Informatics in Education, 2018, Vol. 17, No. 1, 93–103
© 2018 *Vilnius University*
DOI: 10.15388/infedu.2018.06

Computational Thinking Enrichment: Public-Key Cryptography

Frances ROSAMOND
The University of Bergen, Institute of Informatics
Thormohens Gate 55, Bergen 5020, Norway
email: frances.rosamond@uib.no

Received: August 2017

Abstract. The Computer Science Unplugged activities and project has been an influential STEM (Science, Technology, Engineering & Mathematics) initiative, providing enrichment and teaching activities supporting computational thinking. Many of its activities are suitable for children. One of the most popular Unplugged activities is "Kid Krypto", invented by Mike Fellows and Neal Koblitz. Kid Krypto demonstrates the mathematics underlying public-key cryptography without using advanced mathematics. The paper gives an example of a Kid Krypto-style encryption system that is based on disjoint cycles in a graph or network and which is accessible to a very young audience. Also described is the original Kid Krypto system which is based on a version of dominating set called perfect code. The paper urges research scientists to participate in mathematical sciences communication and outreach.

Keywords: Kid Krypto, Computer Science Unplugged, cryptography, computational thinking, STEM.

1. Introduction

Almost every area of our life is influenced by computers and concepts of Informatics. School systems are recognizing that Informatics is more than programming, use of software packages or word processing. The Informatics curriculum in many countries is being revised or invented to emphasize *computational thinking,* a term used by Jeannette Wing to describe attitudes and skills of problem solving related to computer science (Wing, 2006). Teachers are seeking classroom activities and methods for implementing the new curriculum requirements. This paper addresses that need by describing two cryptography enrichment activities. The first has never been published and is based on covering a directed graph with vertex disjoint cycles. The second is a review of the Kid Krypto system in *Computer Science Unplugged* (`www.csunplugged.org`) and is based on a type of dominating set called perfect code.

Sharing or hiding secrets is exciting for young people. Most are aware that cyber-espionage can jeopardize classified information ranging from their parents' electronic credit card transactions to high level government secrets. Personal privacy has become an issue. Michael Fellows, Professor of Computer Science at the University of Bergen, and Neal Koblitz, Professor of Mathematics at the University of Washington, have shown that cryptography is a topic that can be developed into engaging, thought-provoking activities for young students (Fellows and Koblitz, 1993).

Educational systems in many countries, such as Lithuania and New Zealand, are establishing informatics in the curriculum even in primary school. Poland is another country where computer science has been introduced and integrated with the curriculum for K-12 as effective of September 1, 2017. New Zealand has recently included informatics as a formal requirement (Bell *et al.*, 2014), and Tim Bell (University of Christchurch) is developing classroom materials that engage children in foundational computation concepts (Bell *et al.*, 1999; Bell *et al.*, 2012). A 2015 teacher survey on the IT/Informatics curriculum in Lithuania by Valentina Dagienė and Gabrielė Stupurienė (Vilnius University) (2016a), found that almost all the teachers held a strong attitude that basic knowledge of Informatics should be delivered for every student from K-12, and that students need to learn to think algorithmically.

On his webpage (`https://www.math.washington.edu/~koblitz/`) and in (Koblitz, 1997) Neal Koblitz, an inventor of elliptic curve cryptography, discusses five reasons why cryptography has a tremendous potential to enrich algorithmic thinking. As a front-line researcher in the area, Koblitz knows that "a central theme in cryptography is what we do *not* know or can*not* do. The security of a cryptosystem often rests on our inability to efficiently solve a problem in algebra, number theory, or combinatorics. Thus, cryptography provides a way to counterbalance the impression that students often have that with the right formula and a good computer any math problem can be quickly solved." Koblitz gives examples of cryptography that provide an excellent opportunity for interdisciplinary projects and can be done in primary or middle grades (Koblitz, 1997).

