ARCHIVIO ISTITUZIONALE
DELLA RICERCA

Alma Mater Studiorum Università di Bologna
Archivio istituzionale della ricerca

Cryptography in Grade 10: Core Ideas with Snap! and Unplugged

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

Published Version:
Cryptography in Grade 10: Core Ideas with Snap! and Unplugged / Lodi, Michael; Sbaraglia, Marco; Martini, Simone. - ELETTRONICO. - 1:(2022), pp. 456-462. (Intervento presentato al convegno 27th ACM Conference on on Innovation and Technology in Computer Science Education (ITiCSE '22) tenutosi a Dublin nel 11-13 July, 2022) [10.1145/3502718.3524767].

Availability:
This version is available at: https://hdl.handle.net/11585/890499 since: 2022-07-12

Published:

DOI: http://doi.org/10.1145/3502718.3524767

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/).
When citing, please refer to the published version.

(Article begins on next page)

26 April 2024

This is the final peer-reviewed accepted manuscript of:

Michael Lodi, Marco Sbaraglia, and Simone Martini. 2022. Cryptography in Grade 10: Core Ideas with Snap! and Unplugged. In Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1 (ITiCSE '22). Association for Computing Machinery, New York, NY, USA, 456–462.

The final published version is available online at: https://doi.org/10.1145/3502718.3524767

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/)

When citing, please refer to the published version.

Cryptography in Grade 10: Core Ideas with Snap! and Unplugged

Michael Lodi*
Università di Bologna
Bologna, Italy
INRIA Sophia-Antipolis
Valbonne, France

Marco Sbaraglia*
Università di Bologna
Bologna, Italy

Simone Martini*
Università di Bologna
Bologna, Italy
INRIA Sophia-Antipolis
Valbonne, France

ABSTRACT

We report our experience of an extracurricular online intervention on cryptography in Grade 10. Our first goal is to describe how we taught some fundamental cryptography ideas by making students encounter a progression of representative cryptosystems, from classical to modern, and discover their characteristics and limitations. We used Snap! (a visual programming language) to realize hands-on activities: block-programming playgrounds (a form of task-specific programming languages) to experiment with cryptosystems, and an interactive app to support an unplugged (albeit remote) Diffie-Hellman key agreement. After experimenting with each system, the students were involved in a Socratic discussion on how to overcome the discovered limitations, motivating the introduction of the following system in our path. Our second goal is to evaluate the students' perceptions and learning of cryptography core ideas. They appreciated the course and felt that, despite being remote, it was fun and engaging. According to the students, the course helped them understand the role of cryptography, CS, and Math in society and sparked their interest in cryptography and CS. The final assessment showed that the students well understood the cryptography ideas addressed. Our third goal is to discuss what worked and areas of improvement. The "remote-unplugged" Diffie-Hellman, where the meeting chat was a metaphor for the public channel, engaged the students in understanding this groundbreaking protocol. Overall, they praised the activities as engaging, even when challenging. However, a strong "instructor blindness" induced by remote teaching often prevented us from giving the students the right amount of guidance during the exploration activities.

CCS CONCEPTS

• **Security and privacy** → **Cryptography**; *Social aspects of security and privacy*; • **Social and professional topics** → **K-12 education**; • **Applied computing** → *Distance learning*.

KEYWORDS

cryptography education, cryptography playgrounds, task-specific programming, Snap!, Diffie-Hellman, remote unplugged

ACM Reference Format:
Michael Lodi, Marco Sbaraglia, and Simone Martini. 2022. Cryptography in Grade 10: Core Ideas with Snap! and Unplugged. In *Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol 1 (ITiCSE 2022), July 8–13, 2022, Dublin, Ireland*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3502718.3524767

*Also Laboratorio CINI "Informatica e Scuola"

This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *27th ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE 2022). ITiCSE 2022, July 8–13, 2022, Dublin, Ireland*
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9201-3/22/07…$15.00
https://doi.org/10.1145/3502718.3524767

1 INTRODUCTION

Cryptography is at the core of today's digital society's many activities and tools (e.g., instant messaging, e-commerce, stock exchange). Various frameworks (e.g., DigComp [7]) and curricula (e.g., CSTA K–12 CS Standards [18] and the UK computing curriculum [10]) include competencies related to cybersecurity. Some of them are more oriented on using security for personal purposes, others on understanding how digital security works, but they all recognize that cybersecurity skills are essential for students to be active citizens of digital society. Cryptography is one of the foundations of cybersecurity. In addition, novices identified cryptography "as an interesting context for computer science lessons" [23, p. 3]. K-12 education does not aim to train professionals but to help students understand our world and act in it, therefore it is important to help students understand the principles of cryptography and their importance in our society. With this aim, we designed a short course with no prerequisites, build around different types of activities. Since educational research has shown the effectiveness of active and cooperative learning methodologies [25, p. 304], we designed non-traditional hands-on activities to be interactive and meaningful for students. We developed cryptography playgrounds for students to use, understand, and attack emblematic cryptosystems (e.g., Caesar cipher, One-time pad) and a "remote-unplugged" activity to perform the Diffie-Hellman (DH) key agreement in pairs. We realized both types of activities with Snap!, a visual block-based programming language. Due to the ongoing COVID-19 pandemic, we had to design the intervention as remote-only. An English version of all the material we developed is available [24], under an open license.

Our pathway includes a few emblematic cryptographic systems and schemes, carefully selected as representatives of cryptography core ideas. To create a motivating progression, the introduction of a new scheme is always triggered by the *necessity* (which we stimulate in students [32]) to overcome the limitations of the previous one(s).

