

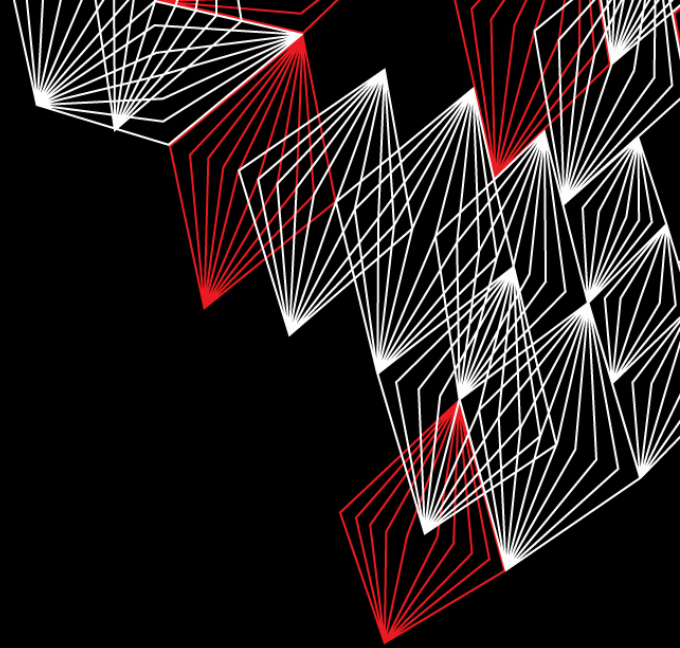
UNIVERSITY OF TWENTE.

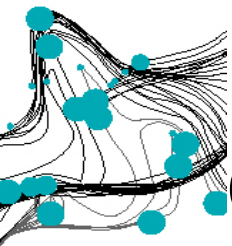
PROGRAMMING

P5.3: SECURITY ENGINEERING (2)

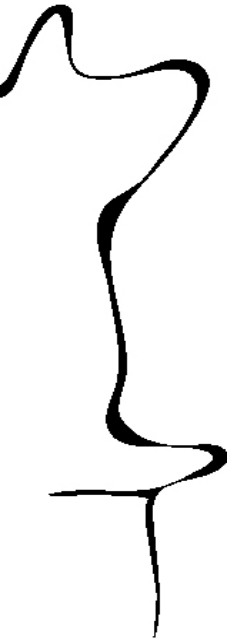
MODULE 1.2: SOFTWARE SYSTEMS

DECEMBER 13, 2019 – MAARTEN EVERTS





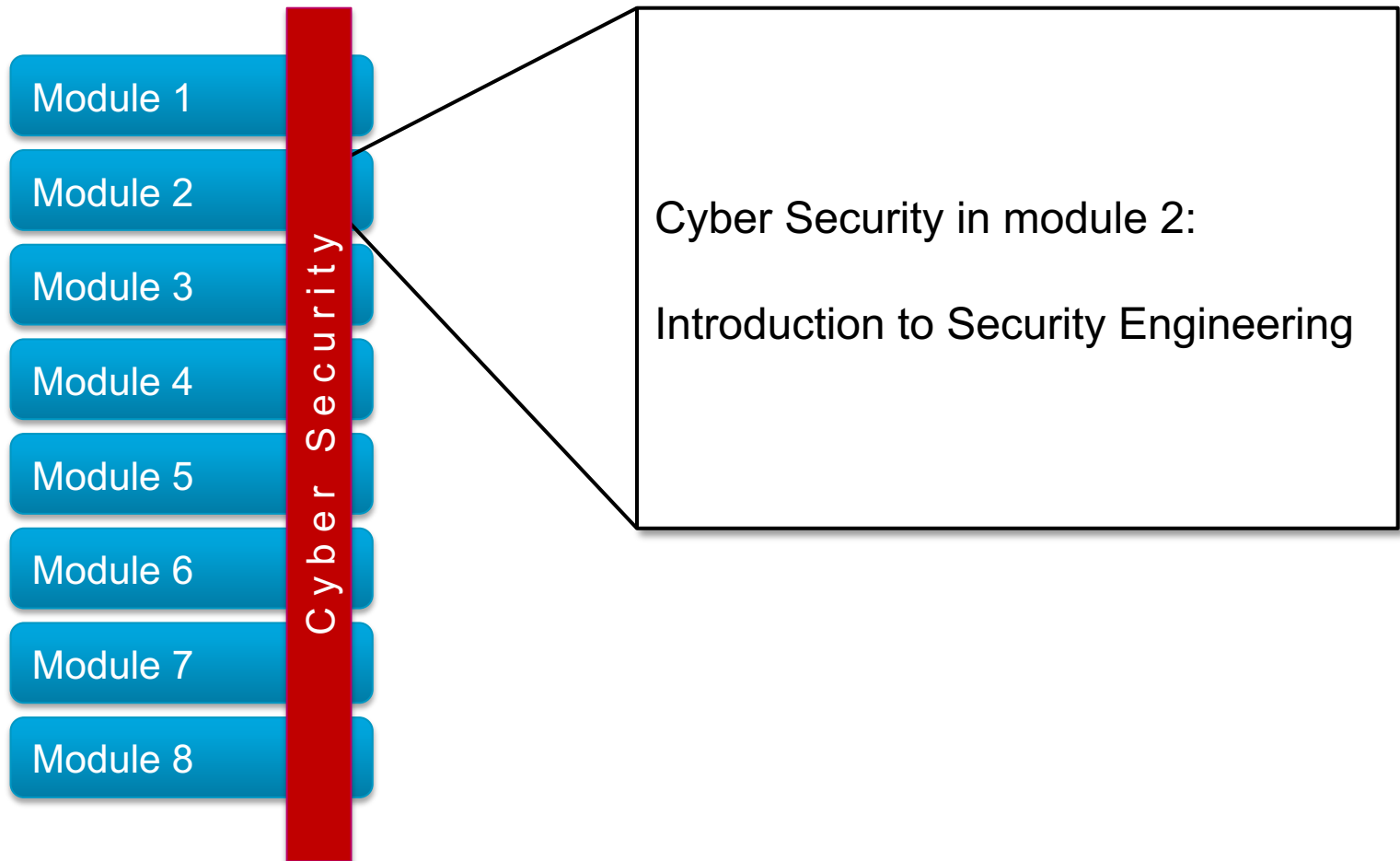
OVERVIEW PROGRAMMING LINE



Week 1 Values and variables Classes and objects	Week 2 Classes and objects Testing	Week 3 Interfaces and Inheritance Subtyping Security 1
Week 4 Arrays and lists List implementations Collections	Week 5 Stream I/O and MVC Exceptions Security 2	Week 6 Concurrency Project kick-off IDE Tips & Tricks
Week 7 Basic Networking Networking and multithreading GUIs	Week 8/9 Advanced Java facilities Test	Week 10 Project Test resit



CROSS-CUTTING CONCERN *CYBER SECURITY*



SECURITY PROPERTIES (RECAP)

