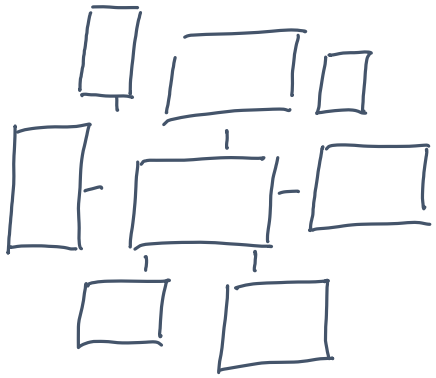
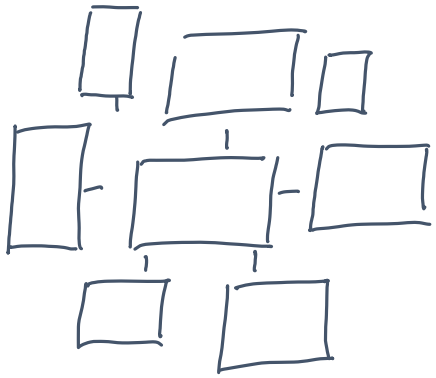


Embedded Systems

Bachelor of Science in Electrical Engineering

Prof. Dr. C. Jakob

University of Applied Sciences Darmstadt
h_da
Faculty of Electrical Engineering and Information Technology
fbeit



Embedded Systems

Bachelor of Science in Electrical Engineering

Prof. Dr. C. Jakob

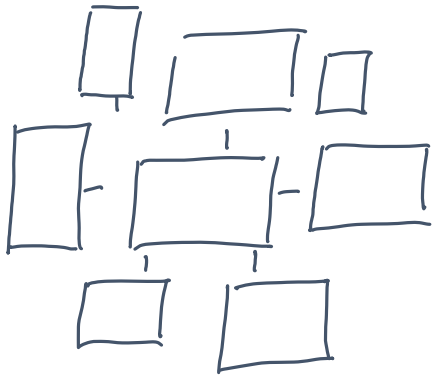
University of Applied Sciences Darmstadt
h_da
Faculty of Electrical Engineering and Information Technology
fbeit

ESY Semester Project WS23/24

Today's Agenda

Lecture Content

- ESY Semester Project
- Secure Hash Algorithm 1 (SHA-1)
- Project specific Readings
- Recommended Readings and Online Resources



Embedded Systems

Bachelor of Science in Electrical Engineering

Prof. Dr. C. Jakob

University of Applied Sciences Darmstadt
h_da
Faculty of Electrical Engineering and Information Technology
fbeit

Implementing the SHA-1 on the ATmega328p

Praktischer Teil

- Umsetzung des SHA-1 Verfahrens auf dem Microchip ATmega328p (genaue Spezifikation siehe nachfolgend)

Implementing the SHA-1 on the ATmega328p

Praktischer Teil

- Umsetzung des SHA-1 Verfahrens auf dem Microchip ATmega328p (genaue Spezifikation siehe nachfolgend)

Schriftliche Ausarbeitung

- (Kurze) Diskussion der zugrundeliegenden Theorie (siehe theoretische Fragen nachfolgend)

Implementing the SHA-1 on the ATmega328p

Praktischer Teil

- Umsetzung des SHA-1 Verfahrens auf dem Microchip ATmega328p (genaue Spezifikation siehe nachfolgend)

Schriftliche Ausarbeitung

- (Kurze) Diskussion der zugrundeliegenden Theorie (siehe theoretische Fragen nachfolgend)
- Analyse/Diskussion der Firmware-Implementierung:
  - Umsetzung des SHA-1 Verfahrens
  - Umsetzung des USART-Kommandointerpreters – Ablaufsteuerung

Implementing the SHA-1 on the ATmega328p

Verbindliche Vorgaben

- Die Programmierung muss mit der ATMEL/Microchip Studio IDE sowie in ANSI C erfolgen.
- Die Benutzung von bereits existierenden Bibliotheken/Lösungen ist nicht gestattet.

Implementing the SHA-1 on the ATmega328p

SHA-1 – Minimale Anforderungen an Ihre Umsetzung

- Die Datenvorverarbeitung (Initial Preprocessing) kann PC-seitig erfolgen.
- Die SHA-1 Verarbeitung bezieht sich lediglich auf einen 512-bit breiten Datenblock.
- Dieser wird über die USART Schnittstelle an den ATmega328p übertragen und temporär zwischengespeichert.
- Durch das nachfolgende Senden des USART-Steuerkommandos **#!** erfolgt erst die eigentliche SHA-1 Verarbeitung des übermittelten Datenblocks. Das Ergebnis wird µC-seitig wieder temporär zwischengespeichert.
- Durch Übertragen des USART-Steuerkommandos **#$** wird das Ergebnis ausgelesen und zurück an die PC-Konsole übertragen.
- Nutzen Sie als Teststring nachfolgende Zeichenkette "**ESY – XYZ**", wobei es XYZ durch Ihre Matrikelnummer zu ersetzen gilt.

Implementing the SHA-1 on the ATmega328p

Bewertung

- Zur Bewertung der Projektarbeit werden die Komplexität des realisierten Projektes (Projektstand bei Abgabe) sowie die Dokumentation des Projektes herangezogen.
- Im Detail orientiert sich die Bewertung Ihrer Arbeit an folgenden Punkten:
  - Qualität und Tiefgang der geleisteten Tätigkeit,
  - der Kreativität sowie dem Einbringen eigener Ideen,
  - Selbständigkeit, Motivation, Systematik,
  - Fähigkeit zur Realisierung bzw. zur praktischen Umsetzung,
  - Aufbau/Struktur/Stil der schriftlichen Projektarbeit,
  - Eigene Ergebnisanalyse, Schlussfolgerungen und Empfehlungen.

Implementing the SHA-1 on the ATmega328p

Formale Gestaltung

- Siehe Template (moodle)
- Gliederung und Aufbau:
  - **Titelblatt**
  - **Eigenständigkeitserklärung**:

    Hiermit bestätige ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken (dazu zählen auch Internetquellen) entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht.

    Der Kernteil der schriftlichen Ausarbeitung (5 Din-A4 Seiten) sollte doppelspaltig verfasst werden und folgenden Aufbau besitzen:

  - **Kurzfassung** (Abstract)
  - **Allgemeine Einführung** (Einleitung)
  - ...

Implementing the SHA-1 on the ATmega328p

Formale Gestaltung

- Hauptteil
  - Dieser kann weiter untergliedert werden …
  - …
- Zusammenfassung
- Schlussfolgerung
- Quellenverzeichnis

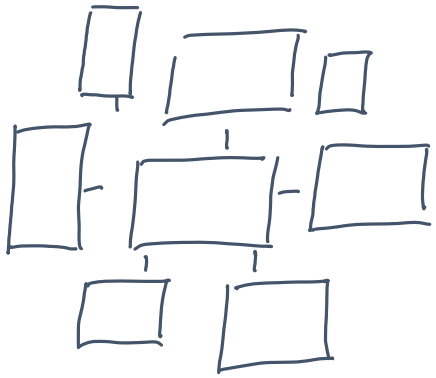
Wichtiger Hinweis:

Aus fremden Arbeiten übernommene Textstellen, Tabellen, Grafiken etc. müssen als wörtliche oder sinngemäße Zitate kenntlich gemacht werden (Angabe im Quellenverzeichnis). Andernfalls liegt ein Plagiat vor, was zur Nichtanerkennung der Projektarbeit und zu einem Nichtbestehen der Veranstaltung führt.

Implementing the SHA-1 on the ATmega328p

Abgabe

- Zur Überprüfung der Arbeit auf mögliche Plagiate muss die Abgabe in folgender Form erfolgen:
  - Schriftliche Ausarbeitung: Adobe pdf oder MS Word Format.
  - Software Gezippter Ordner des Atmel/Microchip Studio Projektes.



The Secure Hash Algorithm 1 (SHA-1)

Bachelor of Science in Electrical Engineering

Prof. Dr. C. Jakob

University of Applied Sciences Darmstadt
h_da
Faculty of Electrical Engineering and Information Technology
fbeit

The Secure Hash Algorithm 1 (SHA-1)

- SHA was designed by the National Institute of Standards and Technology (NIST) and is the US federal standard for hash functions, specified in FIPS-180 (1993).
- SHA-1, revised version of SHA, specified in FIPS-180-1 (1995) use with Secure Hash Algorithm). It produces 160-bit hash values.
- Applications: Hash functions are widely used for rapid data lookup, password storage, data integrity and authentication checks.
- Since 2005 SHA-1 has not been considered secure against well-funded opponents. Since 2010 many organizations have recommended its replacement by SHA-2 or SHA-3.
- Thursday, 23 February 2017, researchers at the Dutch research institute CWI and Google jointly announce that they have broken the SHA-1 internet security standard in practice, publishing two dissimilar PDF files which produced the same SHA-1 hash. We have broken SHA-1 in practice: https://shattered.io/

The Secure Hash Algorithm 1 (SHA-1)

- Just a simple function …

Message (arbitrary length)

FSOC 2021 is fun!

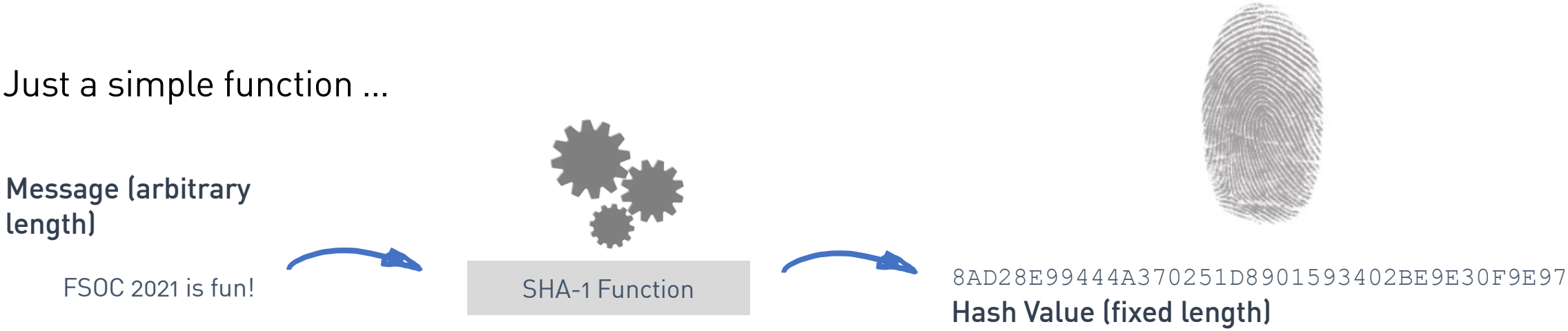
SHA-1 Function

`8AD28E99444A370251D8901593402BE9E30F9E97`

Hash Value (fixed length)

The Secure Hash Algorithm 1 (SHA-1)

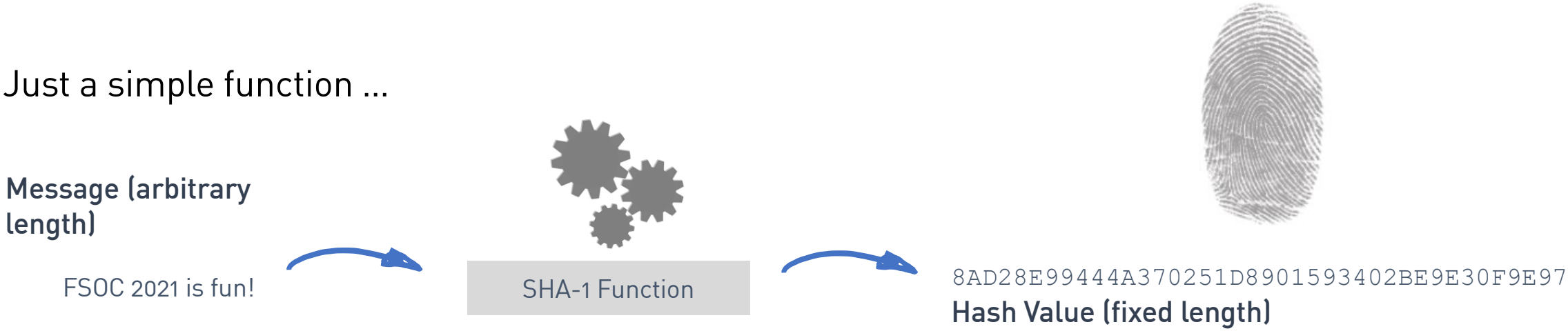
- Just a simple function …



- The SHA-1 algorithm on the left is a one-way function that transforms an arbitrary-length message into a 160-bit hash value (fixed length digest).

The Secure Hash Algorithm 1 (SHA-1)

- Just a simple function …



- The SHA-1 algorithm on the left is a one-way function that transforms an arbitrary-length message into a 160-bit hash value (fixed length digest).
- Cryptographic hash values are sometimes referred to as digital fingerprints.

The Secure Hash Algorithm 1 (SHA-1)

```
student@fsoc-edasys:~$ sudo apt-get install openssl
```

[Linux console]: Install the openSSL toolset under Ubuntu Linux …

The Secure Hash Algorithm 1 (SHA-1)

```
student@fsoc-edasys:~$ sudo apt-get install openssl
```

[Linux console]: Install the openSSL toolset under Ubuntu Linux …

```
student@fsoc-edasys:~$ echo -n "FSOC 2021 is fun!" | openssl sha1
(stdin)= 8ad28e99444a370251d8901593402be9e30f9e97
```

[Linux console]: SHA-1 Hash computation – String Entry

The Secure Hash Algorithm 1 (SHA-1)

```
student@fsoc-edasys:~$ sudo apt-get install openssl
```

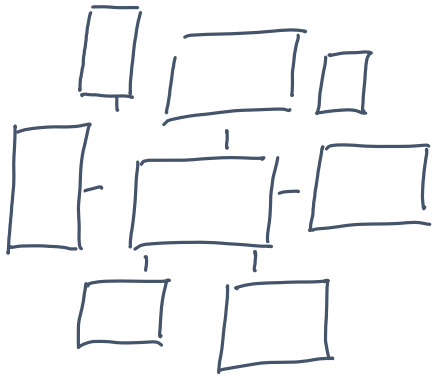
[Linux console]: Install the openSSL toolset under Ubuntu Linux …

```
student@fsoc-edasys:~$ echo -n "FSOC 2021 is fun!" | openssl sha1
(stdin)= 8ad28e99444a370251d8901593402be9e30f9e97
```

[Linux console]: SHA-1 Hash computation – String Entry

```
student@fsoc-edasys:~$ echo -n "FSOC 2021 is fun!" > sha1_input.txt
student@fsoc-edasys:~$ cat sha1_input.txt && echo
FSOC 2021 is fun!
student@fsoc-edasys:~$ sha1sum sha1_input.txt
8ad28e99444a370251d8901593402be9e30f9e97  sha1_input.txt
student@fsoc-edasys:~$
```

[Linux console]: SHA-1 Hash computation – File Entry



SHA-1 - Initial Preprocessing

Bachelor of Science in Electrical Engineering

Prof. Dr. C. Jakob

University of Applied Sciences Darmstadt
h_da
Faculty of Electrical Engineering and Information Technology
fbeit

Initial Preprocessing

Initial Preprocessing

Preparation 1: **Append Padding Bits**: Message is "padded" with a single binary 1 and as many 0's as necessary to bring the message length to 64 bits fewer than an even multiple of 512.