This paper's first goal is to present our learning path (section 2)—focusing on the cryptography principles and schemes that drive the learning progression (2.2)—and the development and testing of Snap! activities (i.e., crypto-playgrounds and unplugged DH agreement, 2.3). The second goal is to evaluate our experience (sections 4 and 5) regarding both the level of understanding of the fundamental concepts covered (4.1, 5.1) and students' satisfaction and perceived utility (4.2). The last goal is to discuss what worked and what can be improved (section 5.1) and help CS educators adopt and adapt our pathway and hands-on activities (5.2).

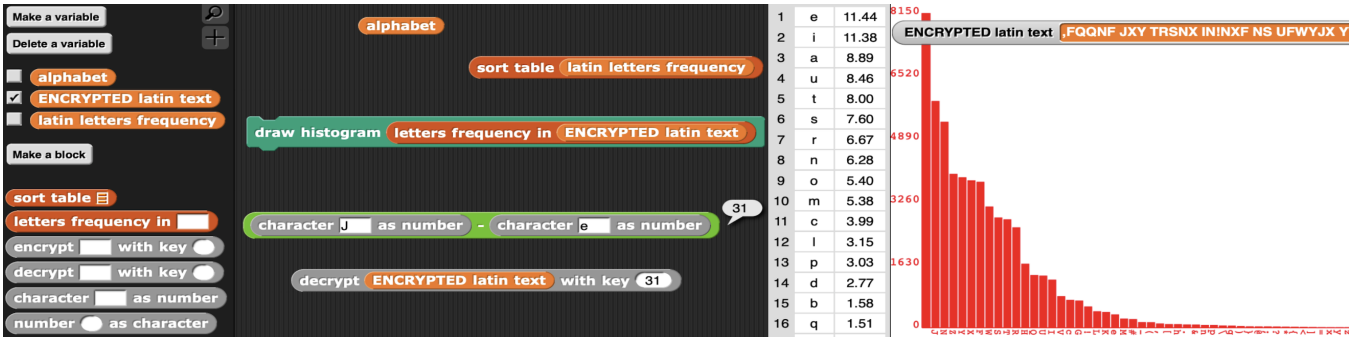

Figure 1: Attacking Caesar cipher with frequencies - Snap! playground

2 OUR COURSE

2.1 Context

Our intervention took place in a *Lyceum*, a strand of Italian high school[1] that gives a theoretical basis in classical, scientific, or artistic areas and naturally leads to university studies. Our intervention took place in the context of an extracurricular experimental project called *Mathematical Lyceum* which aims not to introduce new notions but to reflect on ideas and foundations of knowledge and broaden cultural horizons with a robust interdisciplinary approach [22]. We delivered a cryptography course as one of these extracurricular activities in a local lyceum.

Fifteen students (5 girls and 10 boys), all in their second year of lyceum (ca. 15-16 y.o.), attended our course voluntarily. None of them had previous programming experience. The course was delivered by two of the authors of this paper, who are researchers in CS education and have experience in high school teaching. We held four lessons (2 hrs each) every Tuesday afternoon in February 2021. Due to the COVID-19 pandemic, the lessons were held online through the Google Meet platform adopted by the school.

2.2 Learning path

The intervention was designed to teach fundamental cryptography ideas by making students encounter some representative cryptosystems (from classical to modern) and experience their limitations (through hands-on activities), thus the necessity to overcome those limitations (towards more secure systems). Some relevant mathematical concepts underlying cryptography are also addressed (e.g., permutations, modular arithmetic).

2.2.1 Crypto principles and ideas through representative systems. Here it follows the progression of representative cryptosystems and schemes we used. For each one, we indicate why we choose it as a representative for its class; the teaching motivations; the limitations students experience that support the transition to the next system.

Caesar cipher

- *Representative for:* monoalphabetic substitution ciphers
- *Motivations:* basic example of sym-crypto; easy to show the typical cryptosystem elements (plaintext, ciphertext, encrypt/decrypt functions, key) and simple attacks; easy to understand and play with it

↓ *Problems to overcome:* attackable with both brute-force and frequency analysis

One-time pad cipher

- *Representative for:* polyalphabetic ciphers (taken to the extreme); perfect secrecy; resistant to both brute-force and frequency attacks (no clues about the key or the plaintext from the ciphertext)
- *Motivations:* easy as "a different Caesar for each letter"

↓ *Problems to overcome:* key-distribution problem; feasibility issues (e.g., one-time, random keys)

Simple Substitution-Permutation network

- *Representative for:* modern symmetric block cryptosystems (e.g., AES); confusion and diffusion (avalanche effect); efficient implementation; "only" computationally secure
- *Motivations:* introducing operations on bits; grasping how modern cryptosystems are implemented with computers

↓ *Problems to overcome:* the key-distribution problem

Diffie-Hellman key-agreement protocol

- *Representative for:* shared key generation protocols; groundbreaking solution to the key-distribution problem
- *Motivations:* understanding how the discrete logarithm (easy to calculate, hard to invert) allows generating a shared secret over a public (insecure) channel

↓ *Problems to overcome:* person-in-the-middle attack

Idea of public-key secrecy and authentication

- *Representative for:* asymmetric cryptosystems
- *Motivations:* grasping that the properties of certain math functions (e.g., prime factorization) can be used to achieve both secrecy and authentication

↓ *Problems to overcome:* computationally expensive

Idea of hybrid cryptosystems

- *Representative for:* today's complex cryptosystems
- *Motivations:* learning that the best of symmetric and asymmetric cryptosystems combine in today's practice; grasping how relevant modern services (e.g., e2e instant messaging) work

[1] See [4] for a summary of the Italian secondary school system.

