Kid Krypto

Michael Fellows[1] and Neal Koblitz[2]

[1] University of Victoria, Department of Computer Science, Victoria, B.C. V8W 3P6, Canada
[2] University of Washington, Department of Mathematics, Seattle, Washington 98195, U.S.A.

Abstract. Cryptographic ideas and protocols that are accessible to children are described, and the case is made that cryptography can provide an excellent context and motivation for fundamental ideas of mathematics and computer science in the school curriculum, and in other venues such as children's science museums. It is pointed out that we may all be doing "Kid Krypto" unawares. Crayon-technology cryptosystems can be a source of interesting research problems; a number of these are described.

1 Introduction

The purpose of this paper is to open a discussion of cryptography for children. The fruits of this discussion can serve several worthwhile purposes, such as:
(1) the popularization of cryptography with children and the general public through such forums as children's science museums,
(2) the enrichment and improvement of the school mathematics curriculum by providing a stimulating context for logical and mathematical modes of thinking, and
(3) the amusement and intellectual stimulation of researchers.

We hope to convince the reader that devising ways to present the fundamental ideas of cryptography to children not only makes it possible to expose children to some electrifying mathematics, but also can be stimulating for our own research and can give us a fresh perspective on what we do.

In the following sections of this paper we will describe some examples of cryptographic ideas and constructions that have been or could easily be presented to children. Some of these might be characterized as *pre-cryptography*, i.e., they involve certain elements of cryptography but do not yet constitute a sophisticated protocol. Others are fully developed cryptosystems. The following assertions summarize our outlook.

• By its very essence, cryptography is a most excellent vehicle for presenting fundamental mathematical concepts to children.

Cryptography can be broadly defined as "mathematics/computer science in the presence of an adversary." Implicit in any discussion of cryptography are elements of drama, of theater, of suspense. Few things motivate children as much as wanting to defeat the "bad guys" (or play the role of bad guys themselves). Children are in the business of decrypting the world of adults, and many of the

video games that are now popular with children involve deciphering "clues" in order to achieve some goal.

Cryptography's ability to excite children has long been understood by advertisers of products like Rice Krispies and Crackerjacks. Many of us grew up quarreling with our siblings over who was going to get the decoder ring in the Crackerjacks box. Currently, boxes of Rice Krispies have on the back a "secret algorithm" age guessing game based on binary representation of integers.

• Kid Krypto is a source of interesting research problems.

We are essentially proposing a new criterion for deciding that a cryptosystem is worthy of attention: *accessibility*. As in the case of the more traditional criteria — efficiency and security — the search for cryptosystems that meet the accessibility standard naturally leads to interesting theoretical and practical questions. It is a new challenge to determine how much can really be done with minimal mathematical knowledge, and to find ways to present cryptographic ideas at a completely naive level. Moreover, experiences working with children have suggested some provocative problems in discrete mathematics and theoretical computer science, some examples of which will be described later.

A newly proposed cryptosystem might not be efficient enough or secure enough to compete in the realm of adult cryptography with those that already exist, but may nevertheless be of tremendous pedagogical value and merit our attention for that reason alone. In addition, it seems clear that interesting questions are likely to arise when we take a second look at some cryptosystems that have been too quickly forgotten.

Even a cryptosystem that can be broken in polynomial time may be of use in Kid Krypto, if the mathematics needed to break the system is less accessible than what is needed to implement the system. For example, some versions of the Perfect Code system explained below can be broken by linear algebra modulo m, but the system can be implemented using nothing more sophisticated than addition modulo m. In other words, we are proposing a new security hierarchy, with such notions as *accessible and secure for ages 5–10*, *accessible and secure for high school students*, etc.

• There is no sharp line between Kid Krypto and adult crypto, so it would be unwise for us to belittle the former.

After all, the security of all of our cryptosystems depends upon our assumed inability to perform certain mathematical tasks, such as discover a fast factoring algorithm. If a space alien from a very advanced civilization were to visit the earth, she might be surprised to find us using cryptosystems based on factoring and discrete log. Suppose that on her planet polynomial time algorithms to factor integers, find discrete logs in finite fields, and even find discrete logs on nonsupersingular elliptic curves have been known for centuries, and are routinely taught in high school. She would regard RSA, ElGamal, etc. as suitable only for pre-high school children in her culture.

In other words, in some sense *we are all doing Kid Krypto*, whether we know it or not.

• Kid Krypto is best done without computers.

This is *crayon-technology* cryptography. The tools needed are: pencils, a lot of paper, crayons of different colors, and perhaps some pieces of string or sticks. There is no material obstacle to introducing Kid Krypto in poor school districts as well as rich ones — in Watts and Soweto as well as in Santa Barbara and Scarsdale.

We see the absence of computers as a positive educational step. The public needs to understand that computer science is not about computers, in much the same way that cooking is not about stoves, and chemistry is not about glassware. What children need in order to become mathematically literate citizens is not early exposure to manipulating a keyboard, but rather wide-ranging experience working in a creative and exciting way with algorithms, problem-solving techniques and logical modes of thought.