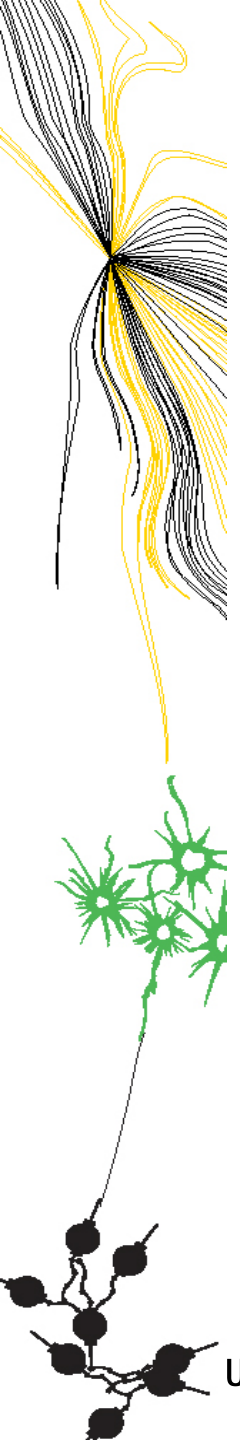
Confidentiality

Integrity

Availability

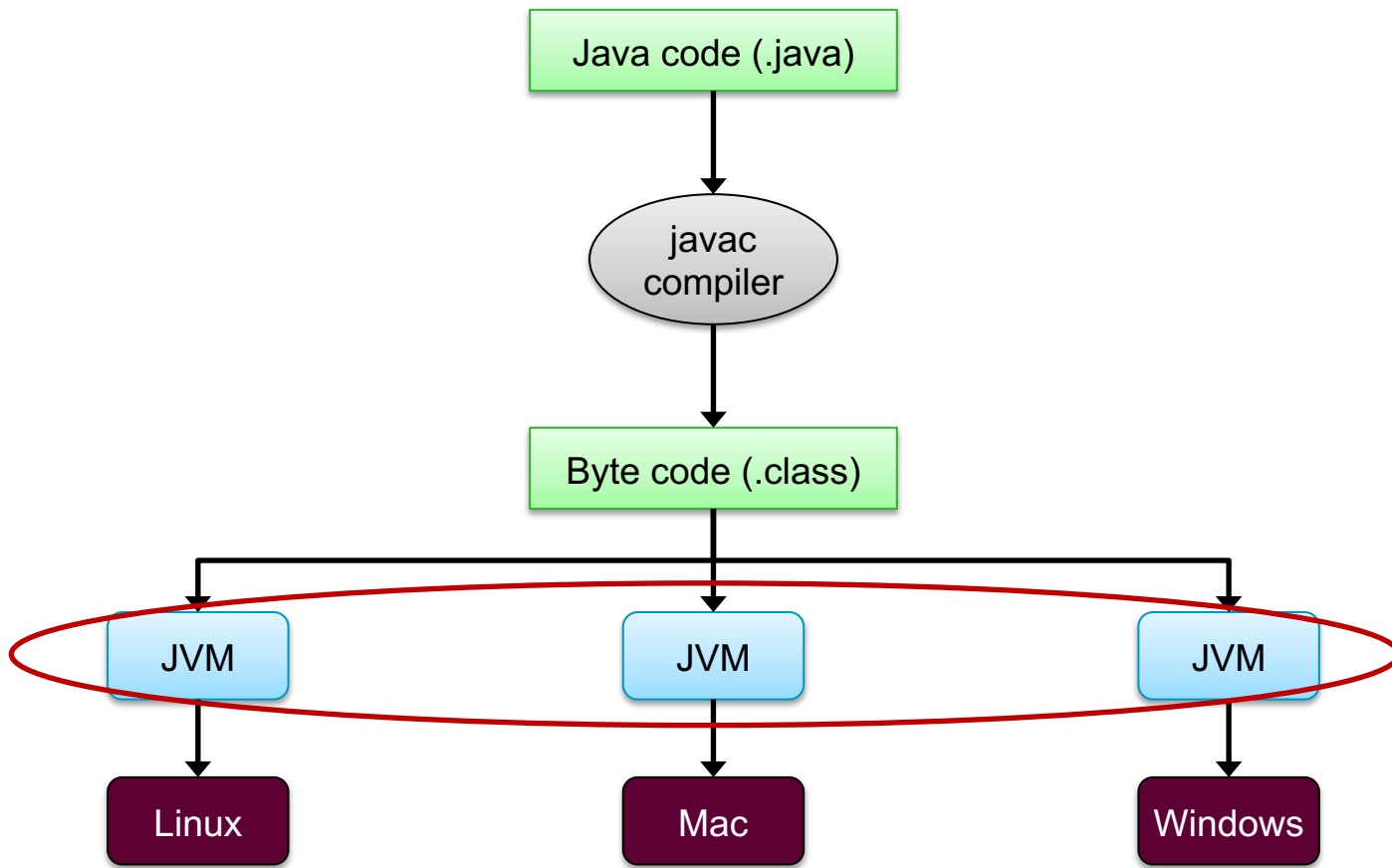
TODAY

- Java & Security
- Binary-to-text encodings (Hex & Base64)
- Hash functions
- Commitment (hash function application)
- HMAC (another hash function application)
- Side-channel attacks

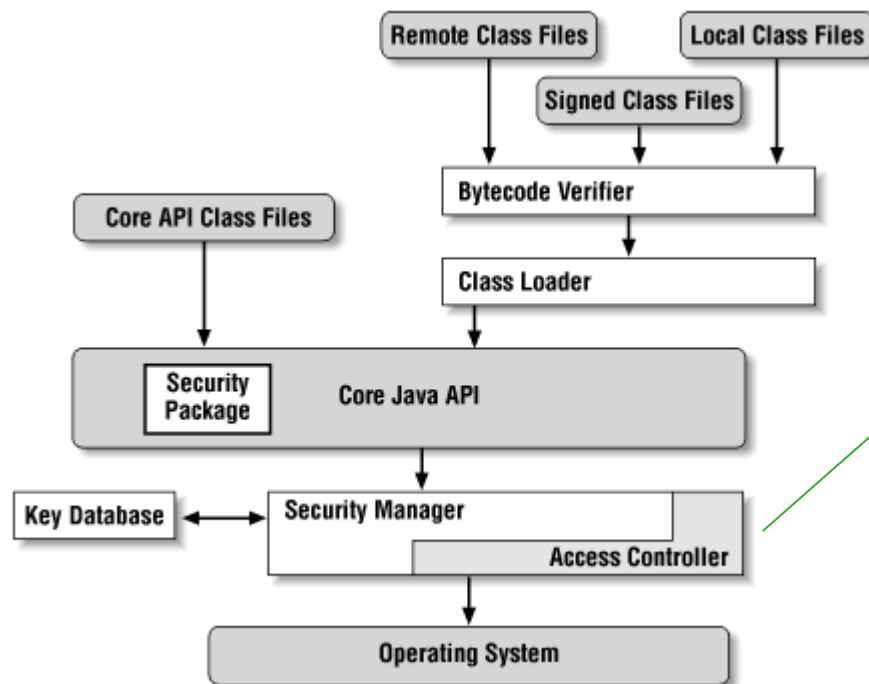


JAVA & SECURITY

SIMPLIFIED JAVA OVERVIEW



JVM SECURITY OVERVIEW



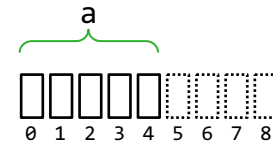
- Manages what resources (network, files, etc.) code can access.
- Allows for the creation of sandboxes.
- Has a bad track-record! (e.g., security of browser applets).

From: http://www.onjava.com/pub/a/onjava/excerpt/java_security_ch1/index.html?page=4

JAVA'S SECURITY ADVANTAGES (COMPARED TO C)

Runtime constraints & bounds checking

```
int a[] = new int[5];  
a[8] = 3;
```

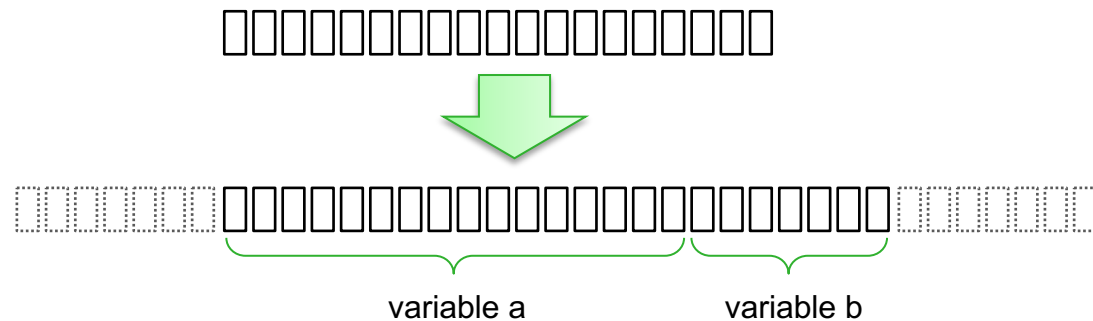


```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 8  
    at module2.security.example.Main.main(Main.java:7)  
    ...
```

JAVA'S SECURITY ADVANTAGES (COMPARED TO C)

- Less susceptible to buffer overflows

This could happen in C:



- No pointer arithmetic in Java

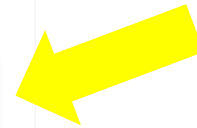
Brief Listing of the Top 25

This is a brief listing of the Top 25 items, using the general ranking.

NOTE: 16 other weaknesses were considered for inclusion in the Top 25, but their general scores were not high enough. They are listed in a separate ["On the Cusp"](#) page.