In several papers and as a co-author of the *Computer Science Unplugged* project (Bell *et al.*, 1999), the internationally known computer scientist Michael Fellows has emphasized that mathematics popularization is a research area of basic interest (Fellows *et al.*, 1994; Fellows, 1993; Fellows, 1991). Exciting mathematical science ideas can only find their way to children and their teachers or the general public with effort on the part of mathematicians and computer scientists to communicate about them in accessible ways. In mathematics (as is done in astronomy or biology), the frontiers of knowledge can and should be put within reach of young students. Public key cryptography is an opportunity to illustrate the rich mathematics underlying computer science turned into outreach activities, having students *do* math and computer science with the excitement and sense of discovery felt by practicing researchers.

The Turing Award is called the "Nobel prize of computing" and is worth a million US dollars. Young people would be interested to know that cryptography is deemed so important that research in this area has been awarded the prestigious prize *three* times. Cryptographers Whitfield Diffie and Martin Hellman were awarded the Turing Award in

2015 for inventing the first public-key cryptography system, allowing people who have never met or communicated to send secret information encrypted in a public key available for all to see. The Diffie-Hellman Protocol protects daily Internet communications and trillions of dollars in financial transactions. The 2012 Turing Award was awarded to Shafi Goldwasser and Silvio Micali for work that laid the complexity-theoretic foundations for the science of cryptography. They pioneered new methods to efficiently verify mathematical proofs in complexity theory. They studied how to protect against attacks and do encryption in a way that remains secure even if the secret memory containing the secret key is partially leaked. Ron Rivest, Adi Shamir, and Leonard Adleman won the 2002 Turing Award. Their "RSA" algorithm (named using their surname initials) is one of the most widely used encryption methods. It uses factoring a number into its prime factors to create a one-way function that is hard to invert. All methods of finding such factors would take many months or years by even the fastest modern supercomputers.

It is important that researchers in every area of the mathematical sciences create outreach activities to support schoolteachers who are trying to communicate the mathematics that is known, or not known or that is not able to be known. It is not an easy task to describe front-line science or basic research, but it can be done, as attested by the variety of activities in *Computer Science Unplugged* and *This is MEGA-Mathematics* (`http://www.ccs3.lanl.gov/mega-math/`). The *Creative Mathematical Sciences Communication* (CMSC) conference (`www.csmaths.org` and `www.cmsc.nz`) is a venue for supporting scientists in creating accessible ways of engaging children with the big ideas from mathematics and computer science, including using outdoor activities, art, dance, drama and all forms of storytelling.

2. Asymmetric Encryption

The significance of the work of the Turing Award winners lies in their development of asymmetric encryption. Previous methods were symmetric. That is, the same "key" used to encrypt a message was used in reverse to decrypt it. An early example is that of 1900 B.C. Egyptians who wrapped a tape around a stick and wrote the secret message on the tape. When the tape was unwound, the writing was meaningless to a spy. The intended receiver of the message would have a stick of the same diameter and use it to re-wrap the tape and decipher the message.

Many of us are familiar with symmetric systems that shift letters of the alphabet to encrypt a message. For example, the *Julius Caesar encryption* depends on alphabet shifts.

Plain:	ABCD**E**FGHIJKLMNOPQR**S**TUVWX**Y**Z
Cipher:	XYZA**B**CDEFGHIJKLMNO**P**QRSTU**V**W

The message "YES" gets transformed to "VBP". To decrypt--do the reverse. Symmetrically, the receiver shifts the letters back in order to read the message. The mathematical concept here is *addition in a cyclic group*. Actual wheels with gears were used at various times in history. *Frequency analysis* can be used to break the cypher. Students can guess which letter appears most frequently. They can compile frequency statistics from paragraphs in books. Many questions can arise; such as: What are the second and third most frequently occurring letters? Are these the same letters in any language? What strategies could one use for a short text to which the assumption about the most frequent letter does not apply? For excellent detail on cryptography addressed to a general audience see Cryptography in the *New World Encyclopedia*.

In contrast to the symmetric system, asymmetric public-key crypto uses two different keys: a public key and a private secret key. Public-key cryptography has three players, generally called Alice, Bob and the Adversary. Alice has a public key, a private key, and an encryption method.

1. Alice publishes her public key in a "phonebook", a trusted, neutral source. She also has an encryption method for anyone using her public key to send her a message. Alice has a secret private key to decode encrypted messages and read them.
2. Bob knows the encryption method for using Alice's public key to send her a message.
3. The Adversary (sometimes called "Eve" for eavesdropper) tries to crack their communication system, knowing how it works (in general).