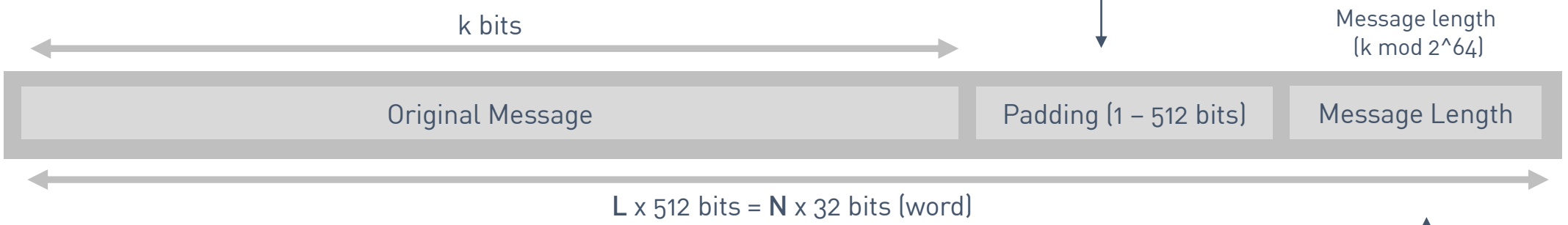
Initial Preprocessing

Preparation 1: **Append Padding Bits**: Message is "padded" with a single binary 1 and as many 0's as necessary to bring the message length to 64 bits fewer than an even multiple of 512.

Preparation 2: **Append the Message Length**: 64 bits are appended to the end of the padded message. These bits hold the binary format of 64 bits indicating the length of the original message.

Initial Preprocessing

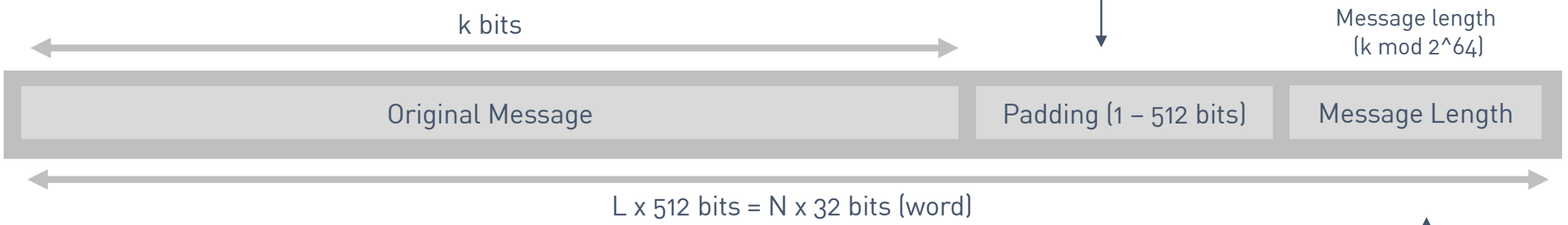
Preparation 1: **Append Padding Bits**: Message is "padded" with a single binary 1 and as many 0's as necessary to bring the message length to 64 bits fewer than an even multiple of 512.



Preparation 2: **Append the Message Length**: 64 bits are appended to the end of the padded message. These bits hold the binary format of 64 bits indicating the length of the original message.

Initial Preprocessing

Preparation 1: **Append Padding Bits**: Message is "padded" with a single binary 1 and as many 0's as necessary to bring the message length to 64 bits fewer than an even multiple of 512.



Preparation 2: **Append the Message Length**: 64 bits are appended to the end of the padded message. These bits hold the binary format of 64 bits indicating the length of the original message.

- Before the actual computation starts, the algorithm has to preprocess the message.

Initial Preprocessing - Example

Taken from: Paar, C. & Pelzl, J.: Understanding Cryptography. A Textbook for Students and Practitioners, pp. 308-309, Springer-Verlag, 2009.

- Given is the message “abc” consisting of three 8-bit ASCII characters with a total length of l = 24 bits:

a b c

Initial Preprocessing - Example

Taken from: Paar, C. & Pelzl, J.: Understanding Cryptography. A Textbook for Students and Practitioners, pp. 308-309, Springer-Verlag, 2009.

- Given is the message “abc” consisting of three 8-bit ASCII characters with a total length of l = 24 bits:

01100001	01100010	01100011
a	b	c

Initial Preprocessing - Example

Taken from: Paar, C. & Pelzl, J.: Understanding Cryptography. A Textbook for Students and Practitioners, pp. 308-309, Springer-Verlag, 2009.

- Given is the message “abc” consisting of three 8-bit ASCII characters with a total length of l = 24 bits:

01100001	01100010	01100011
a	b	c

- We append a “1” followed by k = 423 zero bits, where k is determined by

$$k \equiv \mathbf{448} - (l+1) = 448 - 25 = 423 \bmod 512$$

512bit - 64bit = 448-bit

Initial Preprocessing - Example

Taken from: Paar, C. & Pelzl, J.: Understanding Cryptography. A Textbook for Students and Practitioners, pp. 308-309, Springer-Verlag, 2009.

- Given is the message “abc” consisting of three 8-bit ASCII characters with a total length of l = 24 bits:

01100001	01100010	01100011
a	b	c

- We append a “1” followed by k = 423 zero bits, where k is determined by

$$k \equiv 448 - (l+1) = 448 - 25 = 423 \bmod 512$$ **512bit - 64bit = 448-bit**

- Finally, we append the **64-bit value** which contains the binary representation of the **length** l = 24(dec.) = 11000(bin). The padded message is then given by:

Initial Preprocessing - Example

Taken from: Paar, C. & Pelzl, J.: Understanding Cryptography. A Textbook for Students and Practitioners, pp. 308-309, Springer-Verlag, 2009.

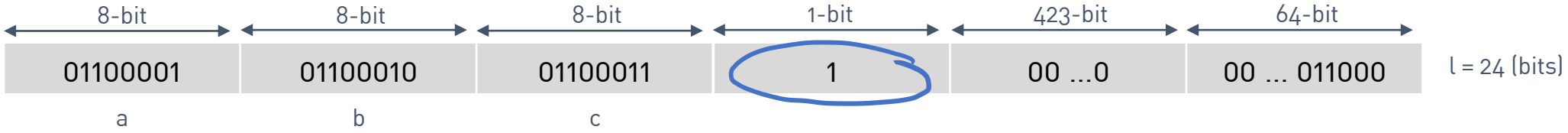
- Given is the message "abc" consisting of three 8-bit ASCII characters with a total length of l = 24 bits:

01100001	01100010	01100011
a	b	c

- We append a "1" followed by k = 423 zero bits, where k is determined by

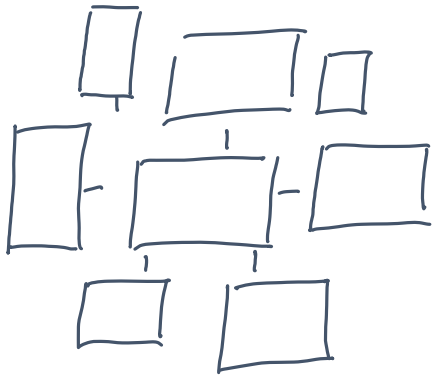
$$k \equiv 448 - (l+1) = 448 - 25 = 423 \bmod 512$$ **512bit - 64bit = 448-bit**

- Finally, we append the **64-bit value** which contains the binary representation of the **length** l = 24(dec.) = 11000(bin). The padded message is then given by:



Initial Preprocessing - Homework

- Prepare the following message for SHA-1 processing: 0x6162636465



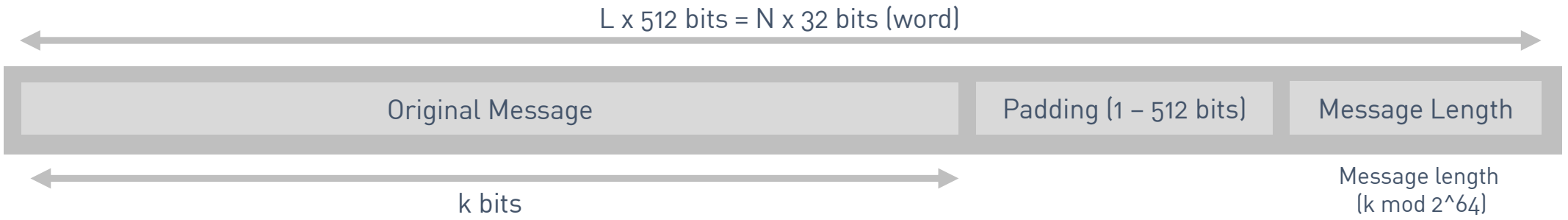
SHA-1 - Basic SHA-1 Structure

Bachelor of Science in Electrical Engineering

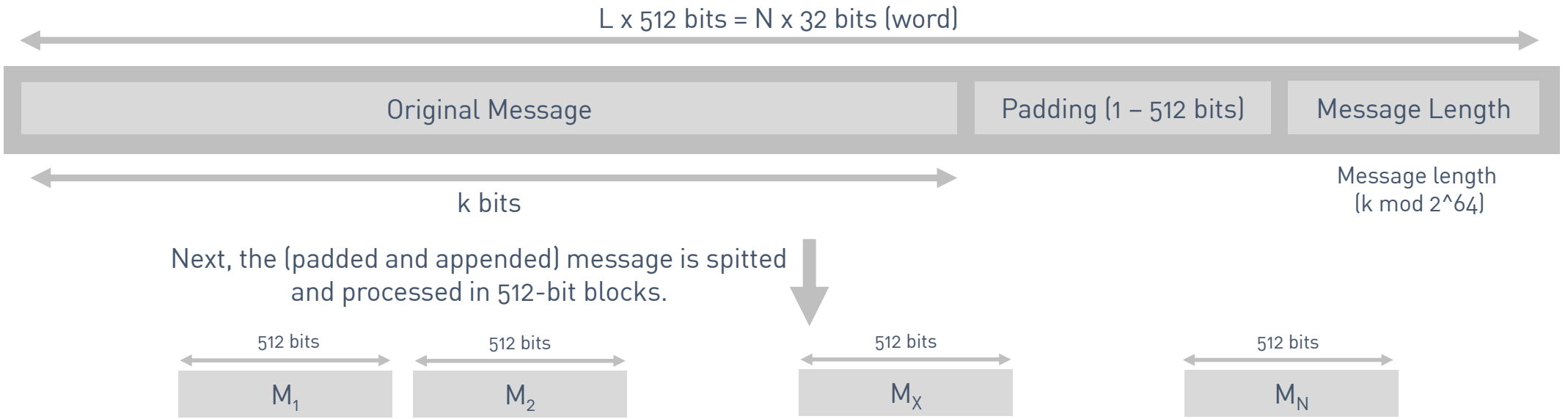
Prof. Dr. C. Jakob

University of Applied Sciences Darmstadt
h_da
Faculty of Electrical Engineering and Information Technology
fbeit