Rank	Score	ID	Name
[1]	93.8	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
[2]	83.3	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
[3]	79.0	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
[4]	77.7	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
[5]	76.9	CWE-306	Missing Authentication for Critical Function
[6]	76.8	CWE-862	Missing Authorization
[7]	75.0	CWE-798	Use of Hard-coded Credentials
[8]	75.0	CWE-311	Missing Encryption of Sensitive Data
[9]	74.0	CWE-434	Unrestricted Upload of File with Dangerous Type
[10]	73.8	CWE-807	Reliance on Untrusted Inputs in a Security Decision
[11]	73.1	CWE-250	Execution with Unnecessary Privileges
[12]	70.1	CWE-352	Cross-Site Request Forgery (CSRF)
[13]	69.3	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
[14]	68.5	CWE-494	Download of Code Without Integrity Check
[15]	67.8	CWE-863	Incorrect Authorization

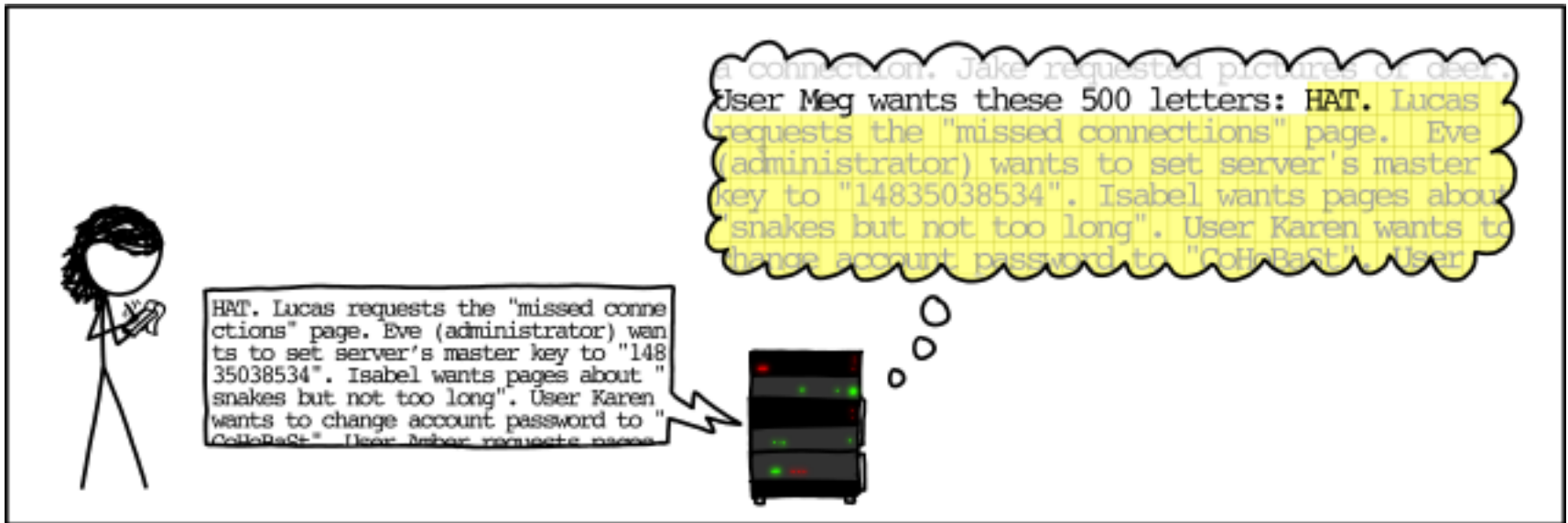
<http://cwe.mitre.org/top25/>



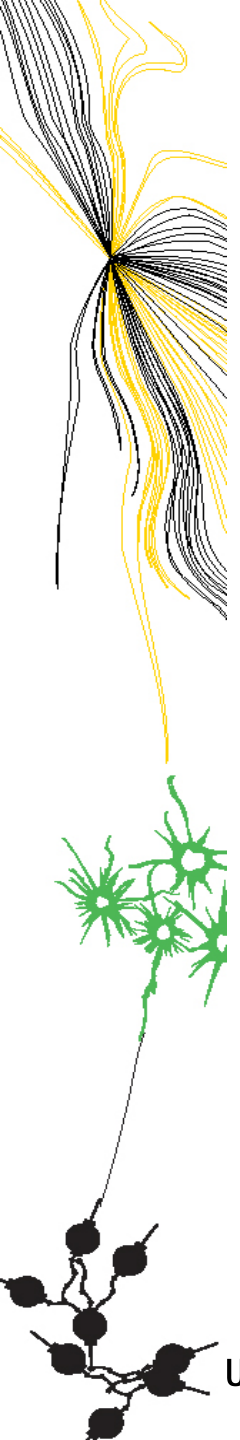
EXAMPLE OF ADVANTAGES OF BOUNDS CHECKING



HOW THE HEARTBLEED BUG WORKS:



From: <http://xkcd.com/1354/>



BINARY-TO-TEXT ENCODINGS

REPRESENTING BINARY DATA (BYTE ARRAYS)

A byte-array:

```
byte[] data;
```

How to print its content?

Turning Strings into byte-arrays (using default encoding).

```
byte[] data = "Software Systemen".getBytes();
```

As a list of numbers?

23, 24, 37, 64, 62

ASCII control characters		
00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	HT	(Horizontal Tab)
10	LF	(Line feed)
11	VT	(Vertical Tab)
12	FF	(Form feed)
13	CR	(Carriage return)
14	SO	(Shift Out)
15	SI	(Shift In)
16	DLE	(Data link escape)
17	DC1	(Device control 1)
18	DC2	(Device control 2)
19	DC3	(Device control 3)
20	DC4	(Device control 4)
21	NAK	(Negative acknowl.)
22	SYN	(Synchronous idle)
23	ETB	(End of trans. block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)
30	RS	(Record separator)
31	US	(Unit separator)
127	DEL	(Delete)

ASCII printable characters					
32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	\$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(72	H	104	h
41)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[123	{
60	<	92	\	124	
61	=	93]	125	}
62	>	94	^	126	~
63	?	95	_		

REPRESENTING NUMBERS (RECAP MODULE 1)

$$0 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

01101101₂

binary
(base 2)

$$1 \cdot 8^2 + 5 \cdot 8^1 + 5 \cdot 8^0$$

155₈

octal
(base 8)

$$1 \cdot 10^2 +$$

decimal
(base 10)

$$0_{10} = 0_{16}$$

$$1_{10} = 1_{16}$$

$$2_{10} = 2_{16}$$

$$3_{10} = 3_{16}$$

$$4_{10} = 4_{16}$$

$$5_{10} = 5_{16}$$

$$6_{10} = 6_{16}$$

$$7_{10} = 7_{16}$$

$$8_{10} = 8_{16}$$

$$9_{10} = 9_{16}$$

$$10_{10} = A_{16}$$

$$11_{10} = B_{16}$$

$$12_{10} = C_{16}$$

$$13_{10} = D_{16}$$

$$14_{10} = E_{16}$$

$$15_{10} = F_{16}$$

$$6 \cdot 16^1 + 13 \cdot 16^0$$

6D₁₆

hexadecimal
(base 16)

ASCII/UTF-8 representation: 'm'

REPRESENTING BINARY DATA (BYTE ARRAYS)

A byte-array:

```
byte[] data;
```

How to print its content?

Hexadecimal!

536f6674776172652053797374656d656e

1st byte (data[0]) 2nd byte (data[1])

Turning Strings into byte-arrays (using default encoding).

```
byte[] data = "Software Systemen".getBytes();
```

ASCII control characters		
00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	HT	(Horizontal Tab)
10	LF	(Line feed)
11	VT	(Vertical Tab)
12	FF	(Form feed)
13	CR	(Carriage return)
14	SO	(Shift Out)
15	SI	(Shift In)
16	DLE	(Data link escape)
17	DC1	(Device control 1)
18	DC2	(Device control 2)
19	DC3	(Device control 3)
20	DC4	(Device control 4)
21	NAK	(Negative acknowl.)
22	SYN	(Synchronous idle)
23	ETB	(End of trans. block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)
30	RS	(Record separator)
31	US	(Unit separator)
127	DEL	(Delete)

ASCII printable characters					
32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	\$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(72	H	104	h
41)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[123	{
60	<	92	\	124	
61	=	93]	125	}
62	>	94	^	126	~
63	?	95	_		

Use more printable characters: base64

ASCII control characters			ASCII printable characters					
00	NULL	(Null character)	32	space	64	@	96	`
01	SOH	(Start of Header)	33	!	65	A	97	a
02	STX	(Start of Text)	34	"	66	B	98	b
03	ETX	(End of Text)	35	#	67	C	99	c
04	EOT	(End of Trans.)	36	\$	68	D	100	d
05	ENQ	(Enquiry)	37	%	69	E	101	e
06	ACK	(Acknowledgement)	38	&	70	F	102	f
07	BEL	(Bell)	39	'	71	G	103	g
08	BS	(Backspace)	40	(72	H	104	h
09	HT	(Horizontal Tab)	41)	73	I	105	i
10	LF	(Line feed)	42	*	74	J	106	j
11	VT	(Vertical Tab)	43	+	75	K	107	k
12	FF	(Form feed)	44	,	76	L	108	l
13	CR	(Carriage return)	45	-	77	M	109	m
14	SO	(Shift Out)	46	.	78	N	110	n
15	SI	(Shift In)	47	/	79	O	111	o
16	DLE	(Data link escape)	48	0	80	P	112	p
17	DC1	(Device control 1)	49	1	81	Q	113	q
18	DC2	(Device control 2)	50	2	82	R	114	r
19	DC3	(Device control 3)	51	3	83	S	115	s
20	DC4	(Device control 4)	52	4	84	T	116	t
21	NAK	(Negative acknowl.)	53	5	85	U	117	u
22	SYN	(Synchronous idle)	54	6	86	V	118	v
23	ETB	(End of trans. block)	55	7	87	W	119	w
24	CAN	(Cancel)	56	8	88	X	120	x
25	EM	(End of medium)	57	9	89	Y	121	y
26	SUB	(Substitute)	58	:	90	Z	122	z
27	ESC	(Escape)	59	;	91	[123	{
28	FS	(File separator)	60	<	92	\	124	
29	GS	(Group separator)	61	=	93]	125	}
30	RS	(Record separator)	62	>	94	^	126	~
31	US	(Unit separator)	63	?	95	_		
127	DEL	(Delete)						