- *Problems to overcome: not raised in the course*

2.2.2 Contents.

Day 1

- The social debate about encryption in digital communication
- Caesar cipher: encryption, decryption, brute-force attack
- *Homework: transposition vs. substitution, Kerckhoffs principle*

Day 2

- Caesar cipher: frequency attack
- One-time pad: encryption, decryption, frequency attack
- *Homework: encoding chars as bits, toy cryptosystem using XOR and bit permutation, hints at modern block ciphers (DES, AES)*

Day 3

- One-time pad: brute-force attack, perfect secrecy, limitations, key-distribution problem
- DH protocol: simulation with colors
- *Homework: quiz on One-time pad and DH protocol*

Day 4

- Math of DH protocol: modular arithmetic, exponential and its inverse, primes and coprimes (and hints at generators)
- DH protocol: an example with small numbers, computational security, person-in-the-middle attack
- Asymmetric cryptography: terminology, key pairs properties, non-technical schemes for authentication and secrecy, intuitive idea of *one-way function* (multiplying vs. factoring)
- Putting all together: how *intuitively* combine asymmetric and symmetric schemes for authentication and secrecy
- *Homework: satisfaction survey, fill-in-the-blanks assessment*

2.3 Tools, activities, and methodology

2.3.1 Tools. The host school uses *Google Workspace for Education*. Therefore, we used *Google Meet* for the meetings and *Classroom* to share announcements, learning materials, and homework. We used *Slides* to present contents and animations and *Docs* to share longer texts[2]. We used *Google Forms* as worksheets to guide the hands-on activities with questions, gather homework, and collect final assessment responses and feedback on the course from students.

Snap! [28] is a block-based programming language. It is a reimplementation of Scratch with many extra features making it suitable for a serious introduction to CS programming for high-school or even college students. We chose Snap! as it allows the creation of new blocks that can return values (i.e., new functions) and whose implementation (the function body) can be hidden from students.

2.3.2 Created tools and activities, methodologies. Using Snap!, we created a progression of playgrounds for the hands-on activities. These environments allow students to experiment with cryptosystems (e.g., Caesar cipher, One-time pad) and their limitations (e.g., ease or difficulty of decryption, computation time required). A playground is a Snap! project with a limited set of visible instructions. By leveraging Snap!'s ability to create custom blocks and hide existing ones, for each cryptosystem we provided only the blocks needed to encrypt and decrypt messages and carry out possible attacks. Our playgrounds can be seen as Teaspoon Languages [43], "task-specific languages [...that]: support learning tasks that teachers (typically non-CS teachers) want students to achieve; are programming languages, in that they specify computational processes for a computational agent to execute; and are learnable in less than 10 minutes, so that they can be learned and used in a one hour lesson" [15]. Since our students had no programming experience, it was not feasible—nor useful, given the high-level objectives related to the cryptographic core ideas—for them to program (the algorithms of) cryptographic systems. However, we wanted to give students the opportunity to build their knowledge by "concretely" manipulating computational objects [30] related to cryptography. Instead of putting the students in microworlds *à la Papert*, usually relying on general-purpose languages such as LOGO—which take time to learn, we developed more abstract and much simpler (thus easier to use) languages narrowed to our specific learning objectives [16]. For example, to attack a Caesar ciphertext using frequencies, some of the blocks in the playground were: calculating frequencies in a text, sorting a table (i.e., a matrix of letters and their frequencies) by frequency, representing a table with a histogram, the table of letters' average frequencies in Latin (fig. 1).

Our playgrounds are available for exploration and use [24]. The students, following our worksheets, engaged in challenges of encryption, decryption, and attack by combining Snap! blocks. Our goal was to make the students experience the cryptosystems and understand their operation and limitations. We contextualized the activities in scenarios meaningful for the students. For example, the activities on Caesar cipher involve a Latin[3] text (to be used for a test) that their quarantined teacher wants to communicate secretly to her substitute without her students being able to intercept it.

Since the students were complete novices in programming, we felt it was too great a challenge for them to program the DH key agreement. Therefore, we then designed an *unplugged activity* to simulate the generation of the shared key. The unplugged approach helps students understand important algorithms at a high level by having them perform these algorithms concretely through kinesthetic and fun activities [2].

We found at least two typical ways to introduce DH key agreement [e.g., 42]: color mixing (more evocative yet simplified) and performing the algorithm with small numbers (more accurate yet complex). We tried to combine these two to move gradually from the color metaphor to the mathematical operations. An executable-only Snap! project (i.e., not modifiable nor explorable in its code) served as an interactive app for "DH color mixing" guiding the students in enacting the protocol [24]. Color mixing is not done with classical additive or subtractive algorithms but is based on the actual DH calculations (initially hidden to the students) on the small numbers (from 0 to 99) that in Snap! represent colors. Using the Snap! project, students in pairs were able to generate a shared secret color. Communication within each pair (i.e., choosing an initial shared color and exchanging the calculated color) took place on the online meeting public chat, which effectively represents an insecure channel since everyone can "listen on" it (see fig. 2).

After the students experienced firsthand the high-level functioning of the DH protocol, we presented the essential mathematical tools for its operation (see 2.2.2). Then, we showed the protocol itself through an animation [24] that displays the correspondence

[2] E.g., homework readings from outreach materials, like Singh's "The Code Book" [35].

[3] All our lyceum students were studying Latin.