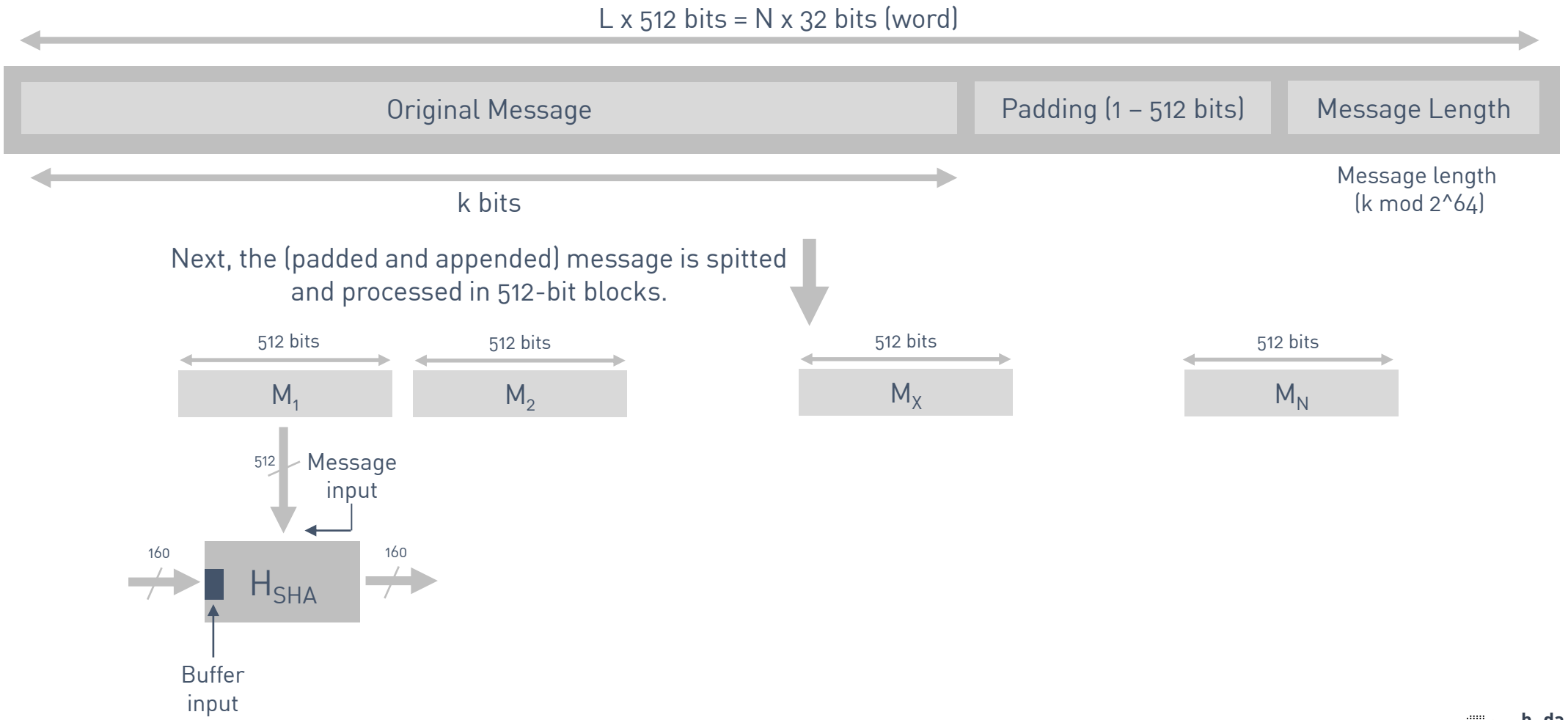
Basic SHA-1 Structure



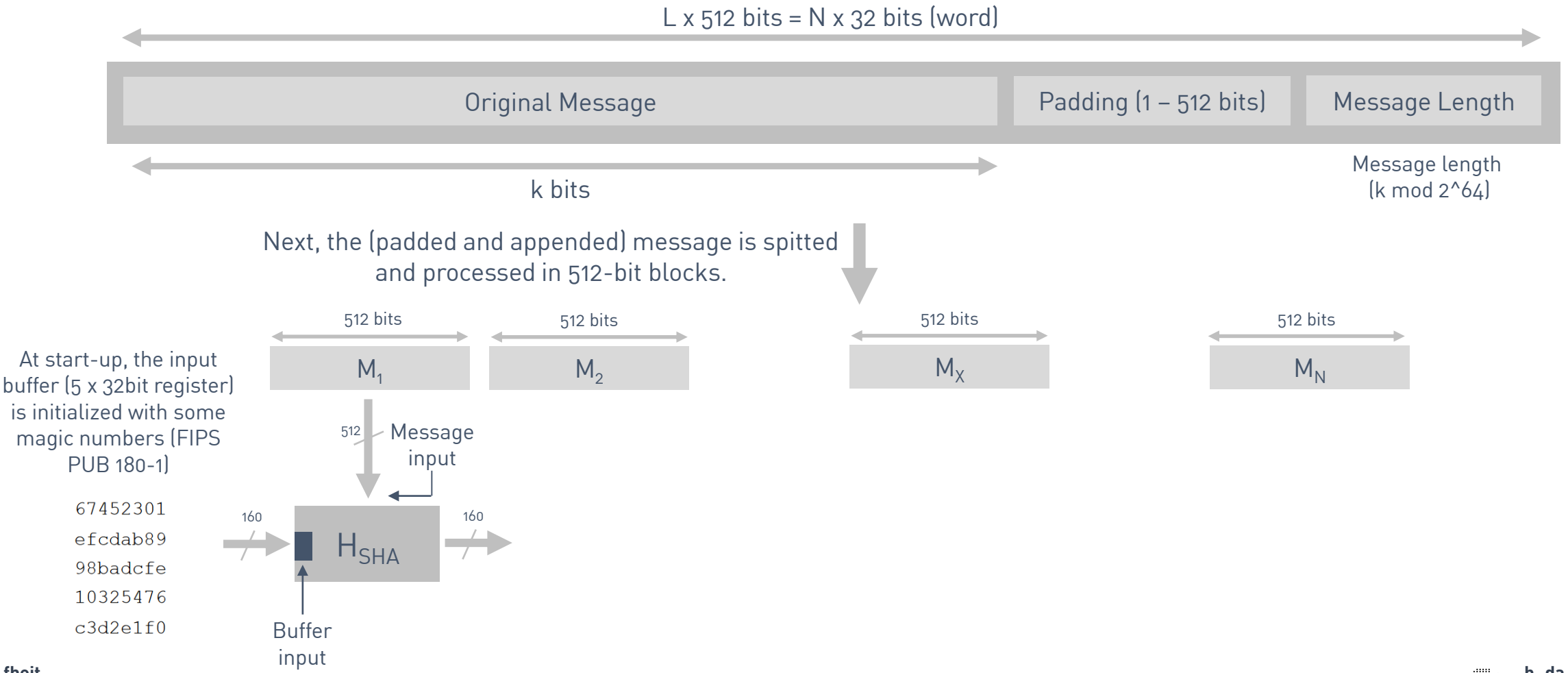
Basic SHA-1 Structure



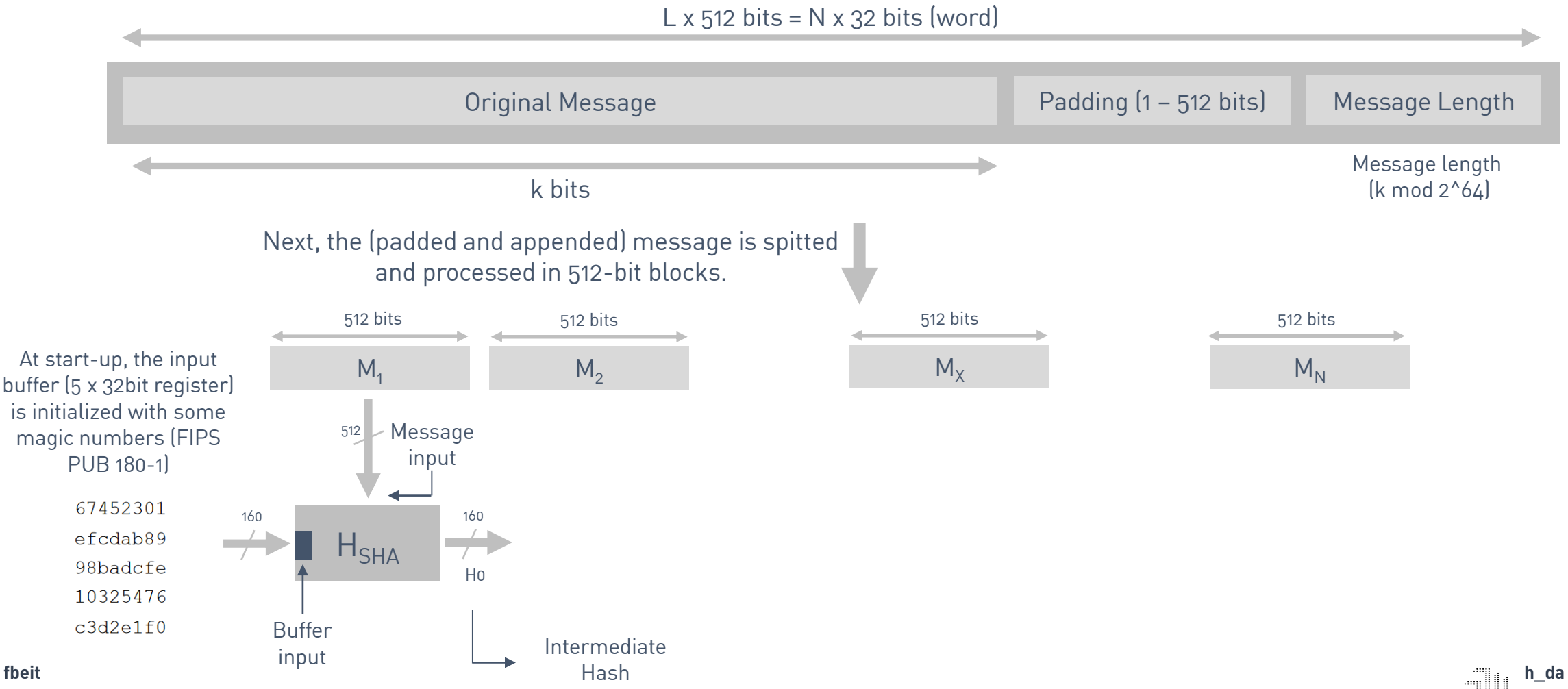
Basic SHA-1 Structure



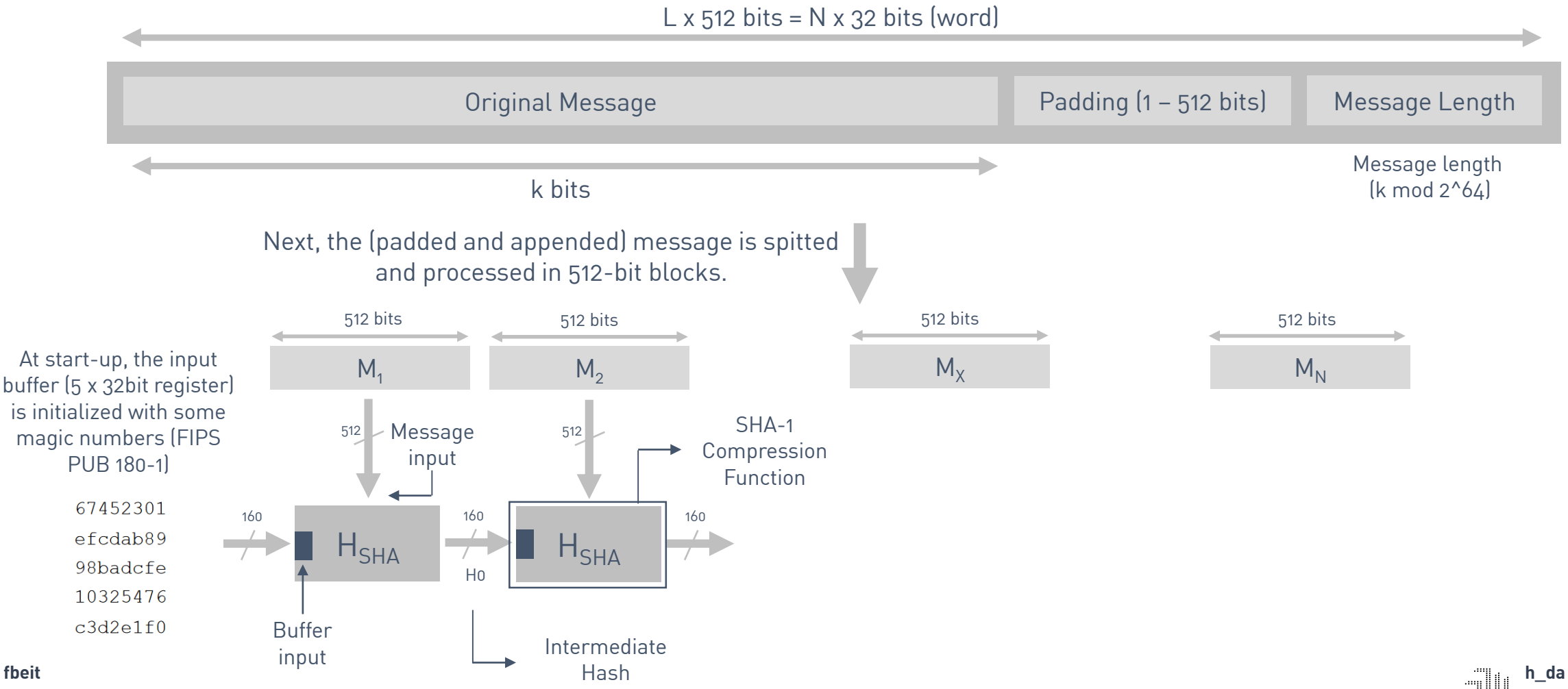
Basic SHA-1 Structure



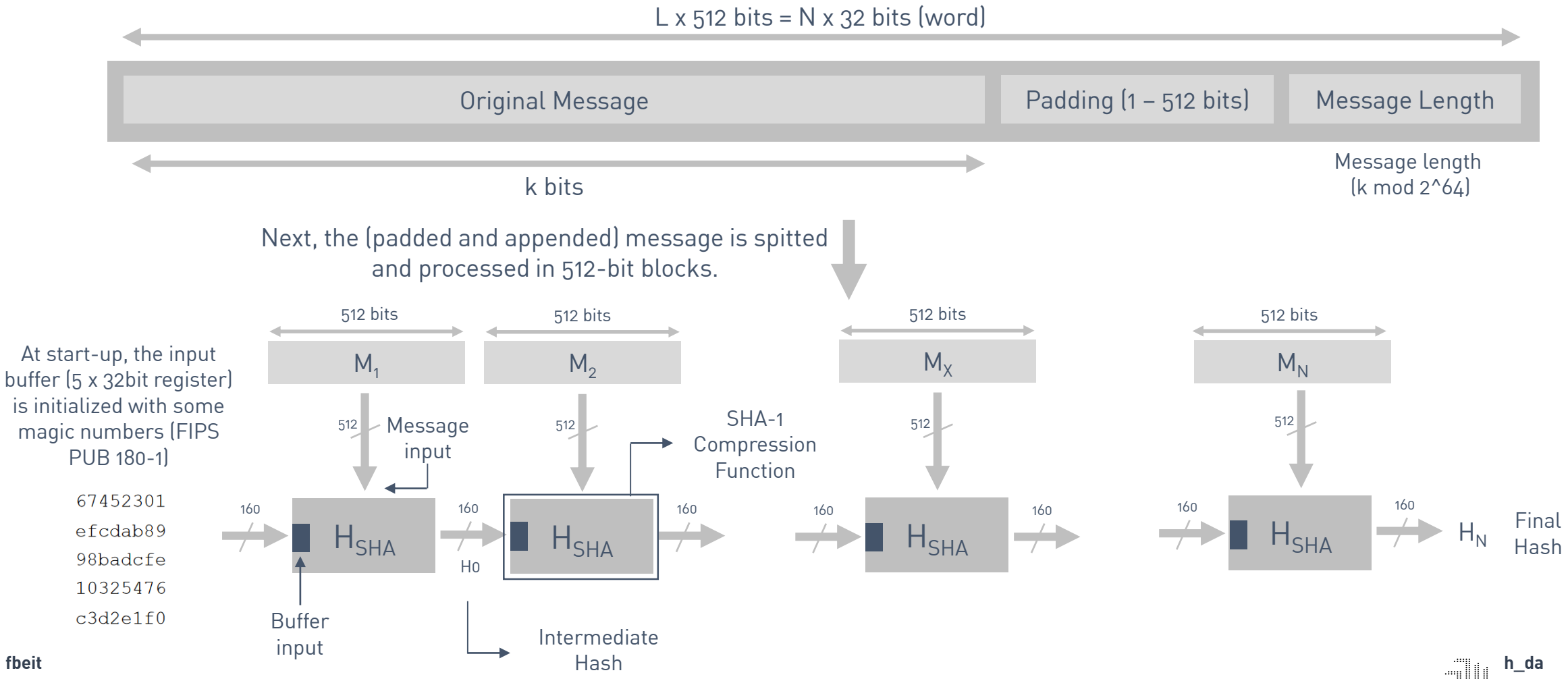
Basic SHA-1 Structure



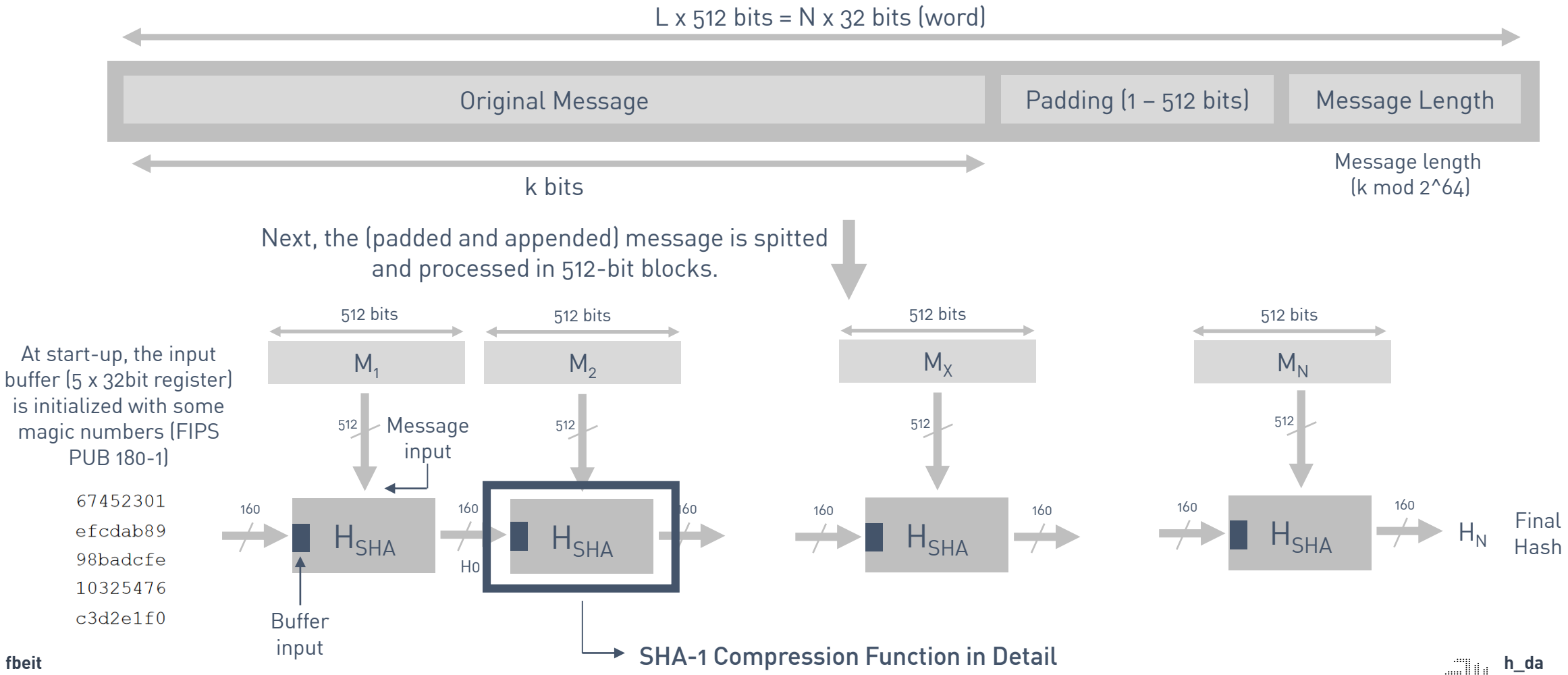
Basic SHA-1 Structure

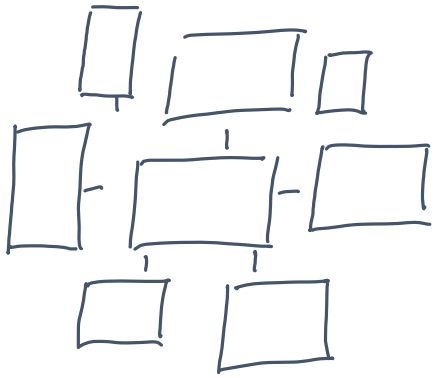


Basic SHA-1 Structure



Basic SHA-1 Structure





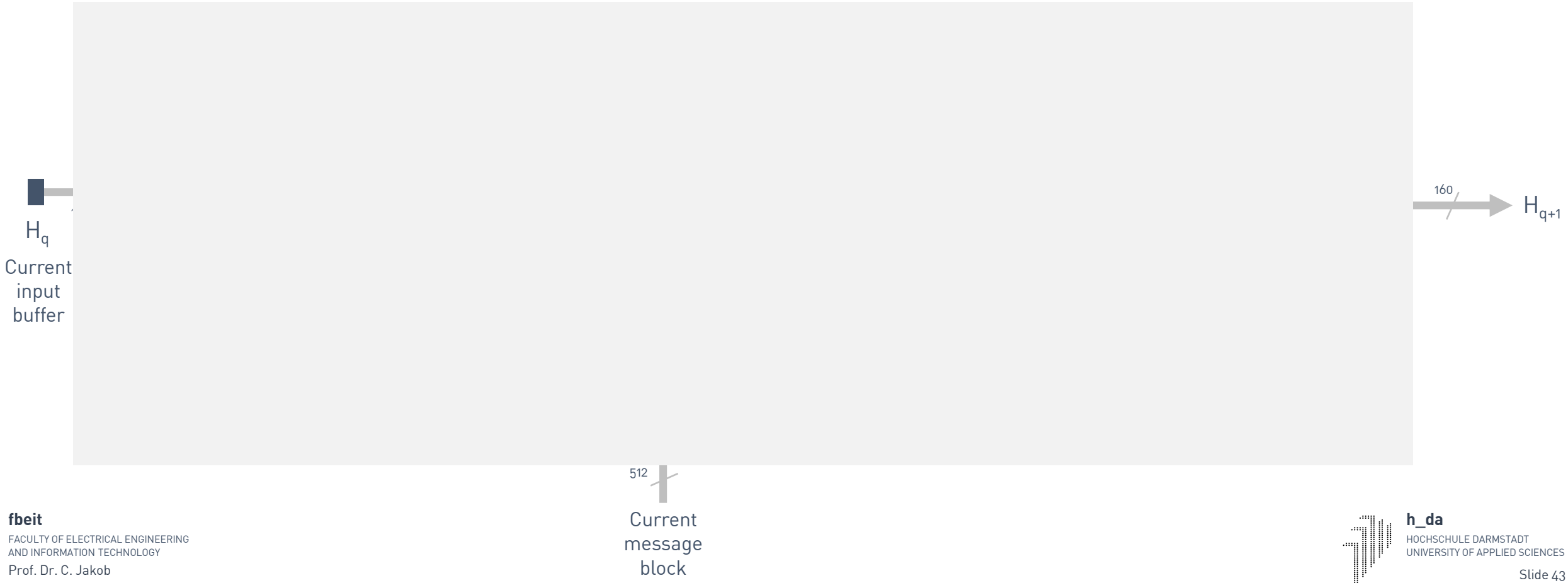
SHA-1 – Compression Function

Bachelor of Science in Electrical Engineering

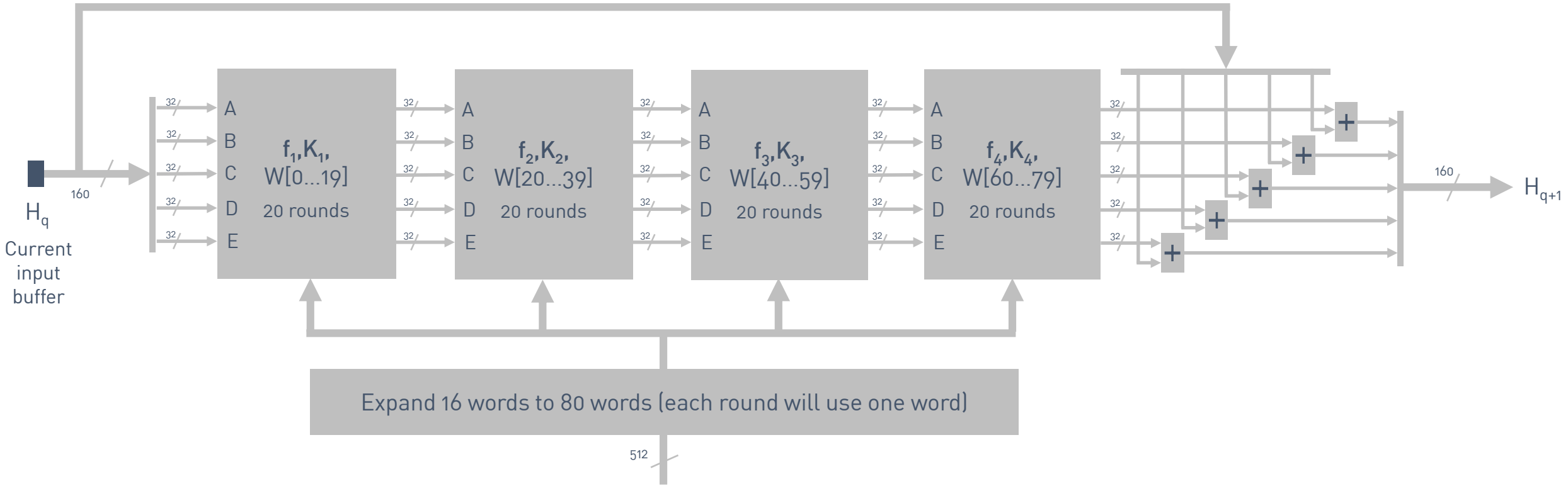
Prof. Dr. C. Jakob

University of Applied Sciences Darmstadt
h_da