A MORE (SPACE) EFFICIENT BINARY-TO-TEXT ENCODING: BASE64

6 bits per character: $2^6 = 64$ different characters (26 + 26 + 10 + 2)

Value	Char	Value	Char	Value	Char	Value	Char
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

Text content	M	a	n	
ASCII	77 (0x4d)	97 (0x61)	110 (0x6e)	
Bit pattern	0100110101011000010110110			
Index	19	22	5	46
Base64-encoded	T	W	F	u

Note: padding is needed!

BASE64 EXAMPLES

```
byte[] data = "Software Systemen".getBytes();
```

Base64 → U29mdHdhcmUgU3lzdGVtZW4=

```
byte[] data = "Software Systemen 2018".getBytes();
```

Base64 → U29mdHdhcmUgU3lzdGVtZW4gMjAxOA==

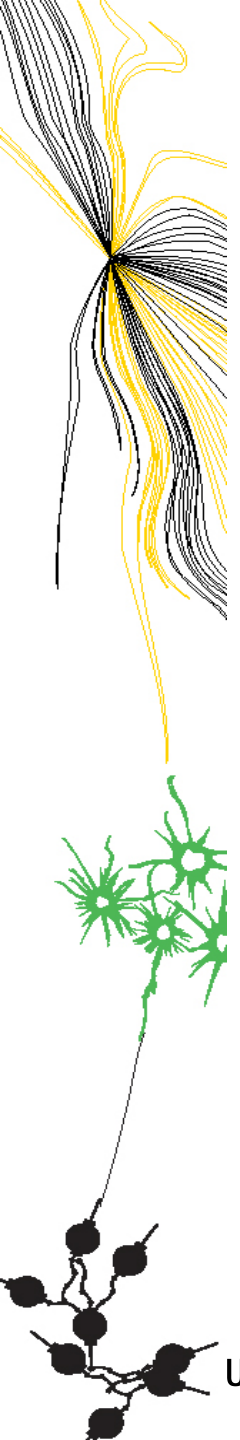
```
byte[] data = " Software Systemen 2018".getBytes();
```

Base64 → IFNvZnR3YXJlIFN5c3RlbWVuIDIwMTg=

padding

NOTES ON ENCODING

- Use libraries for this! (e.g., Apache Commons Codec library)
- Superfluous remark: encoding is not encryption!

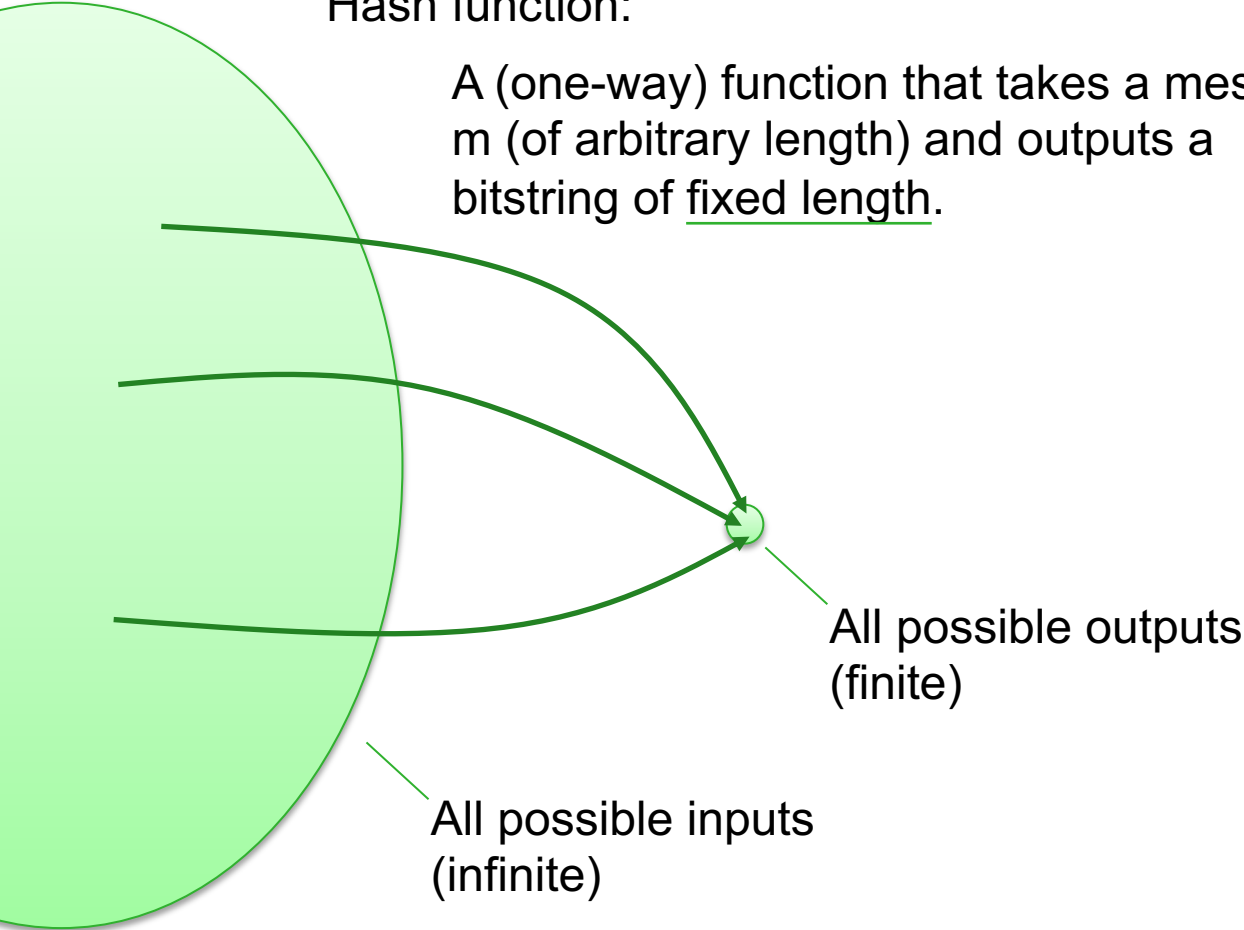


CRYPTOGRAPHIC HASH FUNCTIONS

CRYPTOGRAPHIC HASH FUNCTIONS

Hash function:

A (one-way) function that takes a message m (of arbitrary length) and outputs a bitstring of fixed length.



HASH FUNCTION PROPERTIES

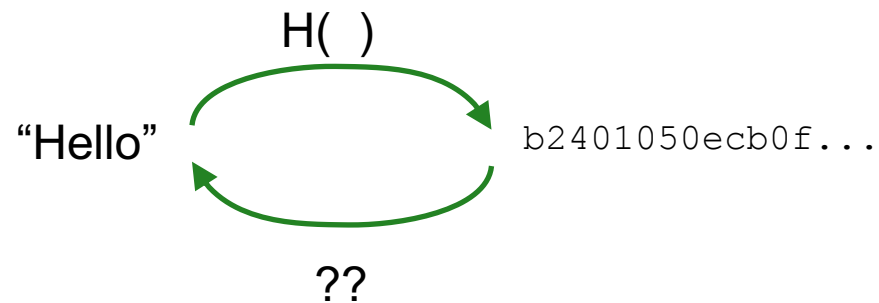
A “good” hash has the following properties:

1. One-way
2. Second preimage resistant
3. Collision resistant

[Based on slides by Bart Jacobs]

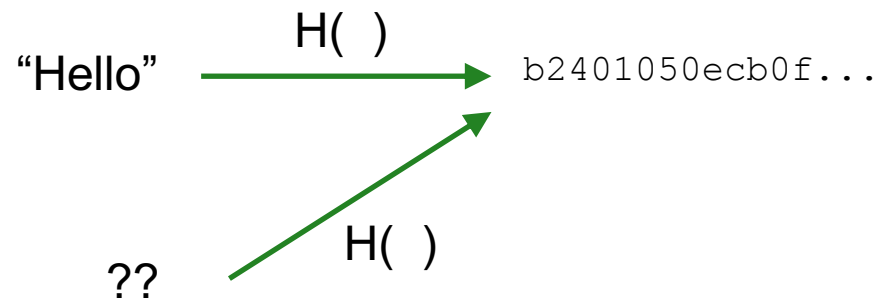
1: ONE-WAY

It is difficult (computationally infeasible) to *invert the hash function*.