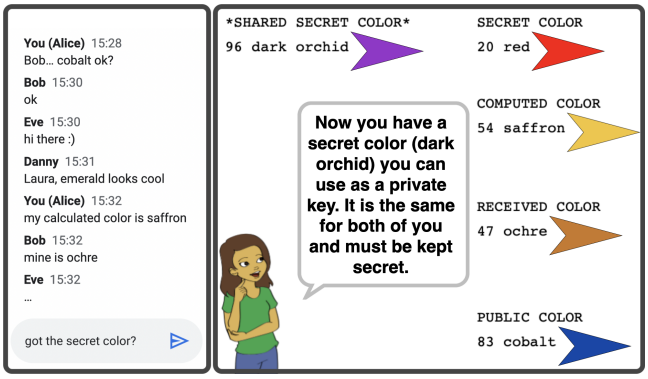

Figure 2: Meeting chat and support app for the DH activity

between the colors and the simple numbers representing them, revealing the actual calculations behind the color mixing.

Given the short duration of the course, for the more advanced schemes based on public-key cryptography, the objective was a high-level understanding of secrecy and authentication mechanisms and cognition of the reasons and scenarios for their use.

We illustrated to the students that it is possible to create two keys, one public and one private, bound by the property that what is locked with one can only be opened with the other. We suggested a mathematical intuition: the public key is related to the multiplication of two primes (easy), and the private key is related to the factoring of the product (difficult). We asked the students to imagine how to use the key pair to achieve secrecy. We also guided them toward the concept of authentication and its realization with an asymmetric scheme. We developed animations [24] of simple and typical communication scenarios; we also used Power Rangers characters so that the actions and messages of the different parties were evident through their respective colors. The various scenes of these animations supported the reasoning done with the students. Once all the communication steps were discussed, the animations let the students visualize the public-key schemes for authentication and secrecy in their entirety (albeit at a very high level).

3 RELATIONSHIP WITH PREVIOUS WORK

The "Cybersecurity Curricula 2017" [29] included "basic concepts in cryptography" (e.g., historical ciphers, modern block ciphers, and the DH key agreement) to be learned early "to build the base for other sections in the knowledge unit" [29, p. 24]. In the "CSTA K–12 CS Standards" [9], cybersecurity is important for all grades. In particular, for Level 2 (grades 6-8, 11-14 y.o.), the standard (2-NI-06) indicates that students should be able to encrypt and decrypt messages with various encryption methods and understand their different levels of complexity, starting with the simplest (e.g., Caesar cipher) to the more complex public-key ciphers, which are better learned through unplugged activities. Overall, our path is in line with these standards, both in terms of contents (adapted to our age target) and methodologies (e.g., unplugged).

A review of cybersecurity education papers over the last ten years of SIGCSE and ITiCSE conferences [41] found that research "predominantly focus[es] on tertiary education in the USA." Only a few works focus specifically on cryptography [e.g., 6, 17] since most of them are more broadly focused on cybersecurity (addressing crypto only as one of the topics) [e.g., 5, 11, 36, 39].

We conducted a similar search for works describing teaching cryptography (and cybersecurity, when cryptography was included) in K-12. The works we found usually include some hands-on crypto activities, often set in motivating real-world contexts (e.g., secure email exchange, communicating robots, a toy social network) [e.g., 13, 21, 33, 44]. Indeed, Konak [20] claimed that hands-on inquiry-based cryptography and cybersecurity activities improve K-12 students' self-efficacy and problem-solving skills. Our activities are contextualized in situations meaningful to students (e.g., "Can you obtain the exam text that your teachers secretly exchange?"), leveraging the dimensions of challenge and adventure that cryptography can offer [13, 23]. Also, our path follows the constant thread of a motivating and timely issue: "What do you need to know to understand how secrecy works in instant messaging? Is it really secret?".

Given the content found in literature and curricula, and considering our target audience and cultural perspective (i.e., K-12 education does not aim to train professionals but to give tools to understand the world, see sec. 1), we aim to make the fundamental principles of cryptography understandable by novices. Therefore, our path focuses on a few strategic cryptosystems, selected to convey cryptography core ideas, digging into technical details only when they are instrumental to understanding those ideas. This perspective informed our design and the development and use of the tools.

Visualization tools or interactive simulations for teaching cryptography have been developed [e.g., 1, 26, 34] to show how ciphers work, their weaknesses, and possible attacks. These tools are accurate but too detailed for young students. Compared to these proposals, which mostly provide ready-to-use software visualizations or simulations where students can only adjust parameters, in our path, students tinker by manipulating and combining objects that represent essential cryptography concepts (see 2.3.2). In our crypto playgrounds, students actively program some parts of their learning experience (e.g., combining blocks to calculate and visualize letter frequencies in order to perform a frequency attack—see fig. 1). The idea of using very high-level programming to better understand cryptosystems and attacks can be found in the course reported by McAndrew, where students implement cryptographic algorithms (from classical to modern) with computer algebra systems [27]. In addition, our approach (although targeted to K-12 students) also shares the vision of using the metaphor of visual programming to realize blocks that represent cryptographic functionalities [40]. Blocks are more abstract and, for novices, should be easier to understand and compose than source code.

Several proposals use unplugged activities for cryptography to get students to experiment with encryption/decryption algorithms, protocols, and attacks at a high level [e.g., 3, 12, 19]. These activities use simple objects (e.g., maps, scissors, boxes, padlocks) and concrete actions (e.g., cutting out a sheet, mixing colors). For example, Fees et al. made the DH color metaphor concrete by making students mix food coloring to perform the key agreement [12]. In our course, we faced the additional challenge of remote teaching but still managed to find an effective way to carry out the DH unplugged activity, taking advantage of the features of the online medium (e.g., public meeting chat) while maintaining the "unplugged spirit" (see 2.3.2).