Faculty of Electrical Engineering and Information Technology
fbeit

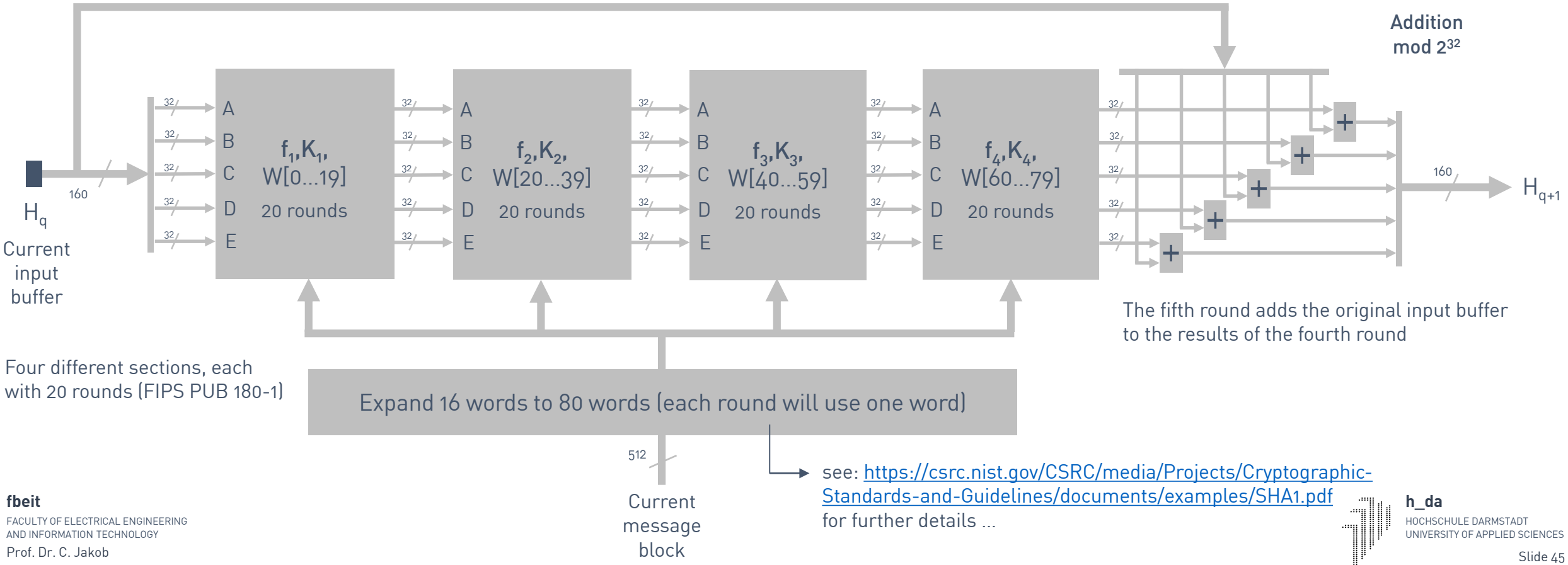
SHA-1 Compression Function



SHA-1 Compression Function



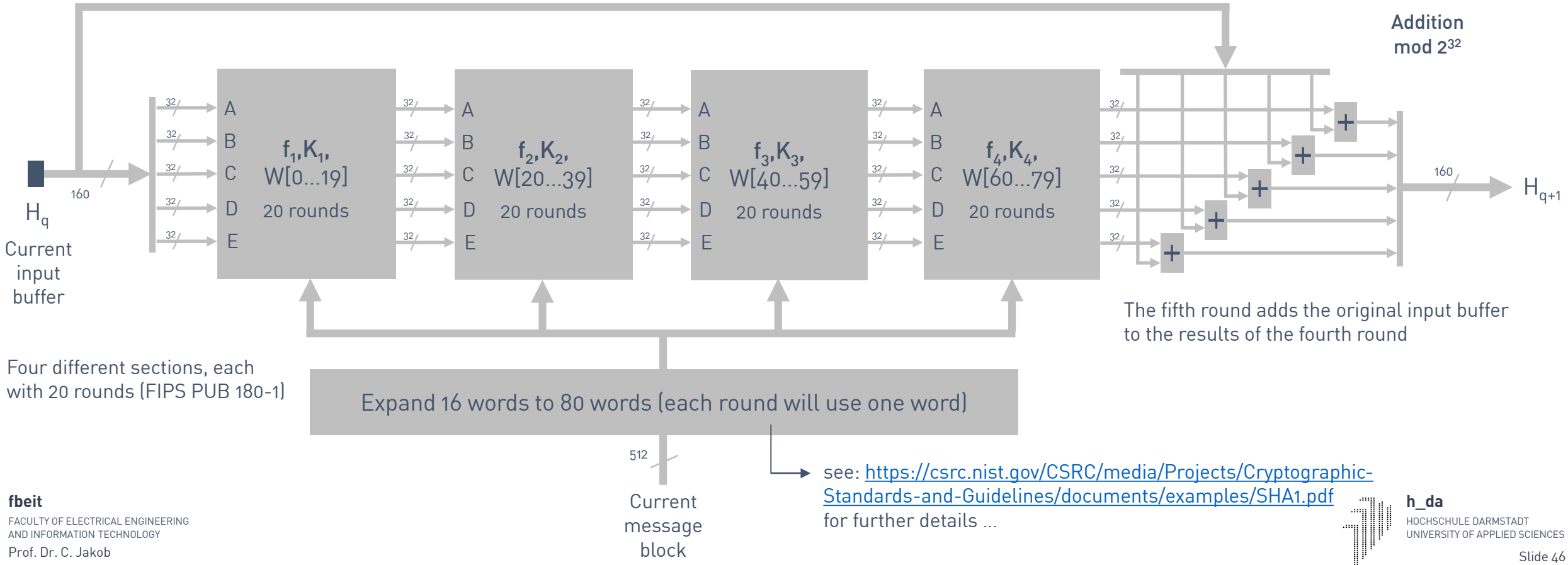
SHA-1 Compression Function



SHA-1 Compression Function

$$f_t(x,y,z) = \begin{cases} Ch(x,y,z)=(x \wedge y) \oplus (\neg x \wedge z) & 0 \le t \le 19 \\ Parity(x,y,z)=x \oplus y \oplus z & 20 \le t \le 39 \\ Maj(x,y,z)=(x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) & 40 \le t \le 59 \\ Parity(x,y,z)=x \oplus y \oplus z & 60 \le t \le 79 \end{cases}$$

SHA-1 Functions



see: https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Standards-and-Guidelines/documents/examples/SHA1.pdf for further details ...