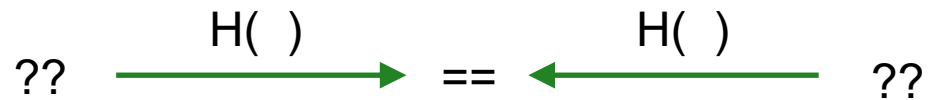
2: SECOND PREIMAGE RESISTANT

It is difficult (computationally infeasible) to
find a second input that hashes a given value.



3: COLLISION RESISTANT

It is difficult (computationally infeasible) to *find two inputs with the same hash value.*



HASH FUNCTION EXAMPLES

hexadecimal representation of the output (message digest)

SHA1("Security in 2019") → 63fc6185233b7d3de05540aea70cc49b702d3601

SHA1("Security in 2020") → 197ba5a6d6070e299a872e4ec05826fd4ff5c169

small change (a few bits)

very different output!

SHA1("Security in 2020 is still important!")

→ 3f9dc2a880b85aeb8b0b0e5b6825c0b499fc6c5e

length remains the same

HASH FUNCTIONS

MD5



Broken **DO NOT USE!**
(collisions found)

SHA-1



Theoretical attacks & more (avoid if possible,
will soon be deprecated)

SHA-2 family
(e.g., SHA-256)



Still safe (for now), your best bet

SHA-3



Recent standard, somewhat 'unproven'

USES OF HASH FUNCTIONS

Digital signatures  See module 1

HMAC  Coming up

Integrity checks

Content Addressable Storage  Version control, e.g., git
 Magnet links:

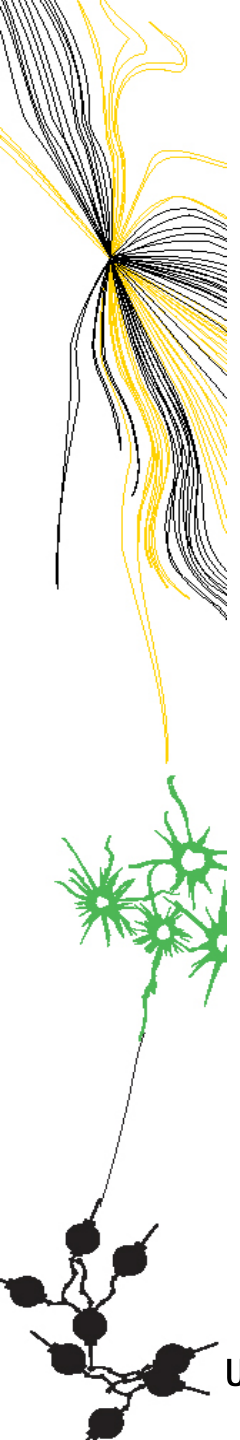
<magnet:?xt=urn:sha1:YNCKHTQCWBTRNJIV4WNAE52SJUQCZO5C>

Storing passwords  See assignments P-6.24-30

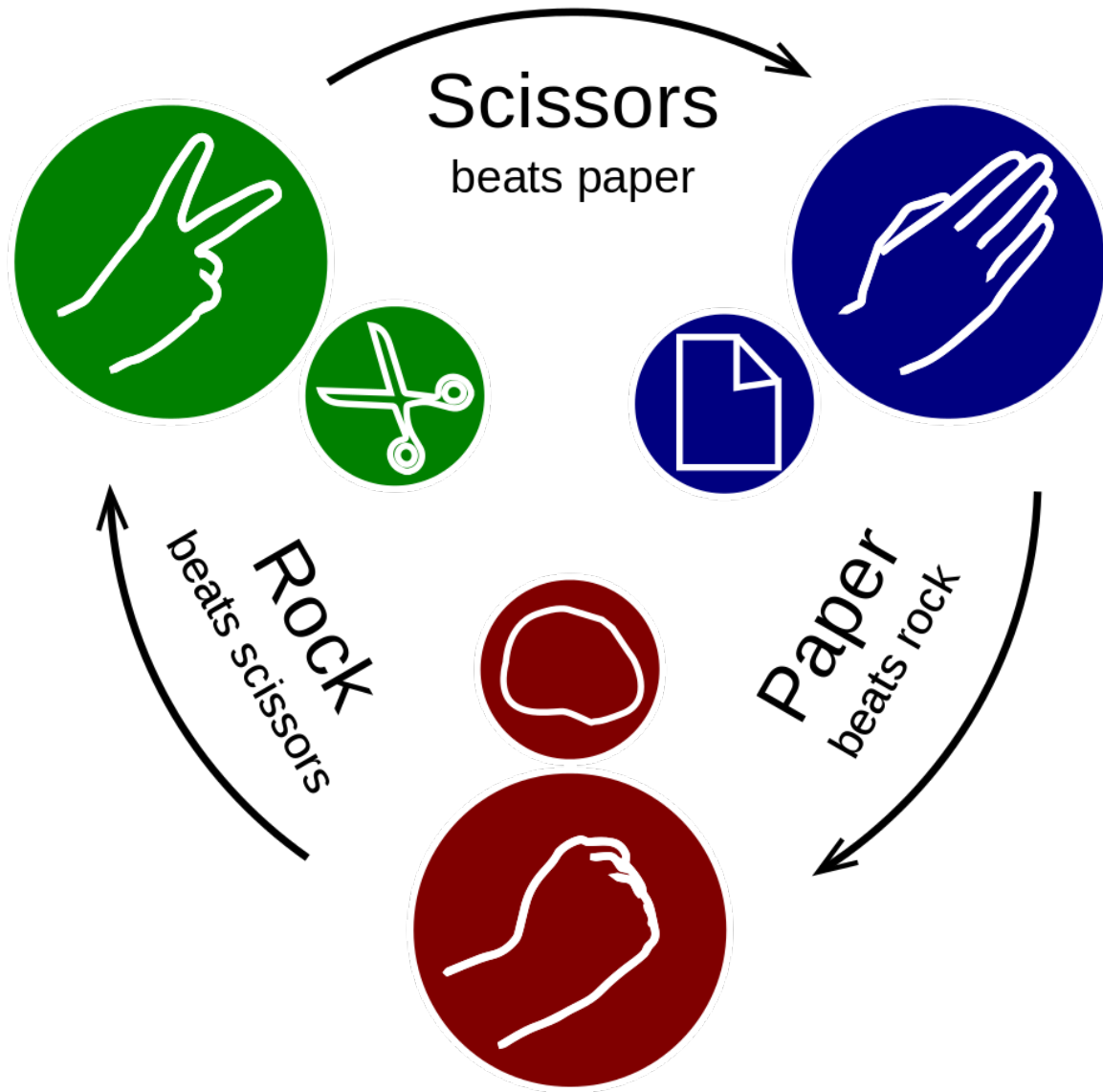
Commitments  Let's play with that!