In his blog, Public Key Encryption for Kids (Oct 24th, 2013), Martin Sústrik describes a simple game that gives an intuitive, hands-on experience with a simple asymmetric encryption system. Buy a dictionary of some exotic language that the children do not know, such as the Eskimo language. Most foreign language dictionaries come in two bands: English-Eskimo dictionary and Eskimo-English dictionary. The public-key is the English-Eskimo dictionary. Everyone has access to this dictionary. The private key is the Eskimo-English dictionary and only one person has this (the intended receiver of the secret message.) To encrypt a message, the sender uses the public key (English-Eskimo dictionary) to translate the message, word-by-word, from English to Eskimo. This is the encryption method. The owner of the secret key (Eskimo-English dictionary) can then easily decrypt the message by translating it back into English. Notice that two different keys are used. If the message gets intercepted by any other game participant (a spy), then decrypting it would be an extremely time-consuming activity. Each word of the message would have to be found in the English-Eskimo dictionary, which would mean scanning the whole dictionary in a page-by-page and word-by-word manner! Students quickly learn that encryption may be easy while decryption may be so hard as to be basically impossible.

3. The Disjoint Directed Cycle Cover Cryptosystem

The first cryptosystem we describe is suitable for very young children. They need only do simple multiplication and subtraction. First, we introduce some basic definitions from graph theory. A *directed graph* (sometimes called a directed network or *digraph*) is a collection of dots called *vertices*, which are connected by lines called *arcs* or *directed edges*. Each arc has an arrow to indicate that it is directed from one vertex to another.

The *head* of the arrow is the end with the point, indicating the direction. A *directed cycle* is exactly what it sounds like, a sequence of vertices starting and ending at the same vertex. For each two consecutive vertices of a cycle, there is an arc directed from the earlier vertex to the later one.

If every vertex is included in a cycle, then we say the cycles *cover* the graph. If the cycles of the cover have no vertices in common, the cover is called *vertex-disjoint* or sometimes simply *disjoint cycle cover*. That is, every vertex has one and only one directed cycle passing through it. Decryption depends on *finding the cycles*.

The encryption replaces vertex labels with labels on the arcs (directed edges) as illustrated in Fig. 1. For example, in (a) the vertices are labeled with the values "2", "3", and "–4". In (b), each arc is labeled with *twice the tail minus the head*. That is, "2 (2) – 3 = 1" and "2 (3) – (–4) = 10". The Fig. 1(c) shows that the sum of the arc labels "(2a – b) + (2b – c) + (2c – a) = a + b + c" and the sum of the vertex labels (a + b + c) on any directed cycle is the same.

It is straightforward to construct a digraph with a disjoint cycle cover. Begin by drawing the cycles: vertices with arcs going from one vertex to another all pointing in the same direction until you are back at the first vertex.

Once the directed cycles have been drawn, disguise them in a graph by adding extra arcs. Connect the cycles with extra arcs, *taking care not to create additional cycles*.

Draw enough additional arcs to disguise the cycles.

The cycles in Fig. 2(a) are disguised with three additional arcs in Fig. 2(b). No additional cycles are created.

The digraph (with the hidden disjoint cycle cover) is the public code. Suppose Alice and Bob are planning a surprise. Alice needs to know what time Bob will arrive to her house, but they do not want anyone else to know. Alice creates a digraph with a disguised disjoint cycle cover as above. Anyone can see the digraph. If the cycles are well disguised, then it does not matter who sees it. It is the public code for anyone to see.

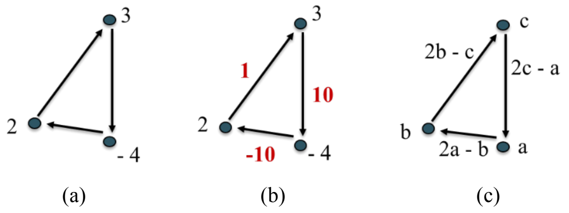


Fig. 1. The encryption method: "Twice the tail minus the head".