Computers have been shamelessly oversold to teachers and school systems. In speaking to parents, teachers and school boards, many company representatives have taken the hard-sell approach: "If you don't buy our latest products you will be neglecting to prepare your children for the 21st century." Because of pressure from the companies and the media, computers have been fetishized to the extent that they threaten to become the Cargo Cult of the 21st century. Most of the time, computers serve as nothing more than an expensive distraction. The main beneficiaries of all the hype have been (1) computer hardware and software companies, and (2) educators who receive generous grants for the purpose of finding a way to use computers in the schools. Most schools would probably be better off if they threw their computers into the dumpster. It is our prediction that the Golly–Gee–Whiz–Look–What–Computers–Can–Do school of mathematical pedagogy will eventually come to be regarded as a disaster of the same magnitude as the "new math" rage of the 1960s.

2 Pre-Crypto

In order to present cryptography to young children, there are certain "building block" ideas which are useful to present first, and that are engaging in their own right. We point to three of these in particular, and describe how they can be simply presented.

(1) The notion of an algorithm, and of computational complexity.
(2) The notion of a one-way function.
(3) The notion of an information hiding protocol.

2.1 Algorithms and Complexity

There are now a tremendous number of delightful ways that the fundamental ideas of algorithmic procedure and computational complexity can be presented to young children. We mention here just a few of our favorites. The examples below have been tried out many times, with great success, with children sometimes as young as 5 or 6.

The first problem is Map Coloring. If you were to visit a first-grade classroom to share this lovely problem, you might very well arrive in a room full of children who are already coloring something anyway! You might tell the story of the poor Map-Colorer, trying to eke out a living with few crayons, and then pass out a map that needs to be colored. The definition of a proper coloring is visual, and can be illustrated with the maps at hand in the classroom. It is only a few minutes until most of the children understand the problem you have posed (finding out the minimum number of colors for the map you have passed out) and are puzzling away at it. As the children work to decrease the number of colors needed, you can display the "best known" solution so as to add to the excitement.

It is a good idea to come to the classroom with plenty of copies of 3 or 4 different maps. It is easy to generate a map that is two-colorable by overlaying closed curves. (Generating such a map is another topic the children may have fun thinking about). See Figure 1. In a typical first-grade classroom, children will figure out the algorithm for 2-coloring on their own, and they will see that it goes very quickly. It is easy enough to explain why it works: it has been called the "Have-to Algorithm" (if a country is red, then its neighbors have to be blue, and their neighbors have to be red, ...). Afterwards, you might distribute a map that requires 3 colors so that they can concretely contrast the 2-coloring experience with the apparent difficulty of finding a 3-coloring of a 3-colorable map.

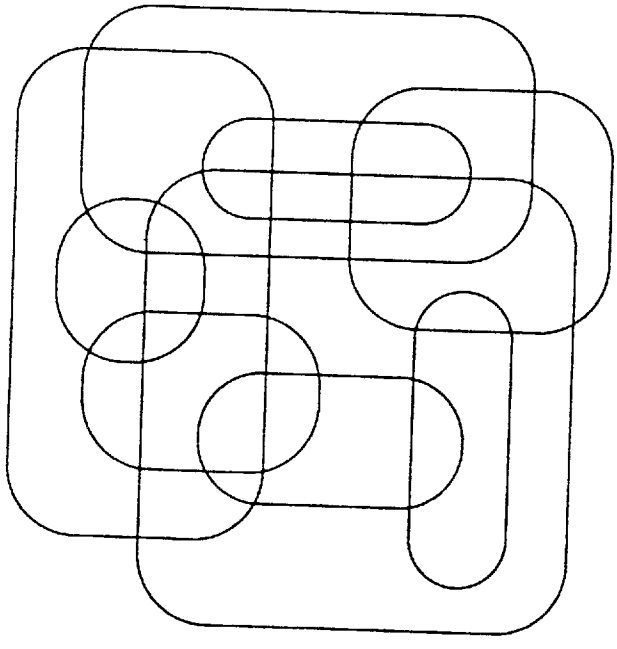

Fig. 1. Example of a 2-colorable map generated by overlaying closed curves

Another excellent topic for children is the problem of computing a Minimum

Weight Spanning Tree in a graph. Several efficient algorithms for solving this algorithmic problem are known and are routinely covered in college level courses on design and analysis of algorithms. The story we use to present the problem is meant to be entertaining, but it should be noted that there are many practical applications of this problem.

The children are given a map of Muddy City and told the story of its woes — cars disappearing into the mud after rainstorms, etc. The mayor insists that some of the streets must be paved, and poses the following problem. (1) Enough streets must be paved so that it is possible for everyone to travel from his or her house to anyone else's house by a route consisting only of paved roads, but (2) the paving should be accomplished at a minimum total cost, so that there will be funds remaining to build the town swimming pool. Thus, the children are asked to devise a paving scheme meeting requirement (1), connecting up the town by a network of paved roads, that involves a minimum total amount of paving. The cost of a paving scheme is calculated by summing the paving costs of the roads chosen for surfacing. For the map shown in Figure 2 a solution of total cost 23 can be found.