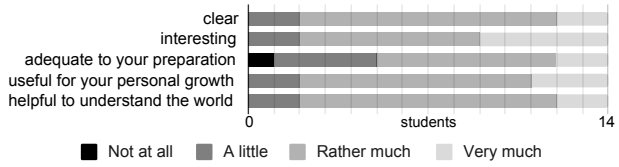

Figure 3: Student perceptions of course activities

Greenlaw et al. took a similar approach when they used the message board of an undergraduate class to demonstrate how a person-in-the-middle attack on a public-key cryptosystem works [14], as did Gramm et al. when they developed an online interactive simulation for RSA with small numbers [13].

4 EXPERIENCE EVALUATION

At the end of the course, we asked the students to fill out two Google Forms to get their feedback and assess their learning. No marks were foreseen for these final activities. Both questionnaires were completed by the same 14 (out of 15) students.

4.1 Learning assessment

We wrote a 2000-words summary [24] of the important ideas and concepts covered in the course, identifying 43 key passages. For each one, the students had to choose between the right filling and a wrong alternative. We wanted the activity to be also an opportunity for the students to review the important contents of the course, so we structured it as a narrative[4]. The results were positive: out of 43 choices, the mean of correct ones was 32.5, and the median was 34, with a range of correct choices between 17 and 41.

4.2 Course satisfaction

We asked the students about their learning experience and the perspectives the course gave them. All answers were mandatory.

Attendance was high: out of 14, 13 students followed the first lesson, 12 the second, 11 the third, and 12 the fourth. About the course length, 9 students found it 'Adequate', 4 'Too short', and only 1 found it 'Too long'. About the activities, most of the students found them clear, interesting, useful for their personal growth and for understanding the world. Some students felt that the activities were too difficult for their previous knowledge (fig. 3).

When asked if they found difficulties during the course, on a four-point scale from 'Never' to 'Always', 1 student chose 'Never', 11 'Sometimes', and 2 'Often'. The difficulties reported (more than one could be chosen) were related to the activities (6), timing and organization (6), the remote setting (5), and homework (2).

On a four-point scale from 'Not at all' to 'Very much' satisfied: on the interaction with the teachers and their support, 9 students chose 'Rather much' and 5 'Very much'; on the overall satisfaction with the course, 1 student chose 'A little', 8 'Rather much', and 5 'Very much'. 13 students would recommend the course to a friend.

Regarding the tools used (see 2.3.1) and created (see 2.3.2), students highly appreciated the DH activity and the explanations with animated slides. Some students found difficulties in both Snap! playgrounds and homework. See fig. 4.

[4]The English translation of the summary is available at [24].

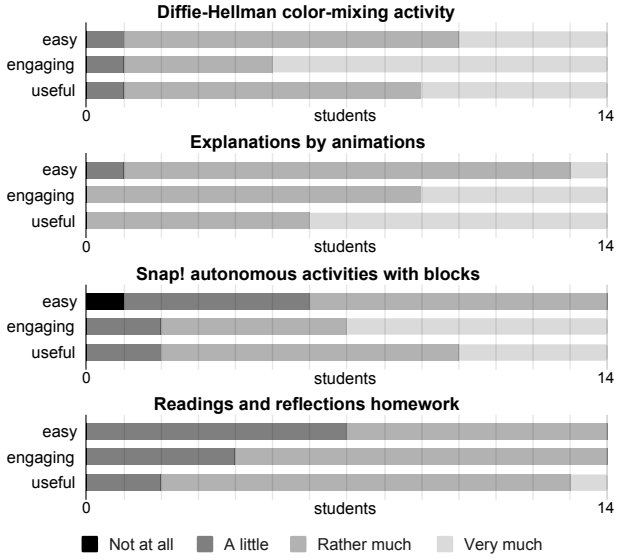

Figure 4: Student evaluation of course elements

About the perceived utility of the course, most of the students found it useful to better understand cryptography, its matter of study, and its role in society. Also, students felt they better understood CS and Math's role in society. While the course clearly sparked interest in cryptography in most students, it also helped stimulate interest in CS in 2/3 of them, while only less than half felt it increased their interest in Math. See fig. 5.

4.2.1 Open comments from students. All but one of the comments were positive. Most students remarked that the activities were interesting and fun (e.g., "I liked the fact that through Snap! we were able to play and experiment with cryptography"). According to them, lessons were engaging and well organized, even remotely. One said she really appreciated the space reserved for collective discussions in each lesson. Many students would have liked the course to be longer. One would have wanted to delve into Snap! programming beyond the scope and boundaries of the playgrounds.

The only non-positive comment reports difficulties in understanding and suggests more explanations, also through animations, before the hands-on activities.

As a positive note, a student used the Caesar cipher playground to encrypt (with a key we had to figure out) her positive comment.

5 OBSERVATIONS AND FINDINGS

5.1 On pedagogy and intervention results

The DH activity was received very well: simulating the insecure channel through the meeting public chat was an essential metaphor to understand the protocol. The students found the activity engaging and fun (fig. 4), confirming our positive impressions on the spot. We refer to it as "remote-unplugged" because it has almost all the characteristics of a *CS Unplugged activity* (i.e., *real computer science*—presenting fundamental CS concepts and algorithms; *learning by doing*; *fun*; *co-operative*; *stand-alone*; *resilient* to student errors; see [8]) except that it was delivered via technological devices. In

