{\ell-1} A^k B\underline{x}_k\right), 2\right]$$

And recall, $\underline{q}_\ell$ serves as output of hash function, for input $[\underline{x}_0, \underline{x}_1 \cdots \underline{x}_{\ell-1}]$.

Observe dependence on initial state $\underline{q}_0$: We get a family of hash functions.

Much like the case of encryption functions, use of a linear function in hashing lessens the "security" of the scheme.

In hash functions, security equates to our (2)-(4) desirable properties

(2) High computational cost for pre-image;

(3) " " " for collision identification

(4) " " " for second preimage identification

And, much like DES and AES encryption, desired nonlinear & function often constructed by multiple "rounds"; that is, & is a composition of functions, with at least some of these building block functions composing & being nonlinear Boolean functions (remember, all our computations involve binary quantities)

Illustration: Again, the U.S. Institute of Standards and Technology (NIST) publishes detailed standards describing algorithms for encryption and hashing.

While better, more secure standards have since eclipsed it, the MD4 hashing standard is simple enough to be summarized here

{ M.D. acronym = "Message Digest" }

More secure, modern standards include SHA-1 and SHA-2

Journal Article

New Second-Preimage Attacks on Hash Functions

Published: June 23, 2015
Citation: *Journal of Cryptology* (June 23, 2015) pp. 1-40

Author(s)

Elena Andreeva (KU Leuven), Charles Bouillaguet (Université Lille 1), Orr Dunkelman (University of Haifa), Pierre-Alain Fouque (Université Rennes 1), Jonathan Hoch (Weizmann Institute of Science), John Kelsey (NIST), Adi Shamir (Weizmann Institute of Science), Sébastien Zimmer (École Normale Supérieure - CNRS)

Announcement

Abstract

In this work, we present several new generic second-preimage attacks on hash functions. Our first attack is based on the herding attack and applies to various Merkle–Damgård-based iterative hash functions. Compared to the previously known long-message second-preimage attacks, our attack offers more flexibility in choosing the second-preimage message at the cost of a small computational overhead. More concretely, our attack allows the adversary to replace only a few blocks in the original target message to obtain the second preimage. As a result,

DOCUMENTATION

Publication:
Journal Article (DOI)

Supplemental Material:
None available

Document History:
06/23/15: Journal Article (Final)

TOPICS

Security and Privacy
secure hashing

MD4 Hash Function Architecture

- State space is equivalent to $\mathbb{Z}_2^{128}$, but is organized as $\mathbb{Z}_2^{32} \times \mathbb{Z}_2^{32} \times \mathbb{Z}_2^{32} \times \mathbb{Z}_2^{32}$ (i.e. state $\underline{q}$ is set of four 32-bit strings or binary vectors).

- Message <u>blocks</u> $\underline{x}_i \in \mathbb{Z}_2^{512}$ (512-bit strings or binary vectors). Again, recall original $\underline{x}$ is of arbitrary length "*," yielding ℓ blocks of length 512.

- The design philosophy of multiple "rounds" shares alot with DES: 16 rounds of nonlinear function applications are used.

Observe: $\underline{\ell} \in (\mathbb{Z}_2^{32})^4$, i.e.

$\mathbb{Z}_2^{128}$ partioned into 4 32-bit strings.

Likewise, a 512-bit block of input partitioned in 16 32-bit strings, ONE FOR EACH ROUND.

So . . .

$\underline{q}_i \in \mathbb{Z}_2^{128}$: state at "time" index i

partitioned into

A_i, B_i, C_i, D_i each $\in \mathbb{Z}_2^{32}$

$\underline{X}_i \in \mathbb{Z}_2^{512}$: i^{th} block of original message, $\underline{X} \in \mathbb{Z}_2^{*}$.