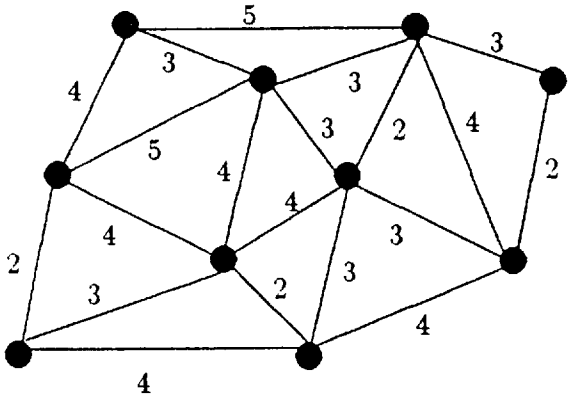

Fig. 2. Muddy City

The children work on the problem, usually in small groups, with the immediate objective of finding the best possible solution. This was typically recorded in a place that everyone could see. Students were asked to describe their strategies and ideas, both as they worked and in a concluding discussion. In classrooms where the students kept mathematics journals, they also wrote descriptions of the problem and of their ideas on how to solve it. These math journals were instituted with great success in a second-grade classroom and a fourth-grade classroom.

As part of the wrap-up discussion, we sometimes presented Kruskal's algorithm (one of several known algorithms for solving this problem efficiently). This method of finding an optimal solution consists simply of repeatedly paving

a shortest street which does not form a cycle of paved streets, until no further paving is required. It is interesting that the children often discovered some of the essential elements of Kruskal's algorithm and could offer arguments supporting them. (Rediscovering Kruskal's algorithm is not the point, of course.)

This problem can be presented to children of ages 5–6 by using maps with distances marked by ticks rather than numerals, so that the total amount of paving can be figured by counting rather than by sums.

Minimum Dominating Set is another problem that can provide a nice illustration of the idea of computational complexity. Recall that a *dominating set* in a graph $G = (V, E)$ is a set of vertices $V' \subseteq V$ such that for every vertex x of G, either $x \in V'$ or x has a neighbor $y \in V'$.

The stories we have told for this problem generally run to the theme of *facilities location*. For example, in Tourist Town we want to place ice-cream stands at corners so that no matter which corner you might be standing on, you need only walk at most one block to get an ice-cream. See Figure 3 for an example of a small, somewhat difficult graph for which $\gamma = 6$.

We allow some time for the children to puzzle over the map of Tourist Town, gradually producing more efficient solutions. Often, none of them is able to find the optimal solution with only six ice-cream stands. The children usually get an intuitive sense that Tourist Town is harder than Muddy City; the former does not seem to lend itself to solution by a quick and simple algorithm. The contrast between these two problems — one solvable in polynomial time and the other apparently intractable — provides a concrete introduction to the notion of computational complexity. We will return to the subject of dominating sets (of a special kind) in Section 5.

2.2 One-Way Functions

After explaining that no one knows a good algorithm for Tourist Town, one can show that there is, however, a simple algorithm for "working backwards," i.e., starting with a set of vertices V' that is to become an efficient solution and constructing a Tourist Town $G = (V, E)$ around it. Namely, one uses a two-step process. First, one forms a number of "stars" made up of "rays" (edges) emanating from the vertices in V'. (Two rays from different vertices in V' are allowed to have a common endpoint.) This graph clearly has V' as a solution. Figure 4 below shows this step in the case of the Tourist Town example. The second step is to "disguise" this easy-to-solve graph by adding more edges. This clearly does not increase the number of vertices required in a dominating set, but it does make the original built-in solution harder to see.

In this way it seems to be relatively easy to generate graphs on a small number of vertices (e.g. 25-30), having a known dominating set of size $6 \leq \gamma \leq 10$, for which it is relatively difficult to work out a solution of size γ by hand. However, no mathematical results are presently known that quantify the computational difficulty of problems such as this for graphs of small size.

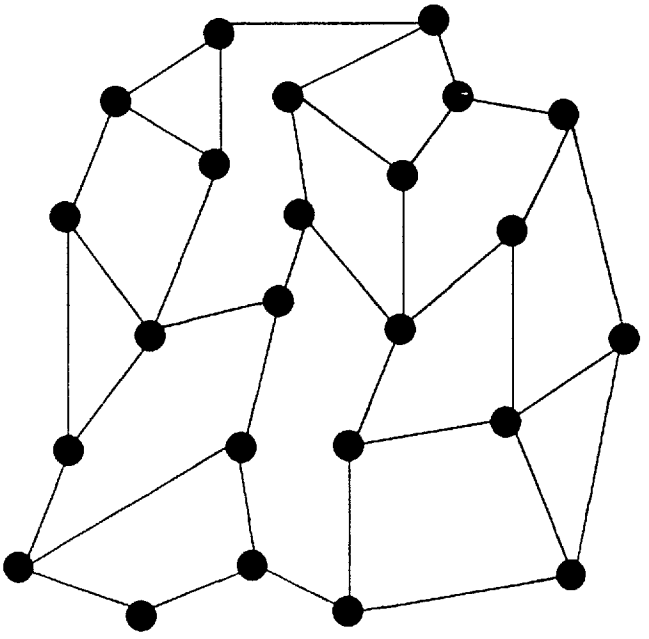

Fig. 3. Map of Tourist Town

This is a nice example of the idea of a one-way function. The children may look forward to trying out on their parents the process of creating a graph for which they secretly know a difficult-to-match solution. Open problem: can we sell this to Rice Krispies?

Remark 1. If the two-step "hidden solution" construction described above is modified by (1) in the first step, requiring that no two stars share a common vertex, and (2) in the second step, requiring that the additional disguising edges be added only between vertices not in V', then the hidden solution will be a *perfect code* in $G = (V, E)$. (A more precise definition of a perfect code will be given later.) This modified construction is useful for the Perfect Code public key cryptosystem described in Section 4.