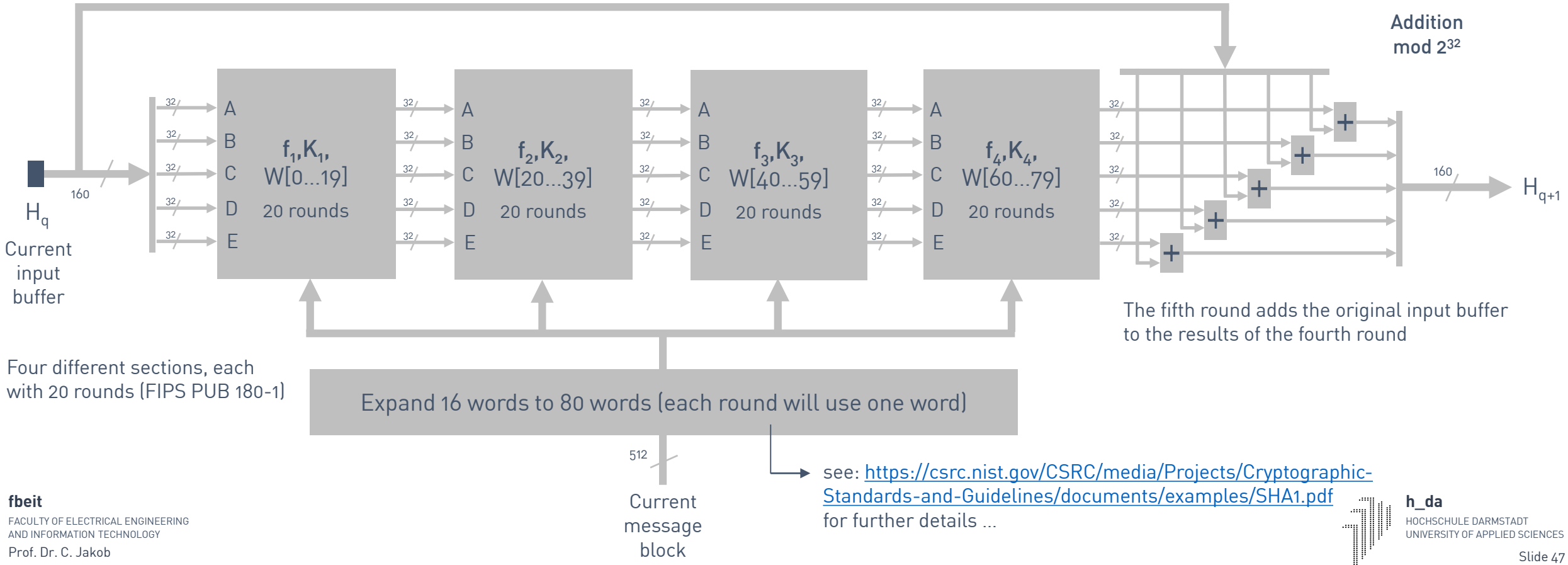
SHA-1 Compression Function

$$f_t(x,y,z) = \begin{cases} Ch(x,y,z)=(x \wedge y) \oplus (\neg x \wedge z) & 0 \le t \le 19 \\ Parity(x,y,z)=x \oplus y \oplus z & 20 \le t \le 39 \\ Maj(x,y,z)=(x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) & 40 \le t \le 59 \\ Parity(x,y,z)=x \oplus y \oplus z & 60 \le t \le 79 \end{cases}$$

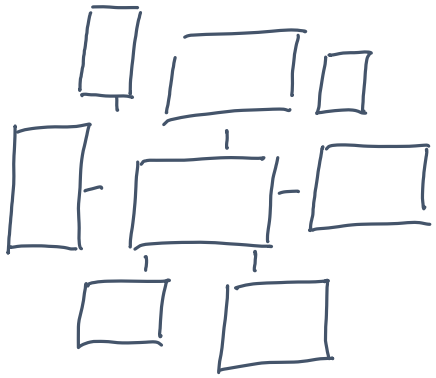
SHA-1 Functions

$$K_t = \begin{cases} \texttt{5a827999} & 0 \le t \le 19 \\ \texttt{6ed9eba1} & 20 \le t \le 39 \\ \texttt{8f1bbcdc} & 40 \le t \le 59 \\ \texttt{ca62c1d6} & 60 \le t \le 79 \end{cases}$$

SHA-1 Constants



see: https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Standards-and-Guidelines/documents/examples/SHA1.pdf for further details ...



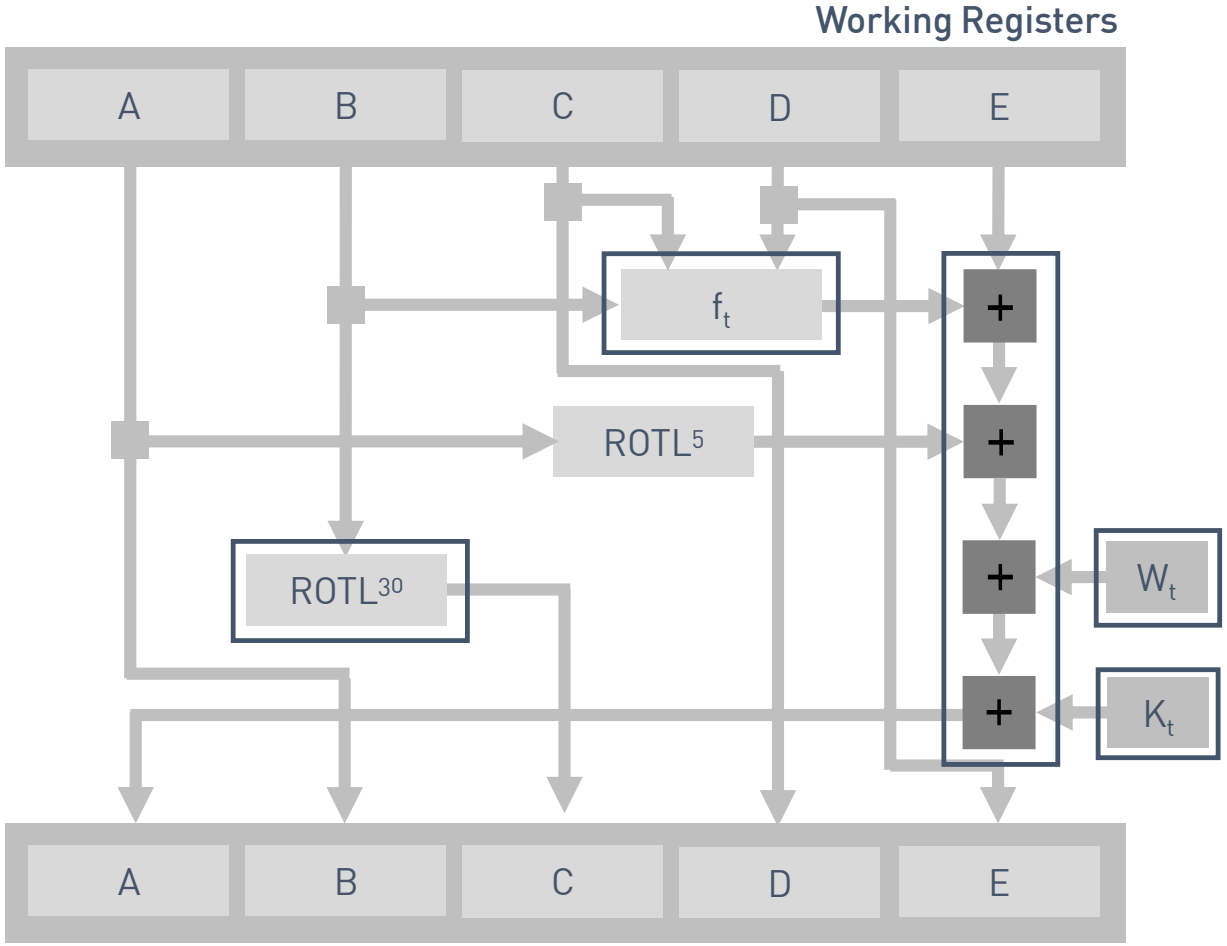
SHA-1 – Core Function

Bachelor of Science in Electrical Engineering

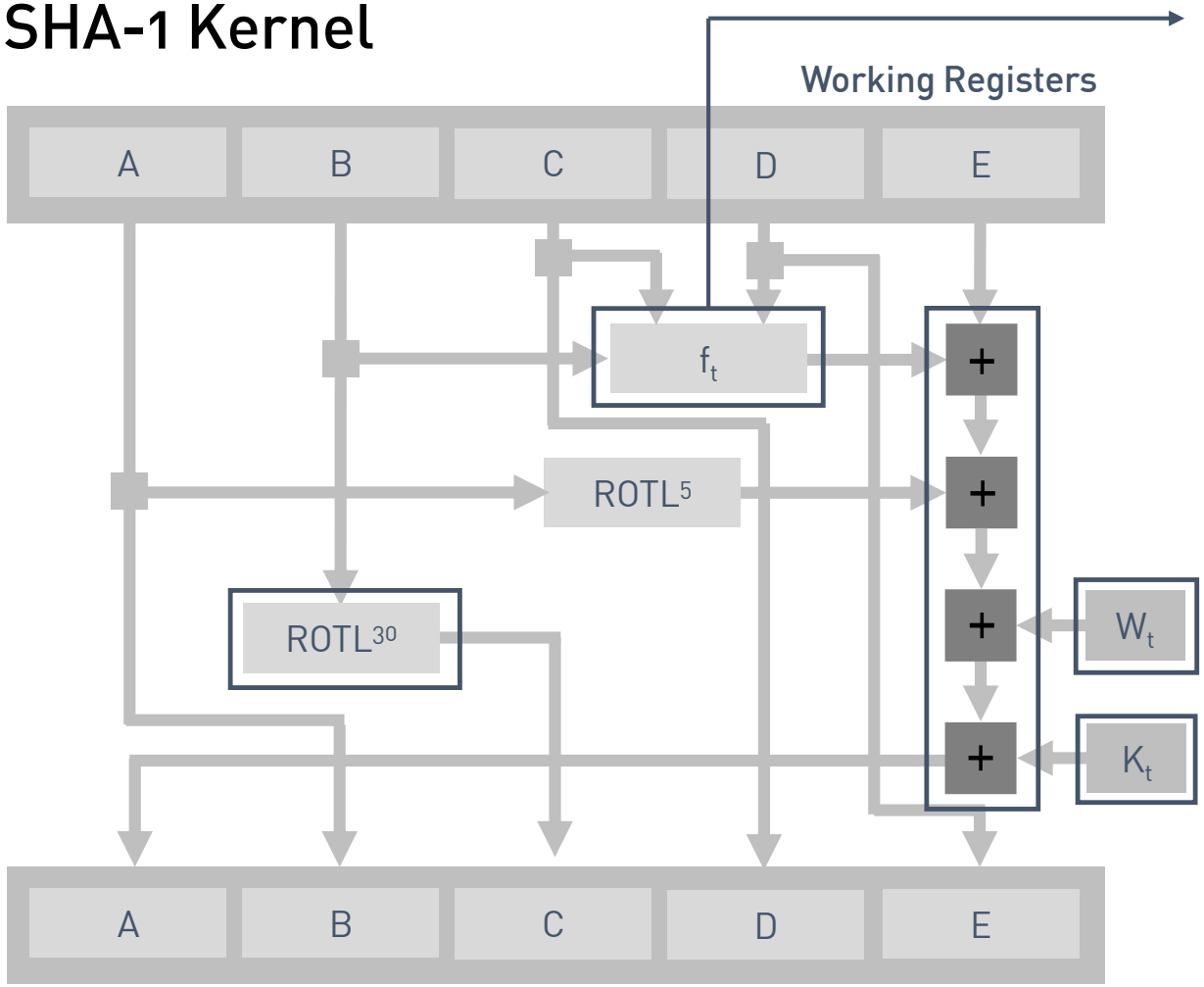
Prof. Dr. C. Jakob

University of Applied Sciences Darmstadt
h_da
Faculty of Electrical Engineering and Information Technology
fbeit