HASH FUNCTIONS IN JAVA

```
byte[] inputData = "The data to hash".getBytes();
MessageDigest md = null;
md = MessageDigest.getInstance("SHA-256");
md.update(inputData);
byte[] digest = md.digest();
```



COMMITMENTS

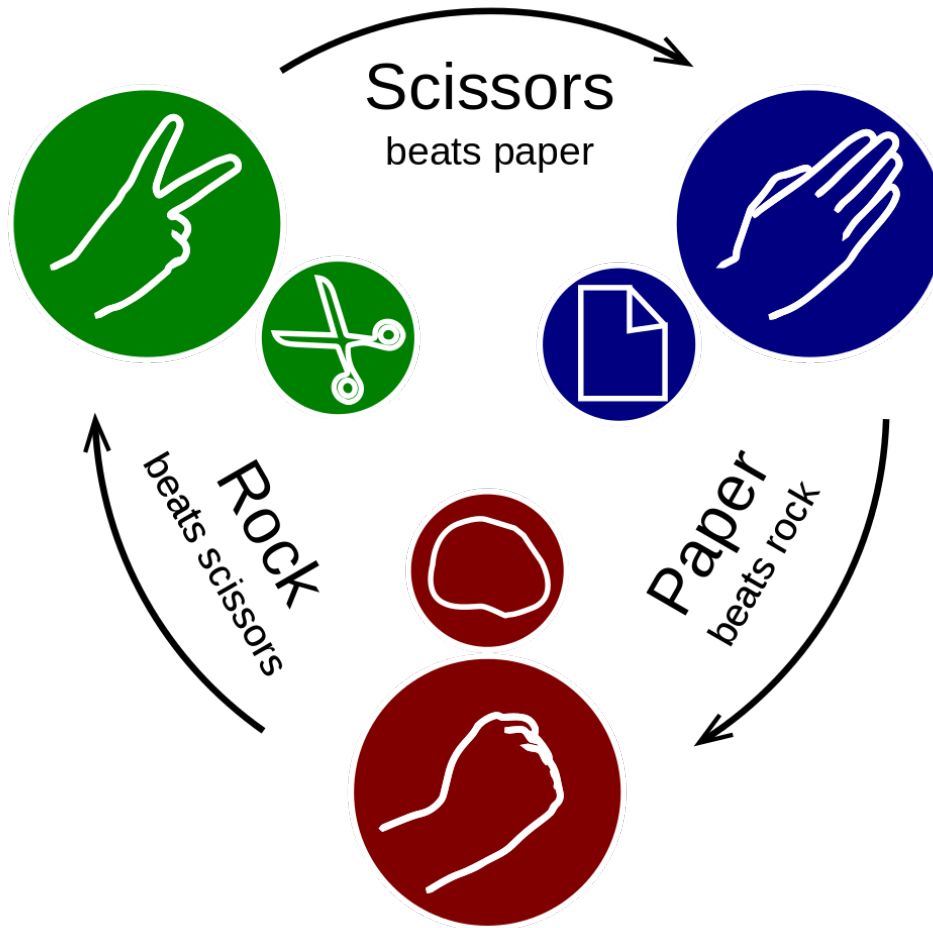






SHELDON COOPER

"Scissors cuts paper, paper covers rock, rock crushes lizard, lizard poisons Spock. Spock smashes scissors, scissors decapitates lizard, lizard eats paper, paper disproves Spock, Spock vaporizes rock, and as always has it; rock crushes scissors."



How to play rock-paper-scissors over the phone?

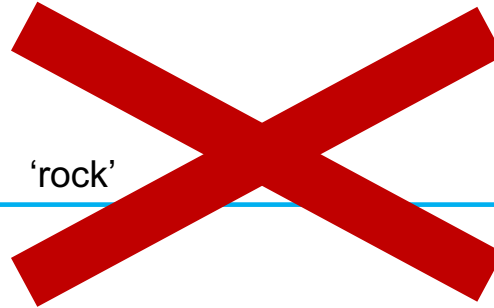
Let's try

Attempt 1:

Alice



'rock'



Bob

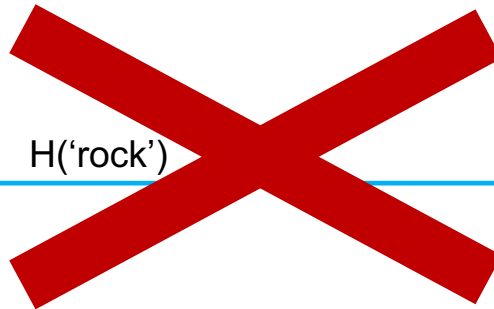


Attempt 2:

Alice




H('rock')



Bob



- == H('rock')? 
- == H('paper')?
- == H('scissors')?

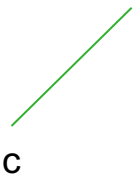
For example: 83751e164d08f068c33ca43d2cf0d9198b2432d4

Alice



Choose value (e.g, 'rock')
r = random bitstring
c = H(r + 'rock')

commitment



c



Bob



Hash function property:
one-way
Bob cannot invert c to determine
a value that always makes him
win.

Choose value (e.g., 'paper')



'paper'

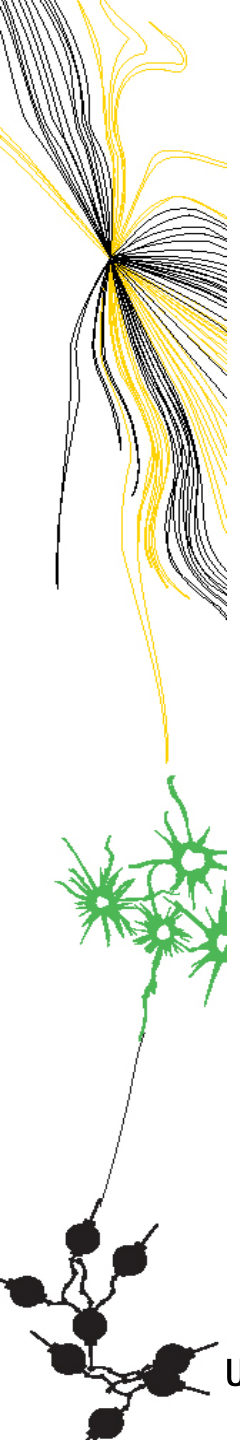
Knows whether lost or won.

Hash-function property:
second preimage resistant
Alice cannot find another input value
that also hashes to the same c.

r, 'rock'



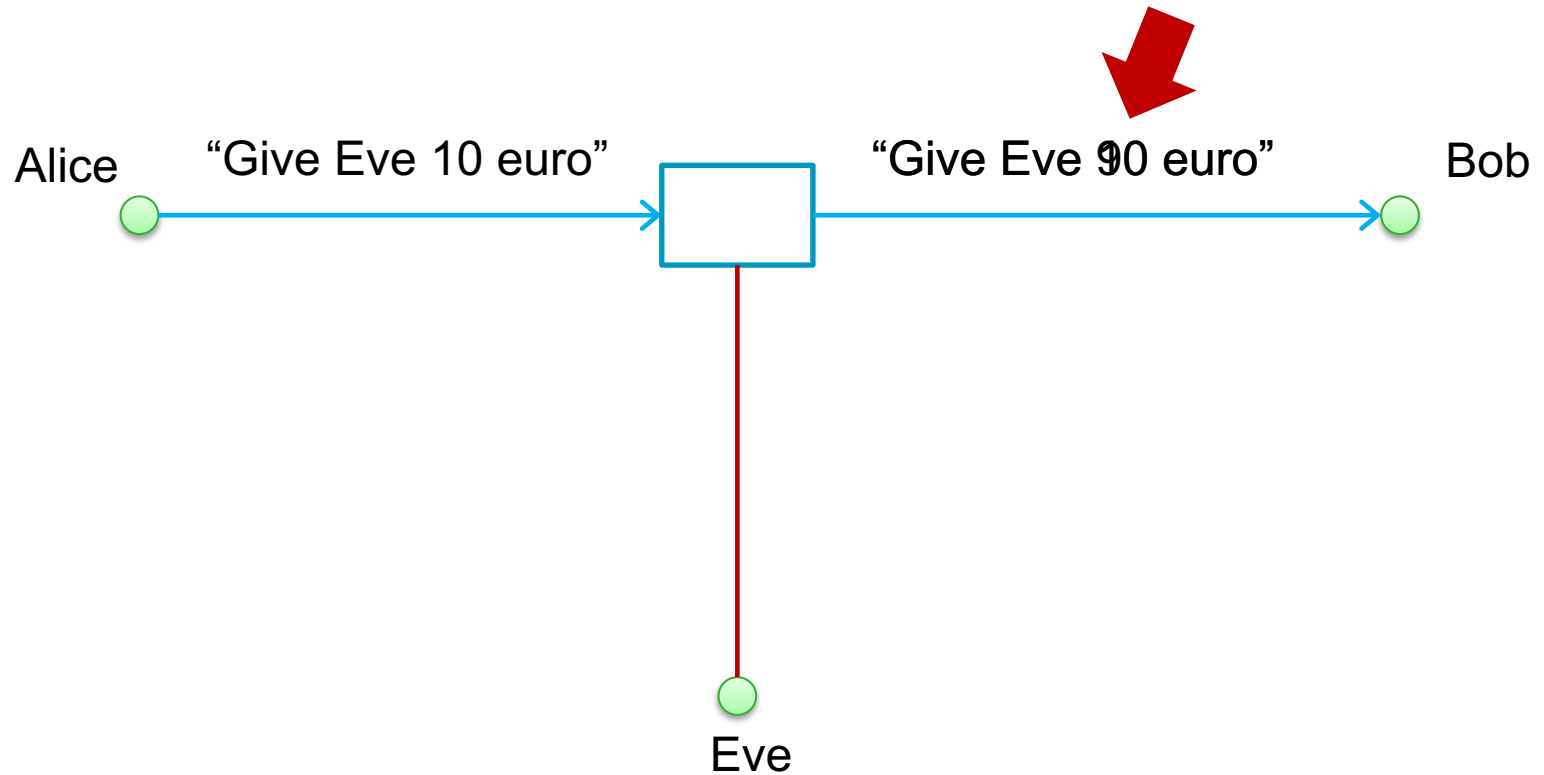
c' = H(r + 'rock')
if c == c': Alice did not cheat