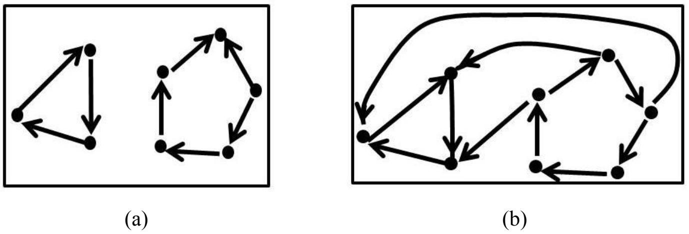


Fig. 2. (a) Draw the secret directed cycles. (b) Additional arcs disguise the secret cycles.

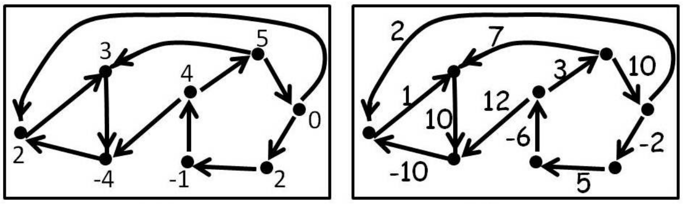


Fig. 3. (a) The sum of the numbers on the vertices are the secret message.
(b) The numbers on the arcs encode and disguise the message.

Bob plans to arrive at 11. He will use Alice's digraph to let her know. Bob does not know where the directed cycles are in the digraph. (This is the secret key that only Alice knows. She has disguised the cycles well.) Bob knows how to encrypt a message in the directed graph. He is using the "twice the tail minus the head" encryption method.

Bob encrypts the message in the following manner. Bob labels the vertices so that all the vertices in the graph add to 11. For every arc, Bob multiplies the vertex number at the tail of the arc by 2, and then subtracts the vertex number at the head of the arc. He puts this value on the arc. He does this for every arc, as in Fig. 3(b), and then erases all the vertex numbers. He does not allow anyone to see what he is doing. All of this is done secretly. Bob destroys all evidence of the calculations.

Once the arcs have been annotated as in Fig. 3(b), anyone can look at the digraph. They will find it meaningless. Only Alice knows where the cycles are. Alice decodes the message by adding the numbers on the edges of the disjoint cycles. She can ignore the disguising arcs. (They were there only to confuse a spy.) Alice finds Bob's message of 11.

Look at what has been accomplished. Alice can publish her public-key digraph in her school newsletter, on the internet, or put it at the bottom of her email. The clever part of public-key encryption is that with complete secrecy and with no prior communication, *anyone* using the encryption method can send Alice a secret message. Only Alice or someone else with the private key can decrypt and read the message. Even if we know that a given directed graph has a disjoint cycle cover, it might be very difficult to find it. The difficulty will be discussed later.

4. The Dominating Set Perfect Code Cryptosystem

A dominating set for a graph is a subset S of the vertices such that every vertex in the graph is either in S or is adjacent to at least one member of S. The vertices of the graph are said to be *dominated* by the vertices of S.

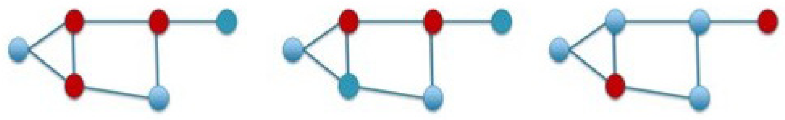
Each of the graphs below might represent a small village with the edges being streets. The Village Council wants to place ice-cream stands on street corners (the vertices) so that nobody needs to walk more than one block to get an ice-cream. The set of red vertices is a dominating set in each graph. Anyone living on a corner having an ice-cream stand doesn't need to walk anywhere. Others need only walk one block.

A *Perfect Code* is a special kind of dominating set. A dominating set S is a perfect code if every vertex is dominated by **exactly one** vertex of S. One may think of the perfect code vertices as the centers of stars. In graph Fig. 4(c), the two red vertices are a perfect code. Not all graphs have perfect codes.