SHA-1 Kernel

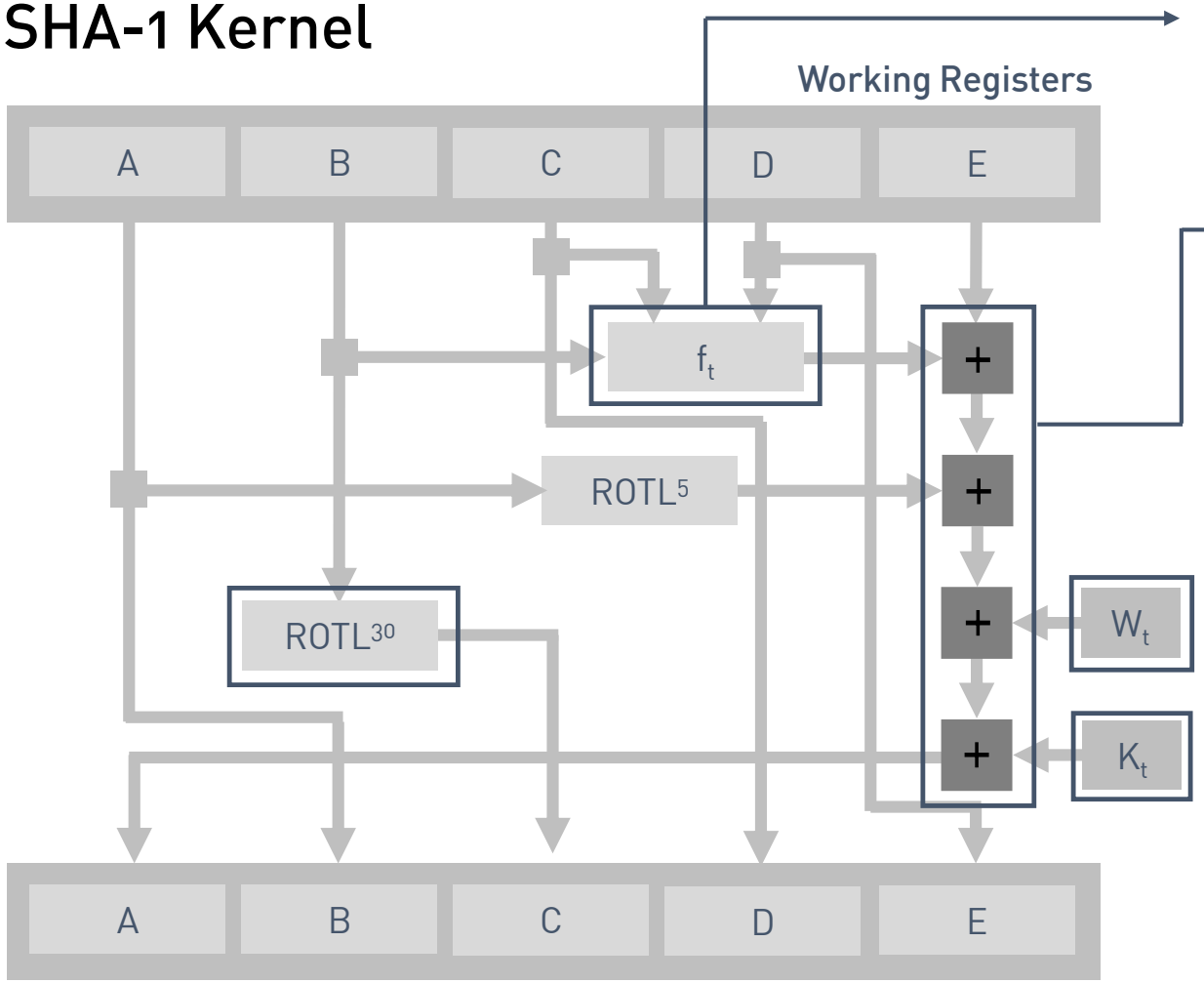


SHA-1 Kernel



$$f_t(x,y,z) = \begin{cases} Ch(x,y,z)=(x \wedge y) \oplus (\neg x \wedge z) & 0 \le t \le 19 \\ Parity(x,y,z)=x \oplus y \oplus z & 20 \le t \le 39 \\ Maj(x,y,z)=(x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) & 40 \le t \le 59 \\ Parity(x,y,z)=x \oplus y \oplus z & 60 \le t \le 79 \end{cases}$$

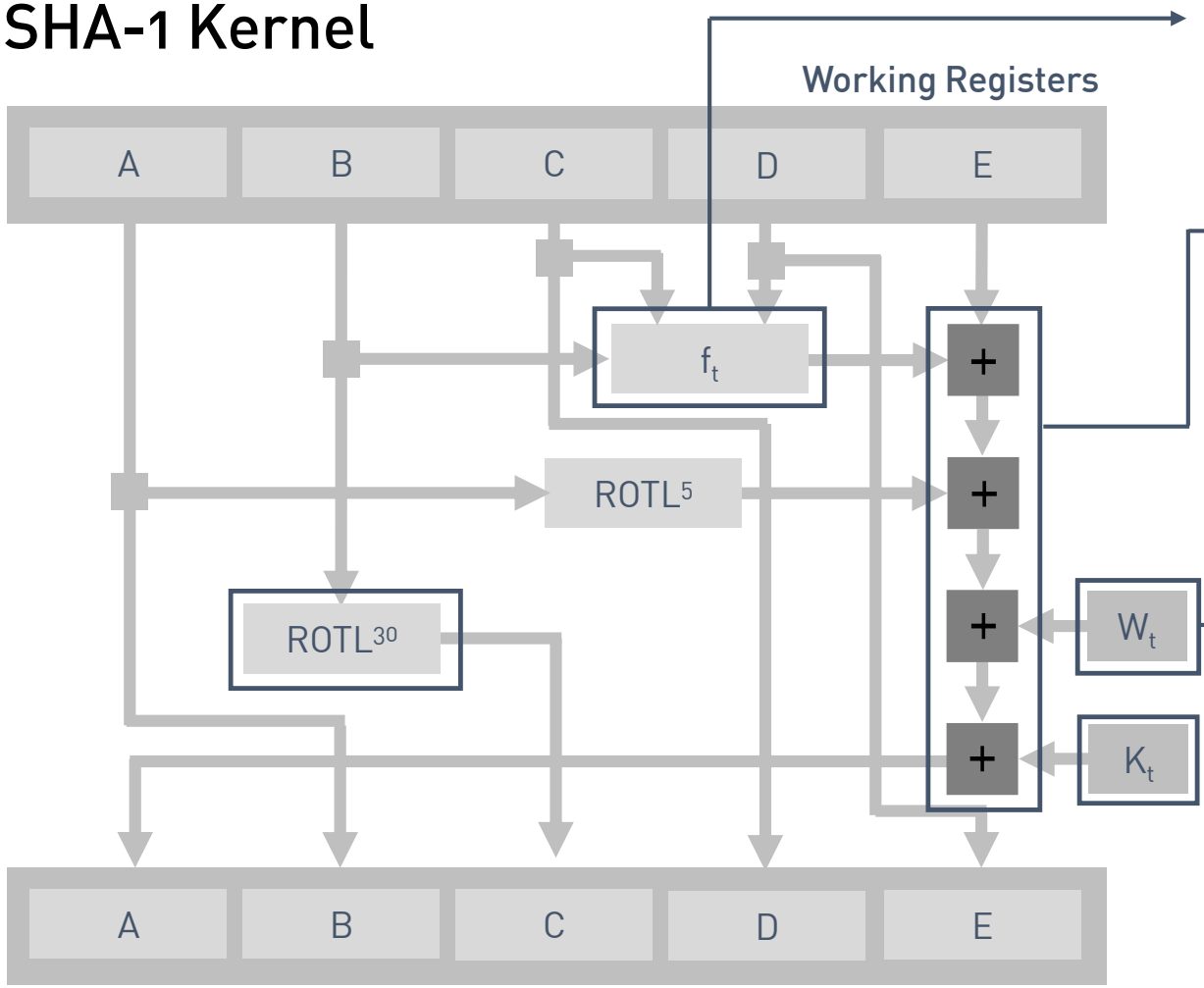
SHA-1 Kernel



$$f_t(x,y,z) = \begin{cases} Ch(x,y,z)=(x \wedge y) \oplus (\neg x \wedge z) & 0 \le t \le 19 \\ Parity(x,y,z)=x \oplus y \oplus z & 20 \le t \le 39 \\ Maj(x,y,z)=(x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) & 40 \le t \le 59 \\ Parity(x,y,z)=x \oplus y \oplus z & 60 \le t \le 79 \end{cases}$$

Addition (+) is performed modulo 2^{32}

SHA-1 Kernel

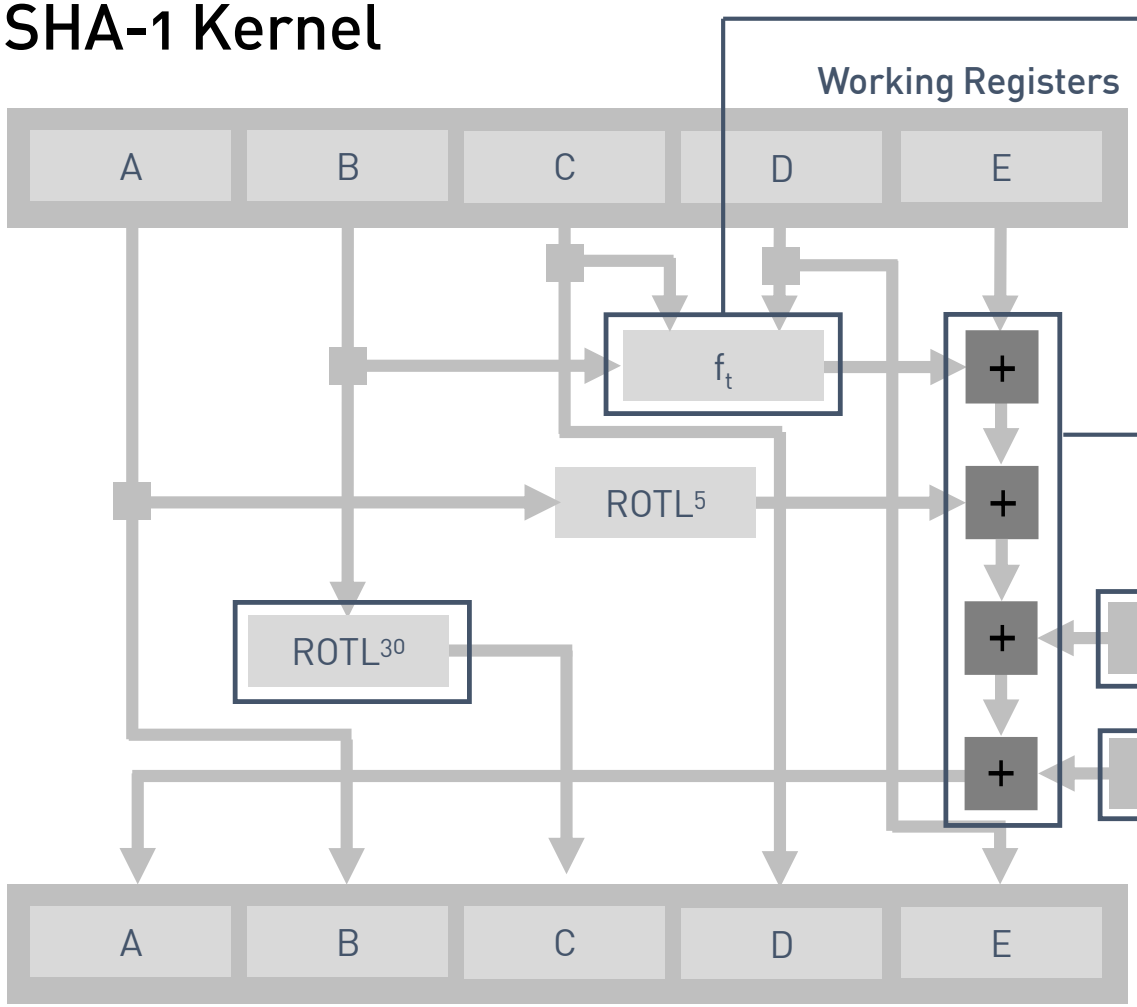


$$f_t(x,y,z) = \begin{cases} Ch(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z) & 0 \le t \le 19 \\ Parity(x,y,z) = x \oplus y \oplus z & 20 \le t \le 39 \\ Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) & 40 \le t \le 59 \\ Parity(x,y,z) = x \oplus y \oplus z & 60 \le t \le 79 \end{cases}$$

Addition (+) is performed modulo 2^{32}

$$W_t = \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ ROTL^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) & 16 \le t \le 79 \end{cases}$$

SHA-1 Kernel



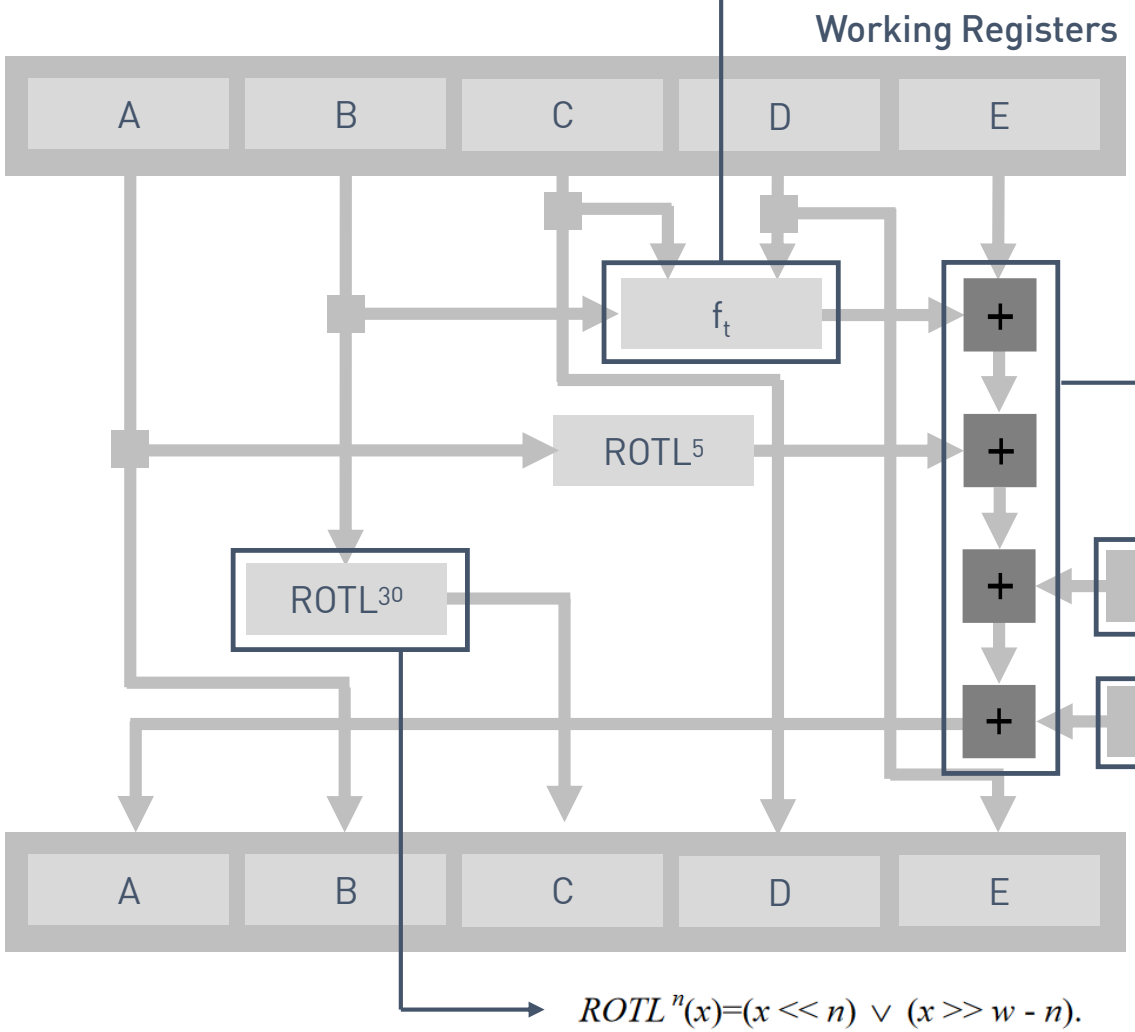
$$f_t(x,y,z) = \begin{cases} Ch(x,y,z)=(x \wedge y) \oplus (\neg x \wedge z) & 0 \le t \le 19 \\ Parity(x,y,z)=x \oplus y \oplus z & 20 \le t \le 39 \\ Maj(x,y,z)=(x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) & 40 \le t \le 59 \\ Parity(x,y,z)=x \oplus y \oplus z & 60 \le t \le 79 \end{cases}$$