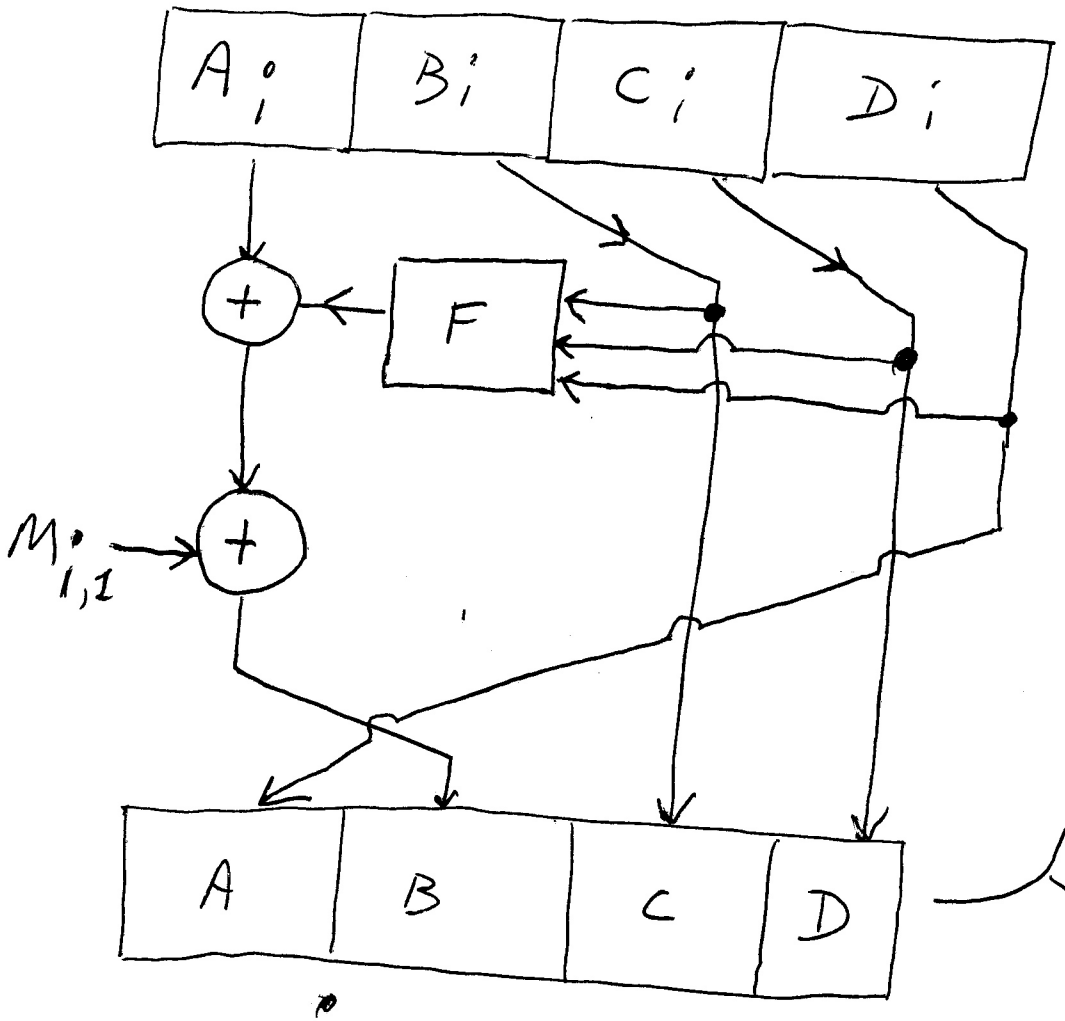
This X_i is further partitioned into 16 subblocks,

$M_{i,1}, M_{i,2} \ldots M_{i,16} \in \mathbb{Z}_2^{32}$

- With these partitions, all computational steps operate on 32-bit strings / binary vectors

- On following page's block diagram, $\oplus$ denotes component-wise addition (mod 2) of vectors in $\mathbb{Z}_2^{32}$

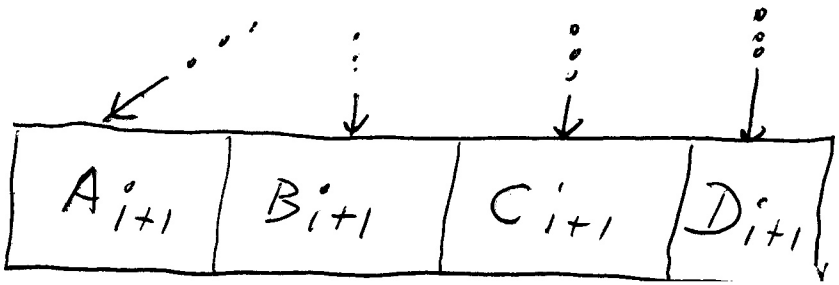
- Exact nature of F not detailed here; simply observe F is nonlinear Boolean function $F: \mathbb{Z}_2^{32} \times \mathbb{Z}_2^{32} \times \mathbb{Z}_2^{32} \to \mathbb{Z}_2^{32}$



← state Q_i at time index i

round 1 intermediate result, "on-the-way" to producing Q_{i+1}

⋮ repeat 15 more rounds w/ $M_{i,2}, M_{i,3} \ldots$



round 16 final result; this becomes $\underline{Q}_{i+1}$

While useful to have a general overview of design philosophy / architecture of MD4, this circa-1990 design is far from state of the art. Like DES, attacks on MD4 are well-known, and well within computational power of circa-2021 computing.

(So, ok for applications of modest security requirements; good security practices uses variants of SHA-2 standard)