Fig. 5 shows how to create a graph with a perfect code:

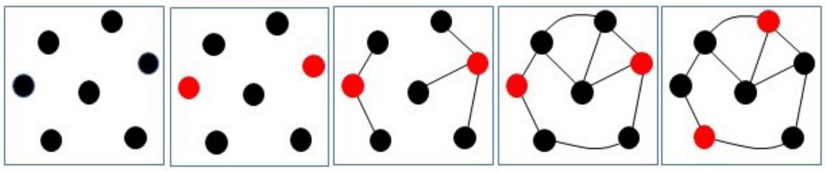
(1) Start with some vertices.
(2) Choose the perfect code vertices. These will become centers of the stars.
(3) Create stars, taking care that every vertex is connected to *exactly one* of the centers from (2). No two stars share a common vertex.
(4) Add additional disguising edges. *Do not draw any new edges from the center of any of the stars.*
(5) A perfect code in a graph may not be unique.

In the graph that has been constructed in Fig. 5, there are several different sets of vertices that are perfect codes. Note that each set has the same number of vertices (two). This is because a perfect code partitions the graph into non-overlapping components each containing the perfect code vertex and all of its neighbors. Thus, if a graph has more than one perfect code, they must all be the same size. This fact is used to create the private key in the Kid Krypto system.



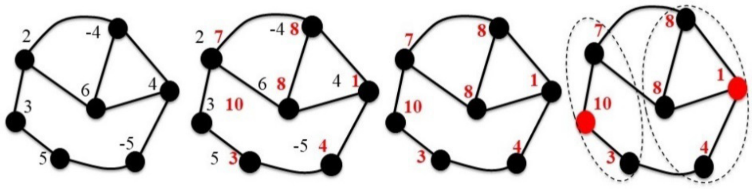
(a) (b) (c)

Fig. 4. The red vertices (ice-cream stands) are a dominating set in each graph.



(1) (2) (3) (4) (5)

Fig. 5. How to create a graph with a perfect code.



(a) (b) (c) (d)

Fig. 6. The Perfect Code encryption and decryption.

Alice wants to be able to receive an encrypted bit of information from Bob. She constructs a graph with a perfect code. The public key is the graph. Alice's private key is the perfect code. Only Alice knows which vertices are the perfect code.

The encryption method has two steps. Step 1: Privately and secretly, Bob puts numbers on vertices that sum to the message. The message is 11. This is what is shown in Fig. 6(a). Step 2: Sum the solid neighborhood of each vertex. These are the numbers in red next to each vertex. For example, the red number 10 is the sum of 5 + 3 + 2. The red number seven is the sum of 3 + 2 + (–4) + 6. No one should see these calculations. Bob now erases the black numbers. The graph (c) with only the red numbers are for anyone to see. This is the encrypted message.

To decipher the message, Alice takes the sum over the perfect code. In Fig. 6 (d), the dotted circles indicate the two non-overlapping components of the perfect code partition. The 10 in the first circle has "collected" all the original values of that component. Similarly, the 1 in the second component. The sum of the two perfect code vertices is 10 + 1 = 11. If Alice had used the perfect code illustrated in the graph in Fig. 5 (5), she would have added 8 + 3 = 11.

5. The Advisory

There are several issues that have not been discussed. What about *eavesdropping*? How difficult is it for a dedicated hacker to crack these codes? Another issue is *authentication*: when Alice receives a message from Bill, how does she know it really comes from him and not some interloper? Graph drawing is an art in itself (and a research field). What are some guidelines for drawing cryptography graphs that help disguise the secret key?

The encryption for Disjoint Directed Cycle Cover (DDCC) is less tedious than for Dominating Set Perfect Code (DSPC), but it is still a lot of work, although the calculations could be done with a software package. The DSPC Kid Krypto has been used by Neal Koblitz as a Math Day activity by over a thousand high school students at the University of Washington (See Koblitz' website at `https://sites.math.washington.edu/~koblitz/`). It has been used by schools and universities around the world with huge positive response.

The DSPC system can be solved with Gaussian elimination. On a graph with n vertices, number the vertices 1 to n, and call the original values given to those vertices a_1, a_2, ..., a_n. For each vertex write the equation that sums the solid neighborhood. If the vertices in the graph in Fig. 6(c) were numbered clockwise starting with the vertex having red number 1, then the equations would be:

$$a_1 + a_6 + a_2 = 1,\ a_2 + a_1 + a_3 = 4,\ \ldots,\ a_7 + a_4 + a_5 + a_6 = 8$$

There are as many equations as there are vertices. The message could be found by solving the system of equations and adding the values $a_1 + a_2 + ... + a_n$.