Addition (+) is performed modulo 2^{32}

$$W_t = \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ ROTL^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) & 16 \le t \le 79 \end{cases}$$

$$K_t = \begin{cases} \texttt{5a827999} & 0 \le t \le 19 \\ \texttt{6ed9eba1} & 20 \le t \le 39 \\ \texttt{8f1bbcdc} & 40 \le t \le 59 \\ \texttt{ca62c1d6} & 60 \le t \le 79 \end{cases}$$

SHA-1 Kernel



$$f_t(x,y,z) = \begin{cases} Ch(x,y,z)=(x \wedge y) \oplus (\neg x \wedge z) & 0 \le t \le 19 \\ Parity(x,y,z)=x \oplus y \oplus z & 20 \le t \le 39 \\ Maj(x,y,z)=(x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) & 40 \le t \le 59 \\ Parity(x,y,z)=x \oplus y \oplus z & 60 \le t \le 79 \end{cases}$$

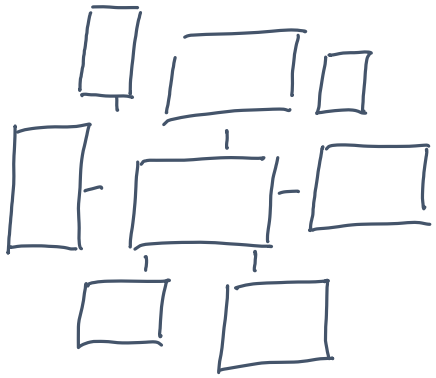
Addition (+) is performed modulo 2^{32}

$$W_t = \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ ROTL^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) & 16 \le t \le 79 \end{cases}$$

$$K_t = \begin{cases} \texttt{5a827999} & 0 \le t \le 19 \\ \texttt{6ed9eba1} & 20 \le t \le 39 \\ \texttt{8f1bbcdc} & 40 \le t \le 59 \\ \texttt{ca62c1d6} & 60 \le t \le 79 \end{cases}$$

$$ROTL^n(x) = (x << n) \vee (x >> w - n).$$

Thus, $ROTL^n(x)$ is equivalent to a circular shift (rotation) of x by n positions to the left.



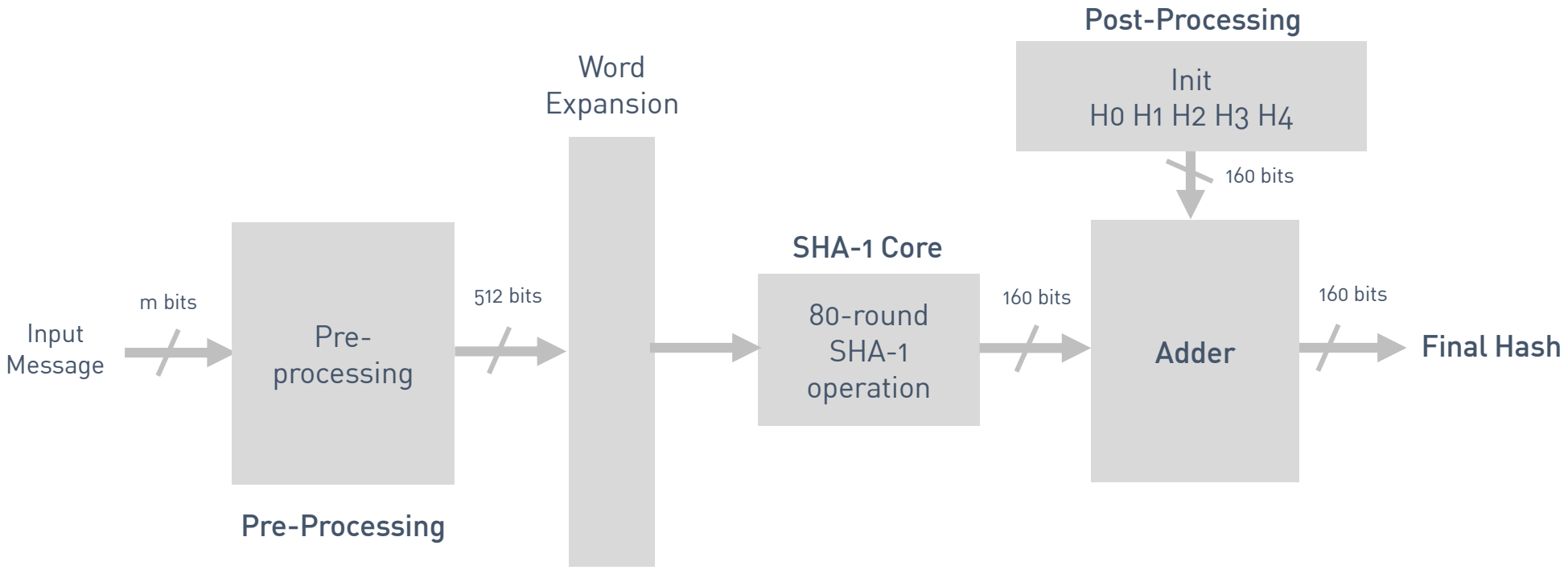
SHA-1 – Top-Level Perspective

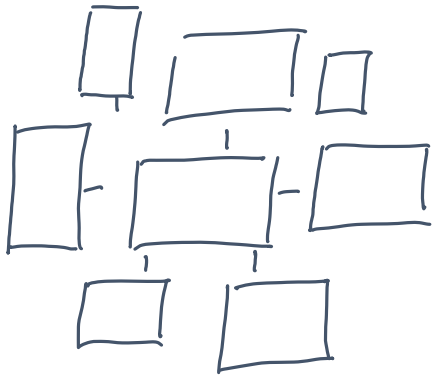
Bachelor of Science in Electrical Engineering

Prof. Dr. C. Jakob

University of Applied Sciences Darmstadt
h_da
Faculty of Electrical Engineering and Information Technology
fbeit

Simplified Block Diagram





SHA-1 – SW Verification

Bachelor of Science in Electrical Engineering

Prof. Dr. C. Jakob

University of Applied Sciences Darmstadt
h_da
Faculty of Electrical Engineering and Information Technology
fbeit

How to verify your implementation?

Taken from: NIST, Cryptographic Standards and Guidelines, https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Standards-and-Guidelines/documents/examples/SHA1.pdf

One Block Message Sample

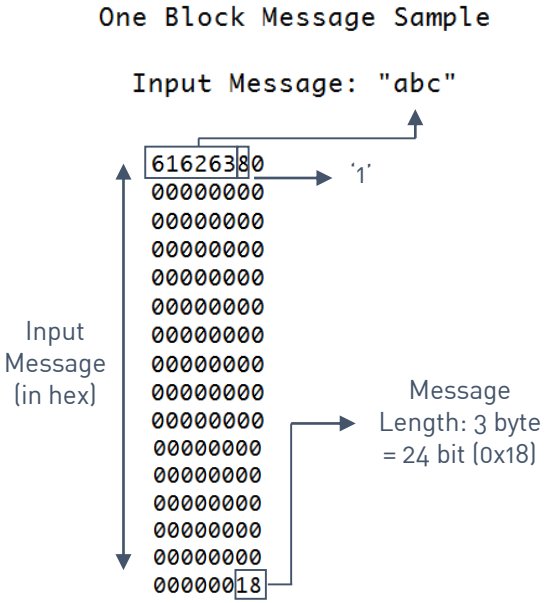
Input Message: "abc"

Input Message (in hex)

```
61626380
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000018
```

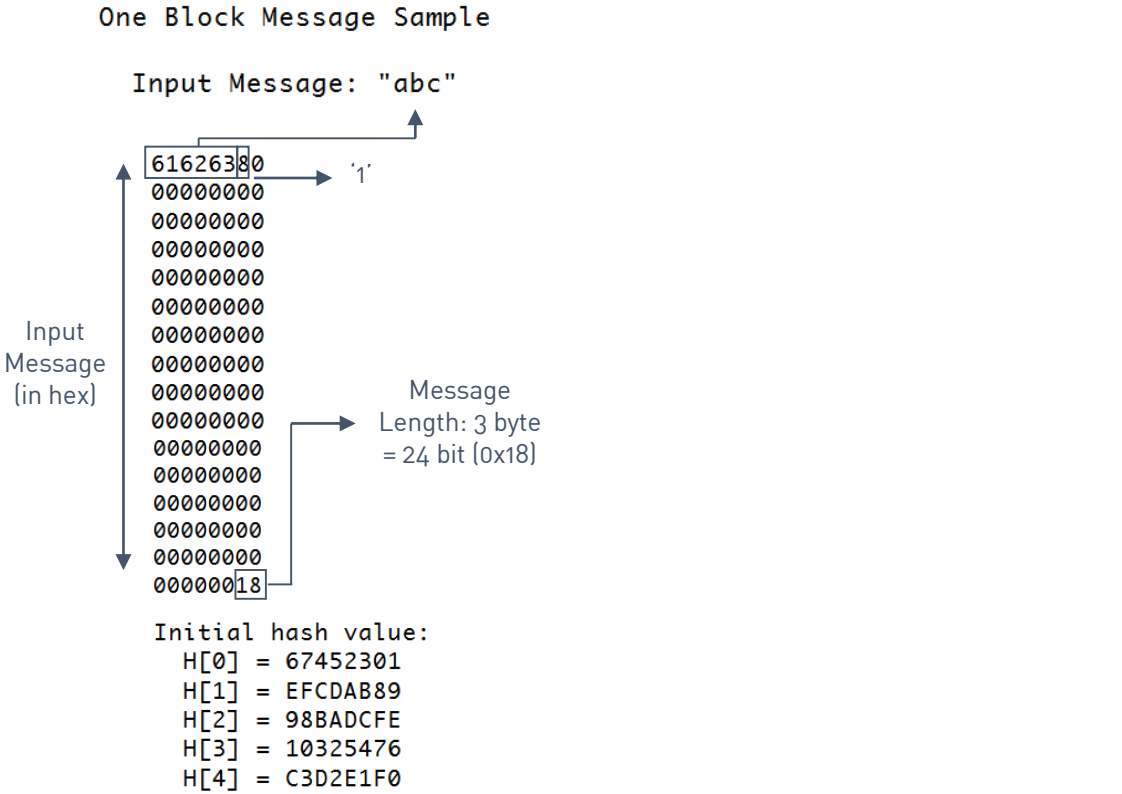
How to verify your implementation?

Taken from: NIST, Cryptographic Standards and Guidelines, https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Standards-and-Guidelines/documents/examples/SHA1.pdf



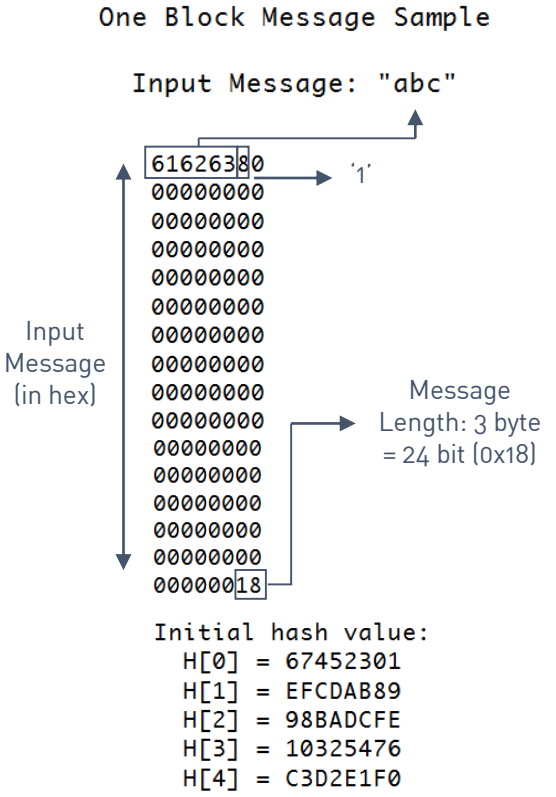
How to verify your implementation?

Taken from: NIST, Cryptographic Standards and Guidelines, https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Standards-and-Guidelines/documents/examples/SHA1.pdf



How to verify your implementation?