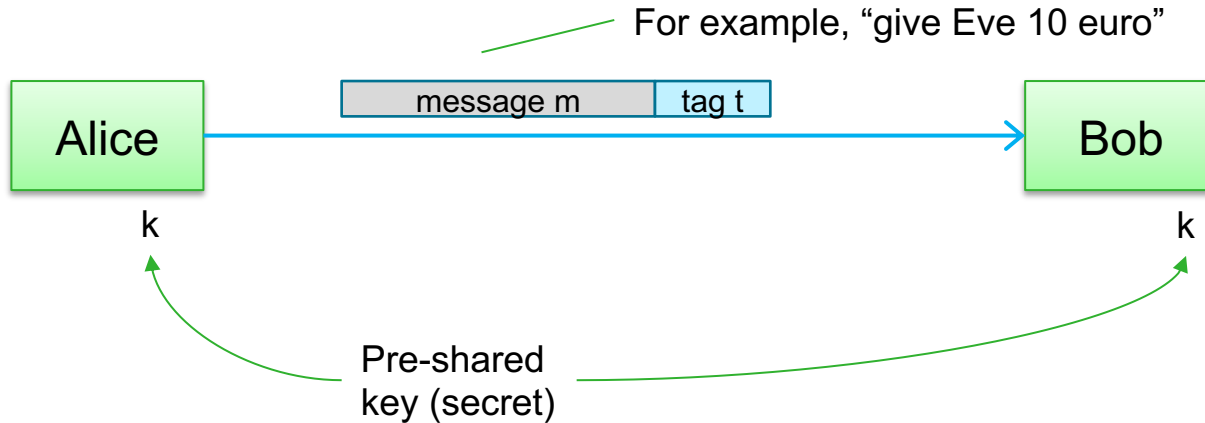
HMAC

UNIVERSITY OF TWENTE.

INTEGRITY EXAMPLE (RECAP)



MESSAGE AUTHENTICATION CODES (MACS)



$$t \leftarrow S(k, m)$$

$$V(k, m, t) \stackrel{?}{=} \text{'yes'}$$

- ➡ Attacker cannot produce a valid tag for a *new* message
 - ➡ Given (m, t) , attacker cannot produce (m, t') for $t' \neq t$
- } Ensuring integrity

[Based on slides by Dan Boneh]

MAC FROM HASH FUNCTION (NAIVE FIRST ATTEMPT)

Hash function Concatenation symbol

$$S(k, m) = H(k \parallel m)$$

Without k , it should not be possible to produce a valid tag for another message, right? Or is it?

Why is this not secure?

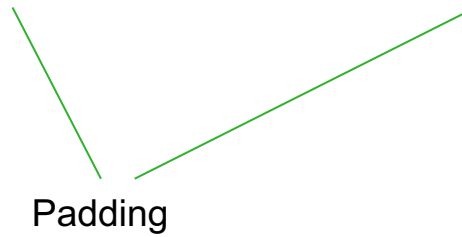
Given $H(k \parallel m)$, an attacker can compute $H(k \parallel m \parallel \text{PB} \parallel w)$ for any w .

Padding bytes

“give Eve 10 euro” \parallel t  “give Eve 10 euro!@. and Eve 90 euro” \parallel t’

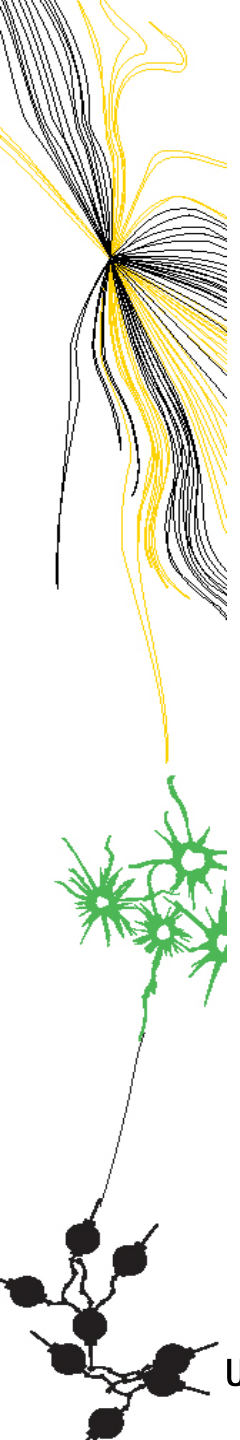
HMAC: STANDARDIZED MAC FROM HASH

$$S(k, m) = H(k \oplus \text{opad} \parallel H(k \oplus \text{ipad} \parallel m))$$



HMAC IN JAVA

```
String HMAC_ALGORITHM = "HmacSHA1";
Mac mac = Mac.getInstance(HMAC_ALGORITHM);
byte[] keyBytes = "HelloWorld".getBytes();
SecretKeySpec signingKey = new SecretKeySpec(keyBytes, HMAC_ALGORITHM);
mac.init(signingKey);
byte[] messageMac = mac.doFinal("Hello World, a signed message.".getBytes());
```



SIDE-CHANNEL ATTACKS

SIDE-CHANNEL ATTACKS

Whenever your program deals with secret information, it may leak information about them:

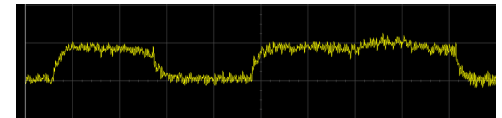
Timings



Power usage




Cache misses



One of the reasons to use proper libraries!

SIDE-CHANNEL ATTACK EXAMPLE

Suppose:

- A simple networked service
- Password-protected
- Password is sent in plaintext  For simplicity, typically a bad idea
- Password: exactly 8 characters (a-z, A-Z, 0-9)

```
public boolean checkPassword(String input)
```

Number of different characters: $26+26+10 = 62$

Number of possible passwords: $62^8 = 218340105584896 (\approx 2 \cdot 10^{14})$

SIDE-CHANNEL ATTACK EXAMPLE

```
private String secretPassword = "Secret12";
public boolean checkPassword(String input) {
    if (input.length() != secretPassword.length()) {
        return false;
    }
    for (int i = 0; i < input.length(); i++) {
        if (input.charAt(i) != secretPassword.charAt(i)) {
            return false;
        }
    }
    return true;
}
```

Suppose: 1 iteration ~10 μ s

Given timing information:

input = "AAAAAAAA" vs. input = "SecAAAAA"?

input = "AAAAAAAA" vs. input = "BAAAAAAAA"?

input = "AAAAAAAA" vs. input = "SAAAAAAAA"?

input = "SAAAAAAAA" vs. input = "SeAAAAAA"?

input = "SeAAAAAA" vs. input = "SecAAAAA"?



Algorithm for side-channel attack!

$8 \cdot 62 = 496$ attempts

vs.

$62^8 = 218340105584896$ attempts

Meltdown and Spectre

Vulnerabilities in modern computers leak passwords and sensitive data.

Meltdown and Spectre exploit critical vulnerabilities in modern processors. These hardware vulnerabilities allow programs to steal data which is currently processed on the computer. While programs are typically not permitted to read data from other programs, a malicious program can exploit Meltdown and Spectre to get hold of secrets stored in the memory of other running programs. This might include your passwords stored in a password manager or browser, your personal photos, emails, instant messages and even business-critical documents.

Meltdown and Spectre work on personal computers, mobile devices, and in the cloud. Depending on the cloud provider's infrastructure, it might be possible to steal data from other customers.



Meltdown

Meltdown breaks the most fundamental isolation between user applications and the operating system. This attack allows a program to access the memory, and thus also the secrets, of other programs and the operating system.

If your computer has a vulnerable processor and runs an unpatched operating system, it is not safe to work with sensitive information without the chance of leaking the information. This means that users should not use a computer with a vulnerable processor and an unpatched operating system.



Spectre

Spectre breaks the isolation between different applications. It allows an attacker to trick error-free programs, which follow best practices, into leaking their secrets. In fact, the safety checks of said best practices actually increase the attack surface and may make applications more susceptible to Spectre.

Spectre is harder to exploit than Meltdown, but it is also harder to mitigate. [However, it is possible to prevent it if you use a patched processor and Spectre mitigations.](#)

Security

Meltdown, Spectre bug patch slowdown gets real – and what you can do about it

Chip flaw fixes not so insignificant after all

By [Thomas Claburn](#) in [San Francisco](#) 9 Jan 2018 at 00:45 129 [SHARE](#) ▼



Analysis Having shot itself in the foot by prioritizing processor speed over security, the chip industry's fix involves doing the same to customers.