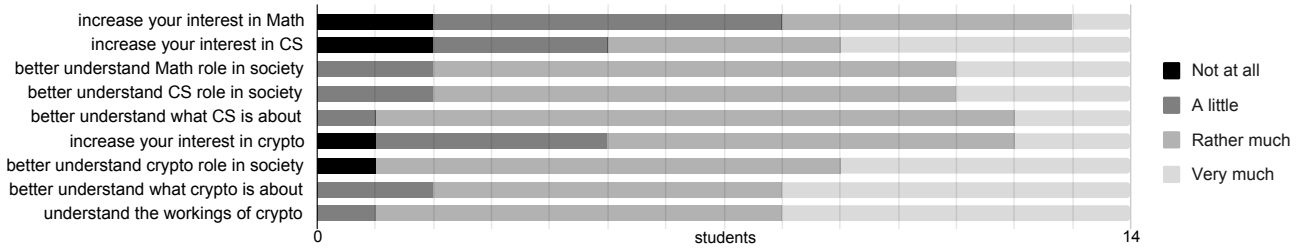

Figure 5: Student perceptions of what the course was useful for

our case, the devices acted as a necessary means of communication rather than being the specific tool of CS. We believe that the interactive, executable-only Snap! project made the DH activity concrete and easy to follow, a good solution for an online setting.

The Snap! playgrounds we designed and implemented worked well, for the easier concepts (e.g., Caesar cipher), as a way to get hands-on experience and understand the elements of a cryptosystem, some of the possible attacks, and its main limitations (e.g., the computational time required). However, half of the students found the more advanced playgrounds difficult. We believe that the setting did not help: the course was held online, with students we did not know in advance (and mostly did not know each other, coming from different classes). This resulted in severe "instructor blindness," making it hard for us to act as facilitators and provide the students with *optimal* guidance [37] while exploring the playgrounds. Thus, these activities resulted to be "minimally guided" [38], which can be too hard, especially for weaker students.

Throughout the course, we maintained a Socratic approach with collective discussions, guided by students' suggestions. They highly appreciated the space given to these interactions (see 4.2 and 4.2.1). For simpler cryptosystems, Snap! playgrounds were used for students to experiment before discussing and analyzing the cryptosystems. The guiding questions helped get the students to encounter precisely the aspects (and limitations) we wanted to highlight. Only after these concrete experiences, collective discussions were used to institutionalize the relevant knowledge. A similar approach was adopted with the DH activity. Even for the more "transmissive" part on public-key cryptosystems, the schemes' formalization occurred after discussing the systems that the students proposed intuitively after the essential ingredients (i.e., the key pair) had been presented.

The final assessment [24] focused on cryptography core ideas, in line with the goals of our intervention. It showed very good results (see 4.1), indicating that the main contents of the course were received. This is good for two main reasons. Every citizen should have the tools to understand today's digital society. It is also an opportunity for university and professional orientation towards the disciplines involved. The satisfaction survey confirmed the achievement of these two goals, indicating that the students better understood the cryptography role in society, its matter of study, and increased their interest in it.

5.1.1 On teaching programming. Our playgrounds can be seen as task-specific languages (see 2.3.2), with narrow scopes on specific cryptosystems. They do not aim to teach students how to program. However, they can convey some general principles about programming, for example, the fact that "programs are assembled out of basic elements, and different orderings of elements can sometimes have the same result, and even that the program determines the computer's behavior (there's no magic)" [43, p. 186].

Compared to other task-specific programming languages [43], our activities expose some classic programming concepts (e.g., sequence, function composition, variables, lists). In perspective, more custom blocks could be developed, together with activities requiring more extensive programming (e.g., using other fundamental elements of structured programming like conditionals and loops; "looking inside" the provided custom blocks to understand and adapt them). Currently, inspecting the code of our playgrounds does not have the educational value it could have. Curious students would find some JavaScript and a few uninteresting workarounds to overcome Snap! limitations. Hence, we are considering building a stack of *notional machines* at different abstraction levels so that students can progressively see more details by looking inside the blocks without being overwhelmed by all the complexity at once [31].

5.2 Suggestions for adoption and adaption

To help other educators adopt and adapt our course, we provide the contents and the learning path for crypto core ideas through representative systems and schemes (see 2.2) and all the tools (e.g., the Snap! playgrounds) and materials (e.g., the final assessment) [24].

The level of guidance in the hands-on activities can be adjusted. If the course is face-to-face, instructors can get a clearer picture of students' difficulties and address them immediately while still leaving a high degree of freedom. If held remotely, we suggest more frequent check-ins and re-alignments to provide adequate guidance.

Instructors can set aside additional time to cover some of the topics we introduced only through homework (e.g., the binary operations underlying the mechanisms of modern symmetric block ciphers) so that students can benefit from the teacher's support when they first encounter them.

Although the course aims at principles of cryptography, we acknowledge that one cannot understand the core ideas of a discipline without addressing them in concrete, even simple, scenarios or systems in which they occur. Should the need arise to readjust timings, we would suggest spending more time on explorations and discussions rather than tackling new cryptosystems—unless these are strategic to other relevant ideas instructors want to convey.

ACKNOWLEDGMENTS

Work partially funded by ACM SIGCSE Special Projects 2020.

REFERENCES

[1] Rachid Anane and Mohammad T. Alshammari. 2020. A Dynamic Visualisation of the DES Algorithm and a Multi-Faceted Evaluation of Its Educational Value. In *Proceedings of the 25th ACM Conference on Innovation & Technology in Computer Science Education* (Trondheim, Norway) *(ITiCSE '20)*. ACM, New York, NY, USA, 370–376. https://doi.org/10.1145/3341525.3387386

[2] Tim Bell, Jason Alexander, Isaac Freeman, and Mick Grimley. 2009. Computer Science Unplugged: school students doing real computing without computers. *New Zealand Journal of Applied Computing and Information Technology* 13, 1 (2009), 20–29.