Remark 2. In presenting the Dominating Set problem to children in El Salvador, the authors had to confront an example of the general problem of *cultural appropriateness* of the stories used to introduce these topics. We found that in El Salvador, as would be the case in many places in the world, the idea of minimizing the number of ice-cream stands makes no cultural sense whatsoever. So we changed the setting for the Dominating Set problem, presenting it by means of a story about minimizing the number of wells in order to achieve an efficient water supply for a village. Such a story is appropriate for a Third World context but would make no sense to children in the developed world.

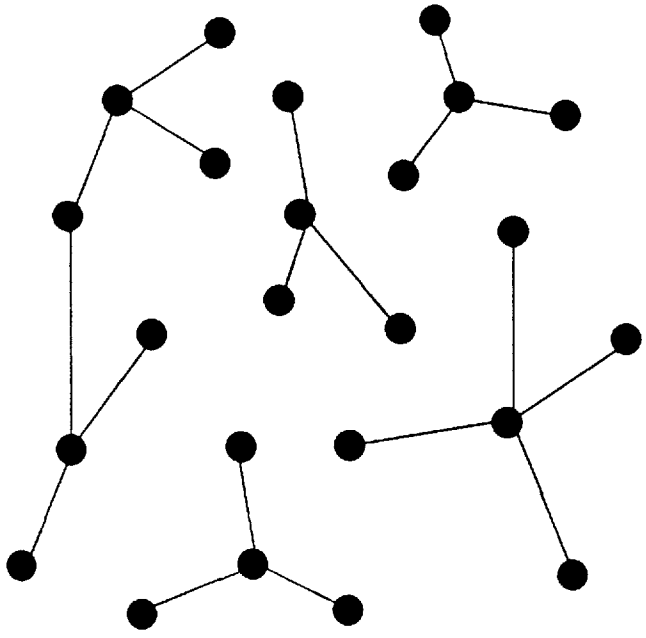

Fig. 4. The first step in the construction of Tourist Town: a configuration of stars

2.3 Information Hiding Protocols

A simple illustration of this is a method for computing the average allowance of children in the classroom, without revealing any individual's allowance. The protocol goes like this. The first person picks a large integer randomly, and adds to it her allowance. The sum is passed secretly to the second person, who adds to it her allowance, and so on. After all the allowances have been privately added in, the final sum is secretly passed by the last person to the first person, who subtracts her original secret large integer and computes the average for the group.

3 The Peruvian Coin Flip

One of the key issues we must face in designing crayon-technology cryptosystems is: what interesting functions can 7-year olds (for instance) compute reliably? That is, what sort of by-hand computing do we have available to work with?

With a little thought, we can see that interesting computations *can* be performed by young children to provide the computational engines for cryptosystems. For example, the outputs of Boolean circuits can be computed; finite-state automata and Mealy machines can be operated. (In principle, Turing machines can also be operated by paper and pencil, but our experience suggests that they are somewhat slow and unwieldy.) Cellular automata, if they are not too complicated, may offer another interesting possibility. Simple rewrite systems are

another candidate for accessible calculations. The following protocol is based on Boolean circuits.

This protocol was first demonstrated by the authors with children in Peru (hence the name). The idea of trying out a crayon-technology cryptosystem in Peru seemed natural for several reasons. In the first place, the improvement of mathematics education is currently a hot topic of discussion among educators in Peru, as in much of the Third World. In the second place, developing countries (and international science development organizations such as the *Kovalevskaia Fund*) have a special interest in the possibility of enhancing math and computer science education in situations where machines are not available.

We first told a story to explain how the need for such a coin-flip protocol might arise. The women's soccer teams of Lima and Cuzco have to decide who gets to be the home team for the championship game. Alicia, representing Lima, and Berta, representing Cuzco, cannot spend the time and money to get together to flip a coin. So they agree to the following arrangement.

Working together, they construct a Boolean circuit made up of and-gates and or-gates (for simplicity, we allow only small and-gates and or-gates, and no not-gates). See Figure 5 for an example. In the construction process, each has an interest in ensuring enough complexity of the circuit so that the other will be unable to cheat (see below). The final circuit is public knowledge. Let n be the number of input bits, and let m be the number of output bits.