Regarding the DDCC system, a vertex disjoint cycle cover (the size of each cycle is greater than or equal to 2) in a directed graph corresponds to a perfect matching in a bipartite graph (Tutte, 1954). The computational time required for finding the perfect matching (and cycle cover) is proportional to n^3, where n is the number of vertices in the graph by an algorithm of Edmonds (Edmonds, 1965). The computational effort to solve the equations for DSPC using Gaussian elimination also is proportional to n^3, which is the cube of the number of equations. For small graphs the process is time-consuming by hand. Even sophisticated software systems would be challenged by a large graph which might involve 10,000 linear equations in 10,000 variables.

The two Kid Krypto systems discussed are based on simple graphs that require a cover or partition of the vertices. We would like to find other examples of graph-related systems, and a meta-theorem that guides the creation of other systems.

6. Conclusion

This paper offers classroom activities to assist teachers in motivating and arousing student interest in Informatics. While it is intended to support teachers in promoting computational thinking and in implementing new Informatics curriculum requirements, it may not be the sort of academic work that teachers can use for student evaluation. These crypto activities and many of the *Computer Science Unplugged* activities can be viewed as “Enrichment”.

There are many Informatics enrichment projects. The bubble-sort dance and other algorithmic folk-dances devised by (Katai *et al.*, 2014) bring multi-sensory methods into the teaching-learning environment. Karl Schaffer and Erik Stern, “Guys who Dance About Math”, have been showing amazing connections between mathematics and dance since 1987 (Schaffer and Stern, `http://www.mathdance.org`). Other arts provide mathematical science enrichment; such as, the on-stage productions of *Passion-Plays, Melodramas about Mathematics* created by Michael Fellows (Rosamond, 2012), and the mathematics-computer science-origami art of MIT scientists Erik and Martin Demaine (`www.erikdemaine.org`). Another form of Informatics enrichment that supports computational thinking is the Bebras project (Dagienė and Stupurienė, 2016b), an international Informatics contest already active in 55 countries.

Researchers considering an attempt at mathematical sciences enrichment can be sure that it does make a difference, and they will find a supportive community (Rosamond *et al.*, 2011). *Computer Science Unplugged* has been translated into 24 languages and has been a model for STEM initiatives and computational thinking around the world (Bell *et al.*, 2012; Rosamond, 2011; Rosamond, 2006). The *Unplugged* workshops given by Tim Bell and others in New Zealand have been instrumental in incorporating Computer Science as a formal academic subject in the final 3 years of the high school curriculum and hugely instrumental in the development of the new standards (Bell *et al.*, 2014).

Finally, as stated by Mike Fellows: “In the same way that children’s art is interesting as art and children’s writing is interesting as writing, mathematics with children can be interesting as mathematics.” (Fellows, 1991; 1993). Research problems sometimes “turn up” during classroom visits. Researchers and teachers are invited to the Creative Mathematical Sciences Communication conferences (`http://www.csmaths.org`, and `www.cmsc.nz`) to explore and create new ways of engaging young people in the discoveries and mysteries of the mathematical sciences.

References

Bell, T., Andreae, P., Robins A. (2014). A case study of the introduction of computer science in NZ schools. *ACM Transactions on Computing Education*, 14(2), article 10, 1–31.

Bell, T., Rosamond, F., Casey, N. (2012). Computer science unplugged and related projects in math and computer science popularization. In: Bodlaender, H.L., Downey, R., Fomin, F.V., Marx, D. (eds.), *The Multivariate Algorithmic Revolution and Beyond.* Springer LNCS 7370, 398–456.

Bell, T., Witten, I., Fellows, M. (1999). *Computer Science Unplugged: Offline Activities and Games for all ages.* Available at www.csunplugged.org. For additional activities, see Fellows, M. *This is MEGA-Mathematics*. `www.c3.lanl.gov/mega-math/menu.html`

Cryptography. (2017). *New World Encyclopedia*. Retrieved 11:36, December 30, 2017 from `http://www.newworldencyclopedia.org/p/index.php?title=Cryptography&oldid=1007978`

Dagienė, V., Stupurienė, G. (2016a). Informatics concepts and computational thinking in K-12 education: a Lithuanian perspective. *Journal of Information Processing,* 24(4), 732–739.