The patches being put in place to address the Meltdown and Spectre bugs that affect most modern CPUs were supposed to be airy little things of no consequence. Instead, for some unlucky people, they're anchors.

Having helped find the flaws, Google [insisted](#) the software fixes that have begun to appear "introduce minimal performance impact," and insisted the performance hit will diminish over time.

Intel said as much in its statement, claiming "any performance impacts are workload-dependent, and, for the average computer user, should not be significant and will be mitigated over time."

Most read



Ticketmaster tells customer it's not at fault for site's Magecart malware pwnage



Windows 10 can carry on slurping even when you're sure you yelled STOP!



Having swallowed its pride and started again with 10nm chips, Intel teases features in these 2019-ish processors



Here's 2018 in a nutshell for you... Russian super robot turns out to be man in robot suit



It is with a heavy heart that we must inform you hackers are targeting 'nuclear, defense, energy, financial' biz

USING CRYPTOGRAPHY – WORDS OF WARNING

DO NOT invent your own crypto!

It will be broken!

Only do it for fun (and learning)!

DO NOT even implement cryptographic primitives yourself!

So many details to get right

“It is typically not the math that is broken, it’s the implementation”

Rules of thumb

Data at rest: use pgp/gpg

Data in motion: use (SSL/)TLS

But be careful!

USE *high-level* cryptographic libraries!

NaCl (“salt”): <http://nacl.cr.yp.to/>

Keyczar: <http://www.keyczar.org/>

Preferably open source!

WHY?

“We’ll just use encryption, AES appears to be good, yes?”



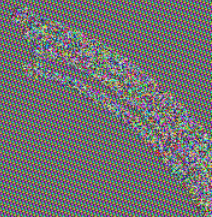
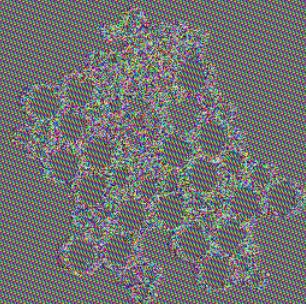
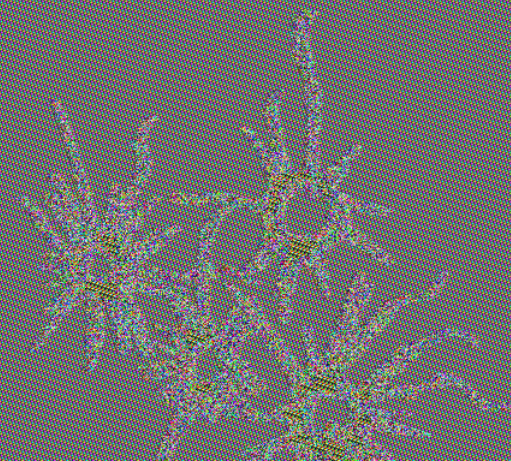
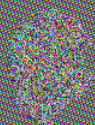
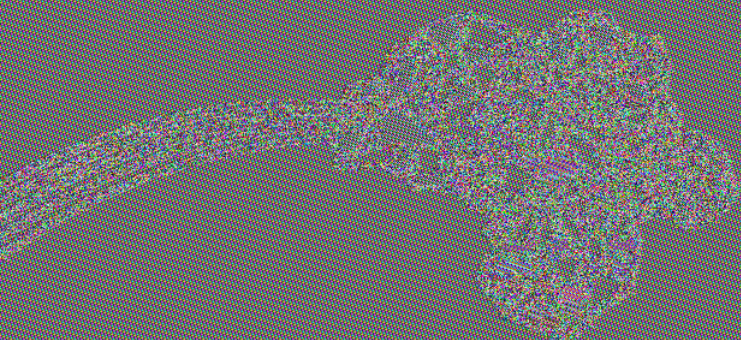
AES in the ECB mode-of-operation
applied to a image

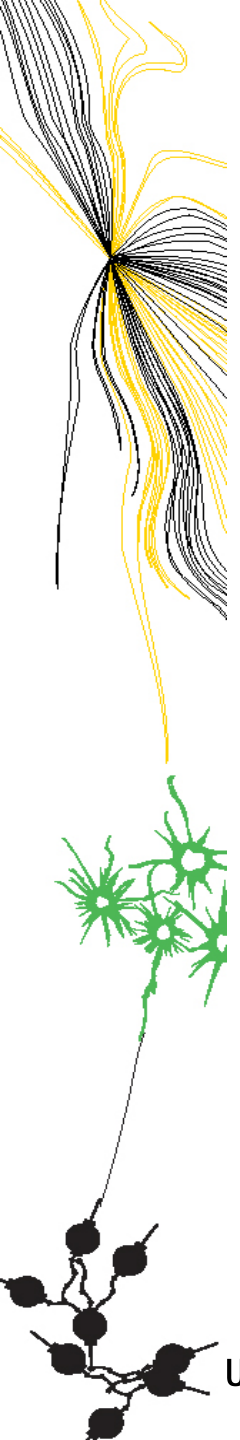
1999-2000

1999-2000

1999-2000

1999-2000





WANT MORE SECURITY?

UNIVERSITY OF TWENTE.

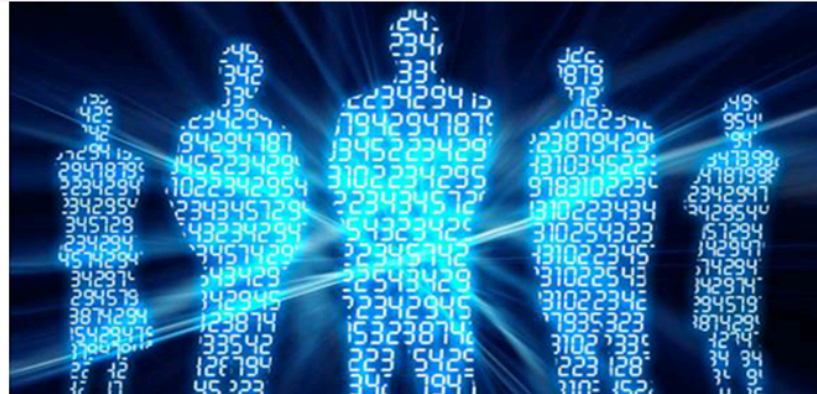
SECURITY IN OTHER MODULES





Cyber Security

The 4TU cyber security master specialisation offers computer science master students a state-of-the-art education and an opportunity to contribute to our cutting-edge research.



Latest updates

Introduction

Importancy of cyber security

Our society critically depends on cyber space for almost everything, including banking, transport & logistics, air travel, energy, telecommunications, flood defences, health care, email, social networks, and even warfare. The consequences of cyber security failures could be disastrous and the demand for cyber security specialists is therefore high and rising. The 4TU cyber security master...

Programme

Course programme

Our programme consists of several courses, an off-site summer school, a choice of electives and an individual final year project. Read more about our extensive course programme.

News

4TU.CybSec New Student Assistant

We thank Lisa de Wilde for her efforts to liaise between the students and the staff of the 4TU.CybSec master specialisation. We wish Lisa much success with her master thesis.

Thursday 1 September 2016

The new 4TU cyber security master specialisation:
<https://www.4tu.nl/cybsec/en/>

Course Overview (11 min) Help Center ✕

Online Cryptography Course Dan Boneh



Introduction

Course Overview



00:02 / 10:34 CC 🔊 🗑

⏪ 1x ⏩ « Prev Next » ⓘ ⚙

MOOC: Coursera's course on Cryptography by Dan Boneh (Stanford)
<https://www.coursera.org/course/crypto>



What is computer security?

- Most developers and operators are concerned with **correctness**: achieving desired behavior
 - A working banking web site, word processor, blog, ...
- Security is concerned with **preventing undesired behavior**
 - Considers an enemy/opponent/hacker/adversary who is *actively and maliciously* trying to *circumvent* any protective measures you put in place



MOOC: Coursera's course on Software Security by Michael Hicks

<https://www.coursera.org/course/softwaresec>



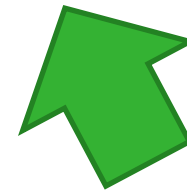


Twente Hacking Squad:

See <http://scs.ewi.utwente.nl/home/TwenteHackingSquad/> for more info.

LAB EXERCISES RELATED TO SECURITY

- P-5.7-9 (Hex & Base64 encoding)
- P-6.4-10 (Password cracking & how to properly store them)
- P-7.6 (Networked attack)
- P-7.17 (Bonus: crypto attack)



Also read the text
surrounding the
exercise!