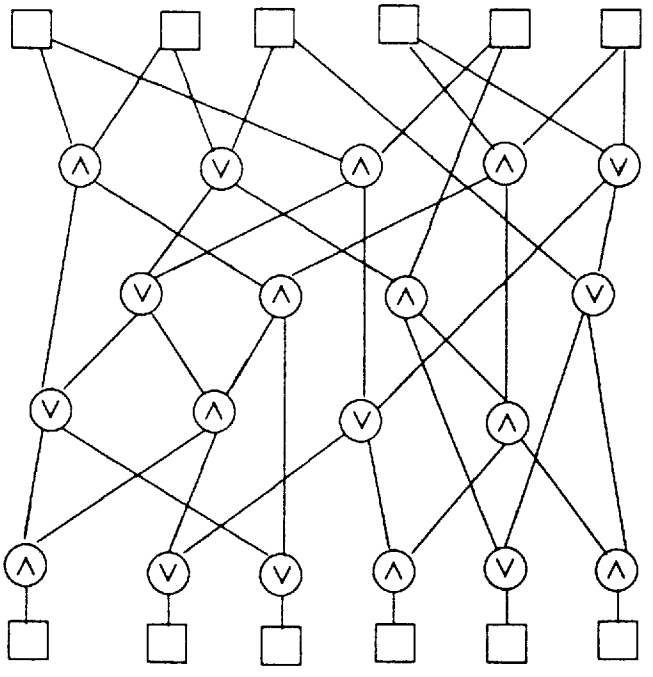

Fig. 5. A Boolean circuit for the Peruvian coin-flip

Alicia selects an arbitrary input string, which she keeps secret. She puts the string through the circuit, and sends Berta the output. Berta must then try to guess the *parity* of Alicia's input, i.e., the sum of its bits mod 2. If she guesses right, then the teams play in Cuzco. If her guess is wrong (which Alicia must demonstrate to her by revealing the input string), then they play in Lima.

Nothing in this description is hard to convey to a child of age 8 or above. Moreover, when we explain to the children the basic ingredient in the protocol ($\wedge$-gates and $\vee$-gates), we are talking about a really basic concept — perhaps *the* most basic concept — in formal logical thought. There is certainly as much justification for teaching about $\wedge$-gates and $\vee$-gates as for long division and addition of fractions!

Remark. An alternative construction would be for Alicia and Berta each to construct a circuit with n input bits and m output bits. Both circuits would be public knowledge. Then Alicia would put her secret input through both circuits, and the final output would be the XOR of the outputs produced by the two circuits. This variant is "cleaner" in the sense that it avoids some interaction; but probably the first variant is easier to explain to kids. More importantly, the first variant is more fun, precisely because of the added interaction.

3.1 Cheating

Berta can cheat if she can invert the circuit, i.e., find the input (or inputs) that produce a given output. Alicia can cheat if she can find two inputs of opposite parity that produce the same output. It seems likely that both forms of cheating are infeasible if the circuit is large and complex.

If the circuit maps many-to-one, we claim that the ability to cheat in Berta's role implies the ability to cheat in Alicia's role. Namely, we have

Proposition 1. *Suppose we have a family C of many-to-one Boolean circuits, with the property that for any output the proportion of inputs in its preimage of given parity (odd or even) is bounded from below. Further suppose that one has an algorithm that inverts any circuit of C in time bounded by $f(n)$, where n is the size of the circuit. Then in time bounded by $kf(n) + p(n)$ (where p is a polynomial and k is a security parameter) one can find two inputs of opposite parity that give the same output.*

Proof. This result — both the statement and the proof — is completely analogous to the result that the ability to take square roots modulo a composite number n implies the ability to factor n. Namely, to find the two desired inputs, select one input at random, and then apply the inversion algorithm to its output. With probability bounded from below, the inversion algorithm will give a second input of different parity for the same output. □

On the other hand, we can entirely prevent Alicia from being able to cheat by choosing a circuit that maps inputs to outputs injectively, i.e., it effects an

imbedding of $\{0,1\}^n$ into $\{0,1\}^m$. If we suppose that the circuit is complicated enough to behave like a random map, then the next proposition shows that it suffices to choose m somewhat larger than $2n$.

Proposition 2. *The probability that a random map from* $\{0,1\}^n$ *to* $\{0,1\}^m$ *is injective, is asymptotic to* $1 - 2^{-(m-2n+1)}$ *as* $m - 2n \longrightarrow \infty$.

Proof. This is a restatement of a well-known combinatorial result (the "birthday paradox"). □

4 Perfect Code Cryptosystems and Molten Arithmetic

The public key system which we will describe in this section can be designed with different levels of accessibility and security. The simplest version, which will be described first, can be mastered by a child who understands only (1) the simplest properties of graphs, and (2) addition (say, modulo 2 or modulo 26). We shall then describe a more complicated version, appropriate for older children, using what we call *molten arithmetic*. The latter term refers to the fluidity in the definition of the cryptosystem. That is, the rules for building the system can be adjusted according to the level of accessibility and security desired.

We begin by considering a special kind of dominating set in a graph called a *perfect code*. In what follows, if u is a vertex of a graph $G = (V, E)$, then the notation $N[u]$ (the "neighborhood" of u) denotes the set of vertices which share an edge with u (including u itself).

Definition 3. A set of vertices $V' \subseteq V$ in a graph $G = (V, E)$ is said to be a *perfect code* if for every vertex $u \in V$ the neighborhood $N[u]$ contains exactly one vertex of V'.

Figure 6 below shows an example of a graph with a perfect code. The vertices of the perfect code are indicated by open circles.

Remark 1. Jan Kratochvíl has shown that the problem of determining whether a graph has a perfect code is NP-complete for d-regular graphs, for all $d \geq 3$ [3].

Remark 2. An interesting detour for kids along the way to our cryptosystem might be to investigate error-correcting codes. For example, let n be of the form $2^k - 1$, and let G be the hypercube graph, whose vertices are $\{0,1\}^n \subset \mathbf{R}^n$ and whose edges are the edges of the n-dimensional unit hypercube. Then a *binary Hamming code* of length $n = 2^k - 1$ and dimension $d = 2^k - k - 1$ corresponds to a perfect code of 2^d vertices in G. For example, when $k = 2$, the (unique) Hamming code is the pair of opposite vertices (0,0,0) and (1,1,1) on the ordinary cube.