[3] Tim Bell, Harold Thimbleby, Mike Fellows, Ian Witten, Neil Koblitz, and Matthew Powell. 2003. Explaining cryptographic systems. *Computers & Education* 40, 3 (2003), 199–215. https://doi.org/10.1016/S0360-1315(02)00102-1

[4] Carlo Bellettini, Violetta Lonati, Dario Malchiodi, Mattia Monga, Anna Morpurgo, Mauro Torelli, and Luisa Zecca. 2014. Informatics Education in Italian Secondary Schools. *ACM Trans. Comput. Educ.* 14, 2, Article 15 (2014), 6 pages. https://doi.org/10.1145/2602490

[5] Christopher Brown, Frederick Crabbe, Rita Doerr, Raymond Greenlaw, Chris Hoffmeister, Justin Monroe, Donald Needham, Andrew Phillips, Anthony Pollman, Stephen Schall, John Schultz, Steven Simon, David Stahl, and Sarah Standard. 2012. Anatomy, Dissection, and Mechanics of an Introductory Cyber-Security Course's Curriculum at the United States Naval Academy. In *Proceedings of the 17th ACM Conference on Innovation & Technology in Computer Science Education* (Haifa, Israel) *(ITiCSE '12)*. ACM, New York, NY, USA, 303–308. https://doi.org/10.1145/2325296.2325367

[6] Suzanne Fox Buchele. 2013. Two Models of a Cryptography and Computer Security Class in a Liberal Arts Context. In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education* (Denver, Colorado, USA) *(SIGCSE '13)*. ACM, New York, NY, USA, 543–548. https://doi.org/10.1145/2445196.2445360

[7] European Commission. Joint Research Centre. 2017. *DigComp 2.1: the digital competence framework for citizens with eight proficiency levels and examples of use*. Publications Office. https://doi.org/10.2760/38842

[8] CS Unplugged. [n.d.]. Principles. https://csunplugged.org/en/principles/

[9] CSTA. 2017. *CSTA K-12 Computer Science Standards, rev. 2017*. Technical Report. Computer Science Teachers Association. http://www.csteachers.org/standards

[10] Department of Education. 2013. *National curriculum in England: computing programmes of study*. https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study

[11] Pranita Deshpande, Cynthia B. Lee, and Irfan Ahmed. 2019. Evaluation of Peer Instruction for Cybersecurity Education. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. ACM, New York, NY, USA, 720–725. https://doi.org/10.1145/3287324.3287403

[12] Rachel E Fees, Jennifer A da Rosa, Sarah S Durkin, Mark M Murray, and Angela L Moran. 2018. Unplugged cybersecurity: An approach for bringing computer science into the classroom. *International Journal of Computer Science Education in Schools* 2, 1 (2018), 3–13. https://doi.org/10.21585/ijcses.v2i1.21

[13] Andreas Gramm, Malte Hornung, and Helmut Witten. 2012. Email for You (Only?): Design and Implementation of a Context-Based Learning Process on Internetworking and Cryptography. In *Proceedings of the 7th Workshop in Primary and Secondary Computing Education* (Hamburg, Germany) *(WiPSCE '12)*. ACM, New York, NY, USA, 116–124. https://doi.org/10.1145/2481449.2481477

[14] Raymond Greenlaw, Christopher Brown, Zachary Dannelly, Andrew Phillips, and Sarah Standard. 2015. Using a Message Board as a Teaching Tool in an Introductory Cyber-Security Course. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (Kansas City, Missouri, USA) *(SIGCSE '15)*. ACM, New York, NY, USA, 308–313. https://doi.org/10.1145/2676723.2677221

[15] Mark Guzdial. 2021. *Helping social studies teachers to teach data literacy with Teaspoon languages*. Retrieved January 14, 2022 from https://computinged.wordpress.com/2021/12/22/helping-social-studies-teachers-to-teach-data-literacy-with-teaspoon-languages/

[16] Mark Guzdial and Bahare Naimipour. 2019. Task-Specific Programming Languages for Promoting Computing Integration: A Precalculus Example. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '19)*. ACM, New York, NY, USA, Article 21, 5 pages. https://doi.org/10.1145/3364510.3364532

[17] Wen-Jung Hsin. 2005. Teaching Cryptography to Undergraduate Students in Small Liberal Art Schools. In *Proceedings of the 2nd Annual Conference on Information Security Curriculum Development* (Kennesaw, Georgia) *(InfoSecCD '05)*. ACM, New York, NY, USA, 38–42. https://doi.org/10.1145/1107622.1107632

[18] K-12 CS Framework. 2016. *K–12 Computer Science Framework*. Technical Report. http://www.k12cs.org

[19] Abdullah Konak. 2014. A cyber security discovery program: Hands-on cryptography. In *2014 IEEE Integrated STEM Education Conference*. 1–4. https://doi.org/10.1109/ISECon.2014.6891029

[20] Abdullah Konak. 2018. Experiential Learning Builds Cybersecurity Self-Efficacy in K-12 Students. *Journal of Cybersecurity Education, Research and Practice* 2018, 1 (2018). https://digitalcommons.kennesaw.edu/jcerp/vol2018/iss1/6

[21] Ákos Lédeczi, Miklós Maróti, Hamid Zare, Bernard Yett, Nicole Hutchins, Brian Broll, Péter Völgyesi, Michael B. Smith, Timothy Darrah, Mary Metelko, Xenofon Koutsoukos, and Gautam Biswas. 2019. Teaching Cybersecurity with Networked Robots. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. ACM, New York, NY, USA, 885–891. https://doi.org/10.1145/3287324.3287450