Taken from: NIST, Cryptographic Standards and Guidelines, https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Standards-and-Guidelines/documents/examples/SHA1.pdf



	A	B	C	D	E
t= 0:	0116FC33	67452301	7BF36AE2	98BADCFE	10325476
t= 1:	8990536D	0116FC33	59D148C0	7BF36AE2	98BADCFE
t= 2:	A1390F08	8990536D	C045BF0C	59D148C0	7BF36AE2
t= 3:	CDD8E11B	A1390F08	626414DB	C045BF0C	59D148C0
t= 4:	CFD499DE	CDD8E11B	284E43C2	626414DB	C045BF0C
t= 5:	3FC7CA40	CFD499DE	F3763846	284E43C2	626414DB
t= 6:	993E30C1	3FC7CA40	B3F52677	F3763846	284E43C2
t= 7:	9E8C07D4	993E30C1	0FF1F290	B3F52677	F3763846
t= 8:	4B6AE328	9E8C07D4	664F8C30	0FF1F290	B3F52677
t= 9:	8351F929	4B6AE328	27A301F5	664F8C30	0FF1F290
t=10:	FBDA9E89	8351F929	12DAB8CA	27A301F5	664F8C30
t=11:	63188FE4	FBDA9E89	60D47E4A	12DAB8CA	27A301F5
t=12:	4607B664	63188FE4	7EF6A7A2	60D47E4A	12DAB8CA
t=13:	9128F695	4607B664	18C623F9	7EF6A7A2	60D47E4A
t=14:	196BEE77	9128F695	1181ED99	18C623F9	7EF6A7A2
t=15:	20BDD62F	196BEE77	644A3DA5	1181ED99	18C623F9
t=16:	4E925823	20BDD62F	C65AFB9D	644A3DA5	1181ED99
t=17:	82AA6728	4E925823	C82F758B	C65AFB9D	644A3DA5
t=18:	DC64901D	82AA6728	D3A49608	C82F758B	C65AFB9D
t=19:	FD9E1D7D	DC64901D	20AA99CA	D3A49608	C82F758B
t=20:	1A37B0CA	FD9E1D7D	77192407	20AA99CA	D3A49608
t=21:	33A23BFC	1A37B0CA	7F67875F	77192407	20AA99CA
t=22:	21283486	33A23BFC	868DEC32	7F67875F	77192407
t=23:	D541F12D	21283486	0CE88EFF	868DEC32	7F67875F
t=24:	C7567DC6	D541F12D	884A0D21	0CE88EFF	868DEC32
t=25:	48413BA4	C7567DC6	75507C4B	884A0D21	0CE88EFF
t=26:	BE35FBD5	48413BA4	B1D59F71	75507C4B	884A0D21
t=27:	4AA84D97	BE35FBD5	12104EE9	B1D59F71	75507C4B
t=28:	8370B52E	4AA84D97	6F8D7EF5	12104EE9	B1D59F71
t=29:	C5FBAF5D	8370B52E	D2AA1365	6F8D7EF5	12104EE9
t=30:	1267B407	C5FBAF5D	A0DC2D4B	D2AA1365	6F8D7EF5
t=31:	3B845D33	1267B407	717EEBD7	A0DC2D4B	D2AA1365
t=32:	046FAA0A	3B845D33	C499ED01	717EEBD7	A0DC2D4B
t=33:	2C0EBC11	046FAA0A	CEE1174C	C499ED01	717EEBD7
t=34:	21796AD4	2C0EBC11	811BEA82	CEE1174C	C499ED01
t=35:	DCBBB0CB	21796AD4	4B03AF04	811BEA82	CEE1174C
t=36:	0F511FD8	DCBBB0CB	085E5AB5	4B03AF04	811BEA82
t=37:	DC63973F	0F511FD8	F72EEC32	085E5AB5	4B03AF04
t=38:	4C986405	DC63973F	03D447F6	F72EEC32	085E5AB5
t=39:	32DE1CBA	4C986405	F718E5CF	03D447F6	F72EEC32
t=40:	FC87DEDF	32DE1CBA	53261901	F718E5CF	03D447F6
t=41:	970A0D5C	FC87DEDF	8CB7872E	53261901	F718E5CF
t=42:	7F193DC5	970A0D5C	FF21F7B7	8CB7872E	53261901
t=43:	EE1B1AAF	7F193DC5	25C28357	FF21F7B7	8CB7872E
t=44:	40F28E09	EE1B1AAF	5FC64F71	25C28357	FF21F7B7
t=45:	1C51E1F2	40F28E09	FB86C6AB	5FC64F71	25C28357
t=46:	A01B846C	1C51E1F2	503CA382	FB86C6AB	5FC64F71
t=47:	BEAD02CA	A01B846C	8714787C	503CA382	FB86C6AB
t=48:	BAF39337	BEAD02CA	2806E11B	8714787C	503CA382
t=49:	120731C5	BAF39337	AFAB40B2	2806E11B	8714787C
t=50:	641DB2CE	120731C5	EEBCE4CD	AFAB40B2	2806E11B

How to verify your implementation?

Taken from: NIST, Cryptographic Standards and Guidelines, https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Standards-and-Guidelines/documents/examples/SHA1.pdf

```
t=51: 3847AD66  641DB2CE  4481CC71  EEBCE4CD  AFAB40B2
t=52: E490436D  3847AD66  99076CB3  4481CC71  EEBCE4CD
t=53: 27E9F1D8  E490436D  8E11EB59  99076CB3  4481CC71
t=54: 7B71F76D  27E9F1D8  792410DB  8E11EB59  99076CB3
t=55: 5E6456AF  7B71F76D  09FA7C76  792410DB  8E11EB59
t=56: C846093F  5E6456AF  5EDC7DDB  09FA7C76  792410DB
t=57: D262FF50  C846093F  D79915AB  5EDC7DDB  09FA7C76
t=58: 09D785FD  D262FF50  F211824F  D79915AB  5EDC7DDB
t=59: 3F52DE5A  09D785FD  3498BFD4  F211824F  D79915AB
t=60: D756C147  3F52DE5A  4275E17F  3498BFD4  F211824F
t=61: 548C9CB2  D756C147  8FD4B796  4275E17F  3498BFD4
t=62: B66C020B  548C9CB2  F5D5B051  8FD4B796  4275E17F
t=63: 6B61C9E1  B66C020B  9523272C  F5D5B051  8FD4B796
t=64: 19DFA7AC  6B61C9E1  ED9B0082  9523272C  F5D5B051
t=65: 101655F9  19DFA7AC  5AD87278  ED9B0082  9523272C
t=66: 0C3DF2B4  101655F9  0677E9EB  5AD87278  ED9B0082
t=67: 78DD4D2B  0C3DF2B4  4405957E  0677E9EB  5AD87278
t=68: 497093C0  78DD4D2B  030F7CAD  4405957E  0677E9EB
t=69: 3F2588C2  497093C0  DE37534A  030F7CAD  4405957E
t=70: C199F8C7  3F2588C2  125C24F0  DE37534A  030F7CAD
t=71: 39859DE7  C199F8C7  8FC96230  125C24F0  DE37534A
t=72: EDB42DE4  39859DE7  F0667E31  8FC96230  125C24F0
t=73: 11793F6F  EDB42DE4  CE616779  F0667E31  8FC96230
t=74: 5EE76897  11793F6F  3B6D0B79  CE616779  F0667E31
t=75: 63F7DAB7  5EE76897  C45E4FDB  3B6D0B79  CE616779
t=76: A079B7D9  63F7DAB7  D7B9DA25  C45E4FDB  3B6D0B79
t=77: 860D21CC  A079B7D9  D8FDF6AD  D7B9DA25  C45E4FDB
t=78: 5738D5E1  860D21CC  681E6DF6  D8FDF6AD  D7B9DA25
t=79: 42541B35  5738D5E1  21834873  681E6DF6  D8FDF6AD
```

```
H[0] = 67452301 + 42541B35 = A9993E36
H[1] = EFCDAB89 + 5738D5E1 = 4706816A
H[2] = 98BADCFE + 21834873 = BA3E2571
H[3] = 10325476 + 681E6DF6 = 7850C26C
H[4] = C3D2E1F0 + D8FDF6AD = 9CD0D89D

Initial hash value:
  H[0] = 67452301
  H[1] = EFCDAB89
  H[2] = 98BADCFE
  H[3] = 10325476
  H[4] = C3D2E1F0
```

Final hash

```
A9993E36 4706816A BA3E2571 7850C26C 9CD0D89D
```

SHA-1 Applications ...

Semester Project WS23/24

fbeit
FACULTY OF ELECTRICAL ENGINEERING
AND INFORMATION TECHNOLOGY
Prof. Dr. C. Jakob

SHA-1 Applications on Wikipedia

Cryptography [edit]

Further information: Cryptographic hash function § Applications

SHA-1 forms part of several widely used security applications and protocols, including TLS and SSL, PGP, SSH, S/MIME, and IPsec. Those applications can also use MD5; both MD5 and SHA-1 are descended from MD4.

Data integrity [edit]

Revision control systems such as Git, Mercurial, and Monotone use SHA-1 not for security but to identify revisions and to ensure that the data has not changed due to accidental corruption. Linus Torvalds said about Git:

> If you have disk corruption, if you have DRAM corruption, if you have any kind of problems at all, Git will notice them. It's not a question of *if*, it's a guarantee. You can have people who try to be malicious. They won't succeed. ... Nobody has been able to break SHA-1, but the point is the SHA-1, as far as Git is concerned, isn't even a security feature. It's purely a consistency check. The security parts are elsewhere, so a lot of people assume that since Git uses SHA-1 and SHA-1 is used for cryptographically secure stuff, they think that, Okay, it's a huge security feature. It has nothing at all to do with security, it's just the best hash you can get.
> ...
> I guarantee you, if you put your data in Git, you can trust the fact that five years later, after it was converted from your hard disk to DVD to whatever new technology and you copied it along, five years later you can verify that the data you get back out is the exact same data you put in. ...
> One of the reasons I care is for the kernel, we had a break in on one of the BitKeeper sites where people tried to corrupt the kernel source code repositories.[22]
> However Git does not require the second preimage resistance of SHA-1 as a security feature, since it will always prefer to keep the earliest version of an object in case of collision, preventing an attacker from surreptitiously overwriting files.[23]

ESY Semester Project WS23/24

h_da – fbeit – Embedded Systems

Theoretical Questions

Semester Project WS23/24

fbeit

FACULTY OF ELECTRICAL ENGINEERING
AND INFORMATION TECHNOLOGY

Prof. Dr. C. Jakob

Theoretical Questions

Eine Auswahl der nachfolgenden Fragestellungen kann im Rahmen der schriftlichen Ausarbeitung diskutiert werden.

- Was ist eine mathematische Einwegfunktion?

Theoretical Questions

Eine Auswahl der nachfolgenden Fragestellungen kann im Rahmen der schriftlichen Ausarbeitung diskutiert werden.

- Was ist eine mathematische Einwegfunktion?
- Was sind typische Anwendungen von Einwegfunktionen?

Theoretical Questions

Eine Auswahl der nachfolgenden Fragestellungen kann im Rahmen der schriftlichen Ausarbeitung diskutiert werden.

- Was ist eine mathematische Einwegfunktion?
- Was sind typische Anwendungen von Einwegfunktionen?
- Definieren Sie die „Preimage Resistance“ und die „Second Preimage Resistance“-Charakteristik einer Einwegfunktion.

Theoretical Questions

Eine Auswahl der nachfolgenden Fragestellungen kann im Rahmen der schriftlichen Ausarbeitung diskutiert werden.

- Was ist eine mathematische Einwegfunktion?
- Was sind typische Anwendungen von Einwegfunktionen?
- Definieren Sie die „Preimage Resistance“ und die „Second Preimage Resistance“-Charakteristik einer Einwegfunktion.
- Was versteht man unter einet Kollision und wie wirkt sie sich auf die Sicherheit einer Hash-Funktion aus? Warum gibt es zwangsläufig Kollisionen?

Theoretical Questions

Eine Auswahl der nachfolgenden Fragestellungen kann im Rahmen der schriftlichen Ausarbeitung diskutiert werden.

- Was ist eine mathematische Einwegfunktion?
- Was sind typische Anwendungen von Einwegfunktionen?
- Definieren Sie die „Preimage Resistance“ und die „Second Preimage Resistance“-Charakteristik einer Einwegfunktion.
- Was versteht man unter einet Kollision und wie wirkt sie sich auf die Sicherheit einer Hash-Funktion aus? Warum gibt es zwangsläufig Kollisionen?
- **Stellt die boolesche XOR-Funktion eine sinnvolle Methode dar, um die Integrität einer Nachricht zu überprüfen? Begründen Sie Ihre Antwort!**

Theoretical Questions

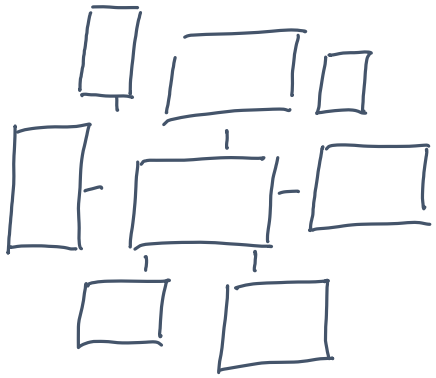
Eine Auswahl der nachfolgenden Fragestellungen kann im Rahmen der schriftlichen Ausarbeitung diskutiert werden.

- Was ist eine mathematische Einwegfunktion?
- Was sind typische Anwendungen von Einwegfunktionen?
- Definieren Sie die „Preimage Resistance“ und die „Second Preimage Resistance“-Charakteristik einer Einwegfunktion.
- Was versteht man unter einet Kollision und wie wirkt sie sich auf die Sicherheit einer Hash-Funktion aus? Warum gibt es zwangsläufig Kollisionen?
- Stellt die boolesche XOR-Funktion eine sinnvolle Methode dar, um die Integrität einer Nachricht zu überprüfen? Begründen Sie Ihre Antwort!
- Erläutern Sie das Geburtstagsproblem und nennen Sie den Zusammenhang mit Hash-Kollisionen.

Theoretical Questions

Eine Auswahl der nachfolgenden Fragestellungen kann im Rahmen der schriftlichen Ausarbeitung diskutiert werden.

- Was ist eine mathematische Einwegfunktion?
- Was sind typische Anwendungen von Einwegfunktionen?
- Definieren Sie die „Preimage Resistance“ und die „Second Preimage Resistance“-Charakteristik einer Einwegfunktion.
- Was versteht man unter einet Kollision und wie wirkt sie sich auf die Sicherheit einer Hash-Funktion aus? Warum gibt es zwangsläufig Kollisionen?
- Stellt die boolesche XOR-Funktion eine sinnvolle Methode dar, um die Integrität einer Nachricht zu überprüfen? Begründen Sie Ihre Antwort!
- Erläutern Sie das Geburtstagsproblem und nennen Sie den Zusammenhang mit Hash-Kollisionen.
- Welche Rolle spielen mathematische Einwegfunktionen im Kontext mit der Kryptowährung Bitcoin?



Embedded Systems

Bachelor of Science in Electrical Engineering

Prof. Dr. C. Jakob

University of Applied Sciences Darmstadt

h_da

Faculty of Electrical Engineering and Information Technology

fbeit

ESY Semester Project WS23/24

Project specific Readings

Literature

- National Institute of Standards and Technology NIST: FIPS PUB 180-4: Secure Hash Standard, 2012, (Online).
- Paar, C. & Pelzl, J.: Understanding Cryptography. A Textbook for Students and Practitioners, pp. 307-312, Springer, 2009, (SpringerLink).
- Delfs , H. & Knebl, H.: Introduction to Cryptography: Principles and Applications, Information Security and Cryptography, Springer 2015, (SpringerLink).

Recommended Readings and Online Resources

- [**Online**]: Das CrypTool-Portal - https://www.cryptool.org/de/
- [**Book**]: Security Engineering, Ross Anderson - http://www.cl.cam.ac.uk/~rja14/book.html
- [**Book**]: Applied Cryptography, Bruce Schneier - https://www.schneier.com/books/applied_cryptography
- [**Book**]: Understanding Cryptography : A Textbook for Students and Practitioners – SpringerLink
- [**Book**]: Practical Cryptography in Python - SpringerLink
- [**Online**]: Awesome security - an open repository - https://github.com/sbilly/awesome-security
- [**Online**]: CVE security vulnerability database/information source - http://www.cvedetails.com