LEHRSTUHL FÜR RECHNERARCHITEKTUR UND PARALLELE SYSTEME

Grundlagenpraktikum: Rechnerarchitektur

MD2 (A505)

Projektaufgabe – Aufgabenbereich Kryptographie

1 Organisatorisches

Auf den folgenden Seiten finden Sie die Aufgabenstellung zu Ihrer Projektaufgabe für das Praktikum. Die Rahmenbedingungen für die Bearbeitung werden in der Praktikumsordnung festgesetzt, die Sie über die Praktikumshomepage[1] aufrufen können.

Wie in der Praktikumsordnung beschrieben, sind die Aufgaben relativ offen gestellt. Besprechen Sie diese innerhalb Ihrer Gruppe und konkretisieren Sie die Aufgabenstellung. Die Teile der Aufgabe, in denen Assembler-Code anzufertigen ist, sind für die 64-Bit x86-Architektur (x86-64) unter Verwendung der SSE-Erweiterungen zu schreiben; alle anderen Bestandteile der Hauptimplementierung sind in C nach dem C11-Standard anzufertigen.

Der **Abgabetermin** ist **Sonntag 24. Juli 2022, 23:59 Uhr (CEST)**. Die Abgabe erfolgt per Git in das für Ihre Gruppe eingerichtete Projektrepository. Bitte beachten Sie die in der `README.md` angegebene Liste von abzugebenden Dateien.

Die **Abschlusspräsentationen** finden in der Zeit vom **24.08.2022 – 01.09.2022** statt. Weitere Informationen werden noch bekannt gegeben. Beachten Sie, dass die Folien für die Präsentation am obigen Abgabetermin im PDF-Format abzugeben sind und keine nachträglichen Änderungen akzeptiert werden können.

Bei Fragen/Unklarheiten in Bezug auf den Ablauf und die Aufgabenstellung wenden Sie sich bitte an Ihren Tutor.

Wir wünschen Ihnen viel Erfolg und Freude bei der Bearbeitung Ihrer Aufgabe!

Mit freundlichen Grüßen
Die Praktikumsleitung

[1] `https://gra.caps.in.tum.de`

2 MD2

2.1 Überblick

Kryptographie ist ein essentieller Bestandteil unserer heutigen Kommunikation, beispielsweise beim Surfen im Internet über `HTTPS` oder beim Versenden Ende-zu-Ende verschlüsselter E-Mails mit PGP. In Ihrer Projektaufgabe werden Sie ein kryptographisches Verfahren ganz oder teilweise implementieren.

2.2 Funktionsweise

MD2[2] ist ein von Ronald Rivest 1988 beschriebene **Hashfunktion**. Dabei wird eine Nachricht beliebiger Länge auf einen 128 Bit Wert gehasht. Die genaue Funktionsweise der Hashfunktion ist in RFC 1319 der IETF [1] beschrieben. Die grobe Vorgehensweise zum Berechnen des Hashwerts unterteilt sich in drei Schritte:

- Zunächst wird mittels *Padding* die Nachricht so verlängert, sodass die Länge in Byte ein Vielfaches von 16 ist.
- Danach wird aus der Nachricht eine 16-Byte-*Prüfsumme* gebildet und an die Nachricht angehängt.
- Für das *Hashing* wird ein 48-Byte-Hilfsblock genutzt, in dem für jeden der 16 Byte langen Blöcke rundenweise Berechnungen durchgeführt werden. Nachdem alle Blöcke inklusive Prüfsumme abgearbeitet wurden, ergeben die ersten 16 Bytes des Hilfsblocks den Hashwert der Nachricht.

2.3 Aufgabenstellungen

Ihre Aufgaben lassen sich in die Bereiche Konzeption (theoretisch) und Implementierung (praktisch) aufteilen. Sie können (müssen aber nicht) dies bei der Verteilung der Aufgaben innerhalb Ihrer Arbeitsgruppe ausnutzen. Antworten auf konzeptionelle Fragen sollten an den passenden Stellen in Ihrer Ausarbeitung in angemessenem Umfang erscheinen. Entscheiden Sie nach eigenem Ermessen, ob Sie im Rahmen Ihres Abschlussvortrags auch auf konzeptionelle Fragen eingehen. Die Antworten auf die Implementierungsaufgaben werden durch Ihren Code reflektiert.

2.3.1 Theoretischer Teil

- Die Länge der zu verarbeitenden Daten muss immer ein Vielfaches der Blocklänge sein. Ein Verfahren zum sogenannten *Padding* auf Blocklänge ist *PKCS#7*. Recherchieren Sie jenes in geeigneter Literatur und beschreiben Sie es in Ihrer Ausarbeitung.

[2] **M**essage-**D**igest **2**

- Beschreiben Sie die genaue Funktionsweise der Berechnung von Prüfsumme und Hashwert. Für beide Schritte wird eine Subsitutionsbox verwendet. Wie kommen deren Werte zustande?
- Wofür werden kryptografische Hashfunktionen wie MD2 eingesetzt? Welche Eigenschaften müssen sie erfüllen?
- Die Hashfunktion MD2 gilt inzwischen als obsolet. Beschreiben Sie die Funktionsweise einiger Angriffe auf diese Hashfunktion. Welche Alternativen zu MD2 werden heute eingesetzt?