Dagienė, V., Stupurienė, G. (2016b). Bebras – a sustainable community building model for the concept based learning of informatics and computational thinking. *Informatics in Education*, 15(1), 25–44.

Edmonds, J. (1965). Paths, trees, and flowers. *Canad. J. Math.*, 17:449–467.

Fellows, M. (1993). Computer SCIENCE and mathematics in the elementary schools. In: Fisher, N., Keynes, H., Wagreich, P. (Eds.), *Mathematicians and Education Reform 1990–1991*. American Mathematical Society, Providence, RI, 143–163.

Fellows, M. (1991). Computer SCIENCE in the elementary schools. In: Fisher, N., Keynes, H., Wagreich, P. (Eds.), *Proceedings of the Mathematicians and Education Reform Workshop of Issues in Mathematics Education*. Conference Board of the Mathematical Sciences, Seattle, vol. 3, 152–172.

Fellows, M., Hibner, A., Koblitz, N. (1994). Cultural aspects of mathematics education reform. *Notices of the American Mathematics Society*, 41, 5–9.

Fellows, M., Koblitz, N. (1993). Kid Krypto. In: Brickell, E. F. (Ed.) CRYPTO 1992. Springer LNCS 740, 371–389.

Katai, Z., Toth, L., Adorjani, A.K. (2014). Multi-sensory informatics education. *Informatics in Education*, 13(2), 225–240.

Koblitz, N. (1997). Cryptography as a teaching tool. *Cryptologia,* 21(4), 317–326. A slightly shortened version is available on the website: `https://www.math.washington.edu/~koblitz/crlogia.html`

Rosamond, F. (2012) Passion plays: melodramas about mathematics. In: Bodlaender, H. L., Downey, R., Fomin, F.V., Marx, D. (Eds.), *The Multivariate Algorithmic Revolution and Beyond.* Springer LNCS 7370, 80–87.

Rosamond, F., Bardohl, R., Diehl, S., Geisler, U., Bolduan, G., Lessmöllmann, A., Schwill, A., Stege, U. (2011). Reaching out to the media: become a computer science ambassador. *Communications of the ACMI*, 54(3), 113–116.

Rosamond, F. (2011). A three-hour tour of some modern mathematics. In: Clark, J., Kissane, B., Mousley, J., Spencer T., Thornton, S. (Eds.), *Mathematics: Traditions and [new] practices*. AAMT and MERGA, Adelaide, 1038–1046.

Rosamond, F. (2006). On-line and off-line computer games and mathematical sciences popularization. In: Rosamond, F., Copes, L. (Eds.) *Educational Transformations: The Influences of Stephen I. Brown*. Authorhouse, Bloomington, 407–426.

Sústrik, M. (2013). *Public Key Encryption for Kids*. `http://250bpm.com/blog:28`

Tutte, W. T. (1954). A short proof of the factor theorem for finite graphs. *Canad. J. Math.*, 6, 347–352.

Wing, J. (2006) Computational Thinking. *Communications of the ACM*, 49(3), 33–35.

F. Rosamond (Professor) is Research Professor in the Institute for Informatics at the University of Bergen, Norway. Previous academic appointments include Charles Darwin University and University of Newcastle, Australia, and National University, San Diego, where she was the founding Chair of the Department of Mathematics. She has organized AWM *Sonia Kovalevskaya Mathematics Days for Girls* (`https://sites.google.com/site/awmmath/programs/kovalevsky-days`). Together with her spouse Michael Fellows, she has given *Computer Science Unplugged* workshops around the world. Frances Rosamond has published over 100 research papers in theoretical computer science and mathematics education. She is Editor of the Parameterized Complexity Newsletter and manager of the Parameterized Complexity community wiki (`www.fpt.wikidot.com`). In 2013, Rosamond founded the Creative Mathematical Sciences Communication (CMSC) conference series (`www.csmaths.org` and `www.cmsc.nz`).