[22] Liceo Matematico [n.d.]. https://www.liceomatematico.it/

[23] Anke Lindmeier and Andreas Mühling. 2020. Keeping Secrets: K-12 Students' Understanding of Cryptography. In *Proceedings of the 15th Workshop on Primary and Secondary Computing Education* (Virtual Event, Germany) *(WiPSCE '20)*. ACM, New York, NY, USA, Article 14, 10 pages. https://doi.org/10.1145/3421590.3421630

[24] Michael Lodi, Marco Sbaraglia, and Simone Martini. 2021. Big Ideas of Cryptography in K-12. https://bigideascryptok12.bitbucket.io/

[25] Michael C. Loui and Maura Borrego. 2019. Engineering Education Research. In *The Cambridge Handbook of Computing Education Research*. Cambridge University Press, 292–322. https://doi.org/10.1017/9781108654555.012

[26] Jun Ma, Jun Tao, Jean Mayo, Ching-Kuang Shene, Melissa Keranen, and Chaoli Wang. 2016. AESvisual: A Visualization Tool for the AES Cipher. In *Proceedings of the 21st ACM Conference on Innovation & Technology in Computer Science Education* (Arequipa, Peru) *(ITiCSE '16)*. ACM, New York, NY, USA, 230–235. https://doi.org/10.1145/2899415.2899425

[27] Alasdair McAndrew. 2008. Teaching Cryptography with Open-Source Software. *SIGCSE Bull.* 40, 1 (2008), 325–329. https://doi.org/10.1145/1352322.1352247

[28] Jens Mönig and Brian Harvey. [n.d.]. Snap! - Build Your Own Blocks. https://snap.berkeley.edu/

[29] Joint Task Force on Cybersecurity Education. 2018. *Cybersecurity Curricula 2017: Curriculum Guidelines for Post-Secondary Degree Programs in Cybersecurity*. ACM, New York, NY, USA. https://dl.acm.org/doi/book/10.1145/3184594

[30] Seymour Papert. 1980. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, Inc., New York, NY, USA.

[31] Marco Sbaraglia. 2021. A Necessity-Driven Learning Design for Computer Science. In *Proceedings of the 26th ACM Conference on Innovation & Technology in Computer Science Education V. 2* (Virtual Event, Germany) *(ITiCSE '21)*. ACM, New York, NY, USA, 664–665. https://doi.org/10.1145/3456565.3460017

[32] Marco Sbaraglia, Michael Lodi, and Simone Martini. 2021. A Necessity-Driven Ride on the Abstraction Rollercoaster of CS1 Programming. *Informatics in Education* 20, 4 (2021), 641–682. https://doi.org/10.15388/infedu.2021.28

[33] Dino Schweitzer and Jeff Boleng. 2009. Designing Web Labs for Teaching Security Concepts. *J. Comput. Sci. Coll.* 25, 2 (2009), 39–45.

[34] Dino Schweitzer and Wayne Brown. 2009. Using Visualization to Teach Security. *Journal of Computing Sciences in Colleges* 24, 5 (2009), 143–150.

[35] Simon Singh. 1999. *The code book : the science of secrecy from ancient Egypt to quantum cryptography*. Fourth Estate, London.

[36] Joel Sommers. 2010. Educating the next Generation of Spammers. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (Milwaukee, Wisconsin, USA) *(SIGCSE '10)*. ACM, New York, NY, USA, 117–121. https://doi.org/10.1145/1734263.1734302

[37] Keith S. Taber. 2012. Constructivism as educational theory: Contingency in learning, and optimally guided instruction. In *Educational theory*, Hassaskhah Jaleh (Ed.). Nova, New York, NY, USA, 39–61.

[38] Sigmund Tobias and Thomas M. Duffy (Eds.). 2009. *Constructivist instruction: Success or failure?* Routledge.

[39] Claude F. Turner, Blair Taylor, and Siddharth Kaza. 2011. Security in Computer Literacy: A Model for Design, Dissemination, and Assessment. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (Dallas, TX, USA) *(SIGCSE '11)*. ACM, New York, NY, USA, 15–20. https://doi.org/10.1145/1953163.1953174

[40] Dirk van der Linden, Awais Rashid, Emma Williams, and Bogdan Warinschi. 2018. Safe Cryptography for All: Towards Visual Metaphor Driven Cryptography Building Blocks. In *Proceedings of the 1st International Workshop on Security Awareness from Design to Deployment* (Gothenburg, Sweden) *(SEAD '18)*. ACM, New York, NY, USA, 41–44. https://doi.org/10.1145/3194707.3194709

[41] Valdemar Švábenský, Jan Vykopal, and Pavel Čeleda. 2020. What Are Cybersecurity Education Papers About? A Systematic Literature Review of SIGCSE and ITiCSE Conferences. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. ACM, New York, NY, USA, 2–8. https://doi.org/10.1145/3328778.3366816

[42] Wikipedia contributors. 2021. *Diffie–Hellman key exchange — Wikipedia, The Free Encyclopedia*. Retrieved August 13, 2021 from https://en.wikipedia.org/w/index.php?title=Diffie%E2%80%93Hellman_key_exchange&oldid=1038627636

[43] Aman Yadav and Ulf Dalvad Berthelsen. 2021. *Computational Thinking in Education*. Routledge. https://doi.org/10.4324/9781003102991

[44] Maximilian Zinkus, Oliver Curry, Marina Moore, Zachary Peterson, and Zoë J. Wood. 2019. Fakesbook: A Social Networking Platform for Teaching Security and Privacy Concepts to Secondary School Students. In *Proc. of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. ACM, New York, NY, USA, 892–898. https://doi.org/10.1145/3287324.3287486