2.3.2 Praktischer Teil

- Implementieren Sie in Ihrem Rahmenprogramm I/O-Operationen in C, mithilfe derer Sie eine ganze Datei in den Speicher einlesen und als Pointer an eine Unterfunktion übergeben können.
- Implementieren Sie in Ihrem Rahmenprogramm eine Funktion, die einen Block auf ein Vielfaches der Blocklänge auffüllt.
- Implementieren Sie in der Datei mit Ihrem C-Code die Funktion:

  ```
  void md2_checksum(size_t len, uint8_t* buf)
  ```

  Die Funktion erhält einen Pointer auf die Nachricht (`buf`) und dessen Länge (`len`) und hängt die berechnete Prüfsumme an die Nachricht an. Der Speicherbereich `buf` sollte entsprechend groß gewählt werden.
- Implementieren Sie in der Datei mit Ihrem C-Code die Funktion:

  ```
  void md2_hash(size_t len, const uint8_t buf[len], uint8_t out[16])
  ```

  Die Funktion führt auf einer Nachricht mit gegebener Länge die eigentliche Hashfunktion aus und schreibt das Ergebnis in das übergebene Array `out`.

2.3.3 Rahmenprogramm

Ihr Rahmenprogramm muss bei einem Aufruf die folgenden Optionen entgegennehmen und verarbeiten können. Wenn möglich soll das Programm sinnvolle Standardwerte definieren, sodass nicht immer alle Optionen gesetzt werden müssen.

- `-V<int>` — Die Implementierung, die verwendet werden soll. Hierbei soll mit `-V0` Ihre Hauptimplementierung verwendet werden. Wenn diese Option nicht gesetzt wird, soll ebenfalls die Hauptimplementierung ausgeführt werden.
- `-B<int>` — Falls gesetzt, wird die Laufzeit der angegebenen Implementierung gemessen und ausgegeben. Das optionale Argument dieser Option gibt die Anzahl an Wiederholungen des Funktionsaufrufs an.

- `<file>` — Positional Argument: Eingabedatei
- `-h` — Eine Beschreibung aller Optionen des Programms und Verwendungsbeispiele werden ausgegeben und das Programm danach beendet.
- `--help` — Eine Beschreibung aller Optionen des Programms und Verwendungsbeispiele werden ausgegeben und das Programm danach beendet.

Sie dürfen weitere Optionen implementieren, beispielsweise um vordefinierte Testfälle zu verwenden. Ihr Programm muss jedoch nur unter Verwendung der oben genannten Optionen verwendbar sein. Beachten Sie ebenfalls, dass Ihr Rahmenprogramm etwaige Randfälle korrekt abfangen muss und im Falle eines Fehlers mit einer aussagekräftigen Fehlermeldung auf `stderr` und einer kurzen Erläuterung zur Benutzung terminieren sollte.

2.4 Allgemeine Bewertungshinweise

Beachten Sie grundsätzlich alle in der Praktikumsordnung angegebenen Hinweise. Die folgende Liste konkretisiert einige der Bewertungspunkte:

- Stellen Sie unbedingt sicher, dass *sowohl* Ihre Implementierung *als auch* Ihre Ausarbeitung auf der Referenzplattform des Praktikums (`lxhalle`) kompilieren und vollständig korrekt bzw. funktionsfähig sind.
- Die Implementierung soll mit GCC/GNU as kompilieren. Achten Sie darauf, dass Ihr Programm keine x87-FPU- oder MMX-Instruktionen und SSE-Erweiterungen nur bis SSE4.2 verwendet. Andere ISA-Erweiterungen (z.B. AVX, BMI1) dürfen Sie nur benutzen, sofern Ihre Implementierung auch auf Prozessoren ohne derartige Erweiterungen lauffähig ist.
- Sie dürfen die angegebenen Funktionssignaturen (nur dann) ändern, wenn Sie dies (in Ihrer Ausarbeitung) begründen.
- Verwenden Sie die angegebenen Funktionsnamen für Ihre Hauptimplementierung. Falls Sie mehrere Implementierungen schreiben, legen wir Ihnen nahe, für die Benennung der alternativen Implementierungen mit dem Suffix „`_V1`“, „`_V2`“ etc. zu arbeiten.
- Denken Sie daran, das Laufzeitverhalten Ihres Codes zu testen (Sichere Programmierung, Performanz) und behandeln Sie *alle möglichen Eingaben*, auch Randfälle. Ziehen Sie ggf. alternative Implementierungen als Vergleich heran.
- Eingabedateien, welche Sie generieren, um Ihre Implementierungen zu testen, sollten mit abgegeben werden; größere Eingaben sollten stattdessen stark komprimiert oder (bevorzugt) über ein abgegebenes Skript generierbar sein.
- Stellen Sie Performanz-Ergebnisse nach Möglichkeit grafisch dar.

- Vermeiden Sie unscharfe Grafiken und Screenshots von Code.
- Geben Sie die Folien für Ihre Abschlusspräsentation im PDF-Format ab. Achten Sie auf hinreichenden Kontrast (schwarzer Text auf weißem Grund!) und eine angemessene Schriftgröße. Verwenden Sie 16:9 als Folien-Format.

Literatur

[1] Burt Kaliski. *RFC1319: The MD2 Message-Digest Algorithm*. `https://tools.ietf.org/html/rfc1319`. 1992.