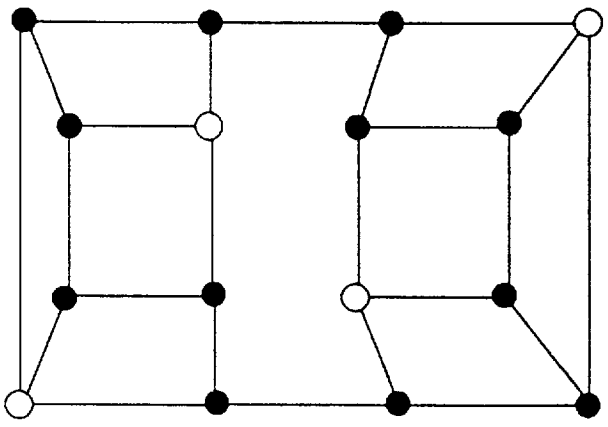

Fig. 6. Example of a perfect code in a cubic graph

4.1 Version A_1 of the Perfect Code Cryptosystem

This version is accessible to children of age 8. Suppose that the children have already mastered the Pre-Crypto topic *construction of a graph that has a well-disguised perfect code* (see the first remark of section 2.2). Now Alice wants to be able to receive an encrypted bit from Bobby. She constructs a graph $G = (V, E)$ with a perfect code V'. The graph G is her public key. Her private key is V'.

To send a bit b, Bobby makes a random assignment of 0's and 1's to all of the vertices of G except one. He then assigns either a 0 or 1 to the last vertex in such a way that the sum mod 2 over all of the vertices is equal to b. Next, he replaces the bit c_u assigned to each vertex u by a new bit c'_u determined by summing (mod 2) all of the bits that had been assigned to the neighboring vertices: $c'_u = \sum_{v \in N[u]} c_v$. He finally returns the graph to Alice with the bits c'_u annotating the vertices.

To decipher the message, Alice takes the sum of c'_u over the perfect code; that is, she has $b = \sum_{u \in V} c_u = \sum_{u \in V'} c'_u$, where the last equality follows from the definition of a perfect code.

4.2 Version A_2

The same as version A_1, but we make it more interesting by working modulo 26, so that Bobby can send Alice an enciphered letter $b \in \{A = 0, \ldots, Z = 25\}$.

Remark. Even if G is a complicated graph, both versions A_1 and A_2 of this cryptosystem can be broken in polynomial time using linear algebra (Gaussian elimination) modulo 2 (respectively, modulo 26). This will be shown later as a special case of a more general result. However, junior high school students have no more knowledge of how to do this than we have of how to factor integers in polynomial time. So with a judicious choice of G, versions A_1 and A_2 appear to be accessible and secure for junior high school.

4.3 Versions *B* and *C*

We now describe more elaborate versions which are harder to crack. We conjecture that version B is accessible and secure for high school students. Version C might be secure even for adults — at least we do not know how to break it.

First we need some notation and definitions. Given a graph $G = (V, E)$, we assign a variable denoted a_u to each vertex $u \in V$. Suppose that G has a perfect code V'. Let $x, y \in \mathbf{Z}$ and $m \in \mathbf{Z} \cup \{\infty\}$, $m \geq 2$. We let $\sigma(x, y, m)$ denote the substitution scheme which evaluates a polynomial in the a_u by setting $a_u = x$ if $u \in V'$ and $a_u = y$ otherwise, and performing the arithmetic modulo m (doing ordinary arithmetic in the case $m = \infty$).

By an *invariant expression* in a neighborhood $N[u]$ relative to the substitution scheme $\sigma(x, y, m)$ we mean a polynomial in the variables a_v, $v \in N[u]$, which evaluates to the same value irrespective of which vertex in $N[u]$ is in the perfect code. Here are some examples:

(1) For any substitution scheme, $\sum_{v \in N[u]} a_v$ is an invariant expression. More generally, any symmetric polynomial in the a_v, $v \in N[u]$, is an invariant expression relative to any substitution scheme.

(2) If the neighborhood $N[u]$ has 4 vertices, whose corresponding variables will be denoted a, b, c, d, and we have the substitution scheme $\sigma(2, 1, 3)$, then each of the following expressions is invariant: $ab + c + d$ (always evaluates to 1), $ab + ac + ad + a$ (always evaluates to 2), $ab + bc + cd + a + d$ (always evaluates to 1).

(3) If the neighborhood $N[u]$ has 4 vertices and we have the substitution scheme $\sigma(2, 1, \infty)$, then each of the following is invariant: $ab + c + d$, $ab + bc + cd + a + d$, $ab + cd$, $abc + d$.

We now describe versions B and C of the Perfect Code cryptosystem. In both cases the public key is the graph $G = (V, E)$ and the substitution scheme $\sigma(x, y, m)$ (i.e., a choice of x, y, m), the private key is the perfect code V', and the message Bobby wants to send is an integer b modulo m.

Version C is the most general. To send the message b, Bobby creates a large, complicated polynomial f from building blocks consisting of invariant expressions in neighborhoods of randomly selected vertices. This polynomial f must have two properties: (1) it evaluates to b under the substitution scheme $\sigma(x, y, m)$; and (2) someone who knows f but not how it was constructed from the building blocks would have great difficulty decomposing f into invariant expressions. Once Bobby constructs such an f, he sends it to Alice. Alice, who knows the perfect code, can correctly evaluate f without knowing how it decomposes into invariant expressions; she merely has to make the substitution $\sigma(x, y, m)$.

Version B is a special case of C. We use the substitution scheme $\sigma(1, 0, m)$ (m is arbitrary). To send the message b (a certain integer modulo m), Bobby chooses an arbitrary set I of subsets of vertices $S \subseteq V$ and a corresponding set

of integers c_S such that $\sum_{S \in I} c_S \equiv b \pmod{m}$. He then forms the polynomial

$$f = \sum_{S \in I} c_S \prod_{u \in S} \sum_{v \in N[u]} a_v.$$

Since each inner sum evaluates to 1, the whole expression obviously evaluates to $\sum c_S = b$.

Remark. Versions A_1 and A_2 are special cases of version B where I consists of one-element sets $S = \{u\}$. Then $f = \sum_{u \in V} c_u \sum_{v \in N[u]} a_v = \sum_{u \in V} c'_u a_u$, where $c'_u = \sum_{v \in N[u]} c_v$.

4.4 Breaking Versions A_1, A_2 and B

Given a polynomial f in the variables a_u, we want to find an identity of the form

$$f = \sum_{S \in I} c_S \prod_{u \in S} \sum_{v \in N[u]} a_v.$$

By writing both sides as a sum of homogeneous terms, without loss of generality we may assume that on the left side of the equation f is homogeneous of total degree d, and on the right side of the equation I is the set of subsets $S \subset V$ of cardinality d. We regard the c_S as unknowns, and equate coefficients of each monomial on the left and right. There are $\binom{n}{d}$ unknowns c_S (here $n = \#V$ is the size of the graph), and there are $\binom{n+d-1}{d}$ monomials of total degree d, and hence $\binom{n+d-1}{d}$ equations. Although there are more equations than unknowns (except in the case $d = 1$), we know that there is a solution, because the f in version B was constructed as such a sum of products. The solution can be found by Gaussian elimination. (In practice, the system of equations will probably be very sparse, in which case special methods are available.)

Notice that if d is unbounded, then the time required to do the linear algebra is not polynomial in the size n of the graph. However, the time is polynomial in the size of the polynomial f that Bobby sends to Alice, unless he has some way of producing sparse polynomials f (polynomials f with mostly zero coefficients). But we know of no way systematically to produce sparse polynomials that are difficult to crack.

Remark 1. Any time the substitution scheme in use is $\sigma(1, 0, m)$ (as in version B), there is a simple way that Bobby can make the cryptosystem harder to break. Bobby knows that any monomial whose variables are not all in Alice's perfect code will evaluate to 0, and hence can be dropped from f before he sends f to Alice. Of course, Bobby does not know the perfect code. However, he knows that if a monomial contains two variables a_u and a_v corresponding to vertices which are at a distance ≤ 2 from one another, then those vertices cannot both be in her perfect code, and hence the monomial can be dropped.

Remark 2. Notice that the f in version B are actually invariant under any substitution scheme $\sigma(x, y, m)$. The f used in version C are much more general, since they are built up from expressions which need only be invariant under our one particular substitution scheme. Thus, the f in version C cannot, in general, be decomposed into building blocks made of symmetric polynomials in the variables in a neighborhood.

Remark 3. In implementing these cryptosystems, the youngsters have to search for invariant building blocks and then build up complicated f, using the distributive law and gathering similar terms so as to disguise the way f was formed. In this way Kid Krypto might add some excitement to the subject of polynomials, which is often presented in school in a dry, unmotivated manner. The decision as to what version of Perfect Code cryptography to use — how complicated to make the possible f — depends on the age of the children and their ability to keep track of a lot of data.

4.5 Cubic Graphs

One way to keep the level of difficulty under control is to use only regular graphs of degree 3. Then the invariant expressions in any neighborhood involve exactly 4 variables. The class of cubic graphs is still plenty complicated to support these cryptosystems — as mentioned before, determining whether a given cubic graph has a perfect code is NP-complete. We now describe a simple construction that gives a large class of cubic graphs having perfect codes. The construction is based on covering spaces of K_4, the complete graph on 4 vertices.

The construction is as follows. Let $n = 4n_0$ be the size of the cubic graph to be constructed. Select four sets of n_0 vertices each, which we denote A, B, C, D. Then randomly create six one-to-one correspondences between the sets: $A \approx B$, $A \approx C$, $A \approx D$, $B \approx C$, $B \approx D$, $C \approx D$. Draw edges between vertices that are associated under any of these six bijections. Let $G = (V, E)$ be the resulting graph. Notice that each neighborhood $N[u]$ contains exactly one vertex from each of the sets A, B, C, D; and each of these sets is a perfect code in G. The construction is completely general: every covering space of K_4 can be produced in this way. It is not presently known whether the problem of recovering such a vertex set partition for a graph that is known to be a cover of K_4 is difficult in the sense of average-case complexity. The problem of deciding whether an arbitrary graph is a cover of K_4, however, has been shown to be NP-complete [3].

5 Kid Krypto Research Problems

The project of sharing the subject of cryptography with children leads to a number of interesting research problems.

5.1 Accessible Combinatorial Cryptosystems

Kid Krypto gives us a reason to have another look at various proposals for cryptosystems based on simple combinatorics. For example, the public key system using reversible cellular automata proposed in [2] may have merit for Kid Krypto. Another combinatorially based cryptosystem was proposed by a group of researchers at Madras Christian College in India and the Hanoi Mathematical Institute in Vietnam. In [5], they show that a rewrite system — based on the word problem in a group — can be used to construct a public-key system. It would be interesting to try to adapt these ideas for Kid Krypto.

It is worthwhile to develop a variety of examples of Kid Kryptosystems. In that way one can convey some of the richness and interconnectedness of mathematics, and at the same time give oneself flexibility when using Kid Krypto in the classroom.

5.2 Other Fundamental Protocols

At this point, a number of fundamental cryptographic primitives are still unexplored from the Kid Krypto point of view. Can we find elegant and accessible implementations of *oblivious transfer*, *secure 2-party computation*, *secret sharing*, *zero-knowledge proof*, etc.?

5.3 The Complexity of Small Things

Crayon-technology cryptosystems work with mathematical objects that are essentially very small, mathematically speaking — such as graphs on fewer than 25 vertices, circuits of similar size, and two-digit integers. From limited experience, it seems that it is relatively easy to generate small hard examples for the Minimum Dominating Set problem, but small hard examples of the 3-Coloring problem for planar graphs seem to be more difficult to generate. Is it possible to study this issue mathematically?

5.4 Breaking the Perfect Code Cryptosystem

Can the most general version of the Perfect Code system (version C) be broken in polynomial time?

5.5 Robustness Under Not Following Directions Properly

Classroom experiences seem inevitably to turn up intriguing questions in a playful vein. For example, the following question arose when the first author presented Map Coloring on one occasion. What is the minimum number of colors with which one can always color a planar map in a situation where one takes turns with an "incompetent helper" who is only assumed to color legally, but not necessarily judiciously? A bound of 33 was recently proved [4] for this lovely problem.

In presenting the Peruvian coin-flip to a junior high school audience, the authors encountered the situation where children attempted to evaluate the n-input/n-output circuit *upside down*. This leads to the following natural question, to which we do not know the answer. Let us suppose that all gates of our circuit have fan-out (as well as fan-in) of 2. (An alternative would be to allow large gates, i.e., gates with arbitrary fan-in and fan-out.) In addition, let us put $\vee$'s and $\wedge$'s in the input gates in an arbitrary way, with the understanding that such a gate (with a fan-in of 1) leaves the input bit unchanged. Under these assumptions the circuit makes sense if the child turns it upside down, of course with each $\vee$-gate now becoming a $\wedge$-gate and vice-versa. A natural question is whether it makes much difference (to a cheater) whether the circuit is right side up or upside down. More precisely, can one find a family of circuits which are easy to invert, but which when turned upside down are hard to invert? Can the problem of inverting the circuits in some presumably hard-to-invert family $\mathcal{C}$ be shown to be polynomial time equivalent to the problem of inverting the upside down circuits of $\mathcal{C}$?

5.6 Physical Realizations of Cryptographic Protocols

Some cryptographic ideas can be effectively demonstrated by employing physical props. Such demonstrations can be useful in conveying the central concepts of cryptography to children and other mathematically unsophisticated audiences, such as in popular lectures. Although in this paper we have focused on the design of cryptosystems accessible to children that are fully mathematical and do not rely on physical props, cryptosystems based on physical primitives might also prove to be a source of interesting mathematics for children.

For example, a number of fundamental protocols, such as oblivious transfer and multi-party secure computation, can be nicely demonstrated by means of ordinary playing cards [1]. Note that these familiar physical objects have a number of cryptographically useful properties "built in": they have a convenient means of randomization (shuffling), are uniquely identifiable, and when face down are all indistinguishable. A number of entertaining research problems arise in constructing cryptosystems based on such physical primitives (see [1] for further discussion).

6 The Research Community and Mathematics Education

The past year has seen the inception of at least three major projects originating in the research communities of mathematics and theoretical computer science to develop engaging mathematical materials for children in the elementary grades: (1) The education projects associated with the Center for Discrete Mathematics and Theoretical Computer Science (DIMACS), located at Rutgers University. (2) The compendium project of the Association for Computing Machinery Special Interest Group on Algorithms and Computation Theory (SIGACT).

(3) The Megamath Project of the U.S. National Laboratories in Los Alamos, New Mexico.

All three of these projects are concerned with developing resource materials from the extensive treasury of accessible, active and applicable mathematics that has emerged in recent years in the intertwined subjects of discrete mathematics and computer science.

DIMACS now publishes the newsletter *In Discrete Mathematics* (the premiere issue is dated Nov. 1991) which contains articles on topics in discrete mathematics intended to be useful to teachers introducing discrete mathematics to their classes. The newsletter will also serve as a networking service and clearinghouse for ideas and materials related to discrete mathematics in education in the lower grades. Further information can be obtained from Joe Rosenstein (joer@math.rutgers.edu). Members of the cryptographic community who wish to contribute something would certainly be welcome.

The SIGACT compendium project was initiated at the business meeting at STOC in May, 1992. The newly formed SIGACT Committee on Education has as its first goal the production of a compendium of theoretical computer science topics and presentation strategies that may be useful in a variety of settings with children (for example, children's science museums). This project makes no commitment to any particular direction in school curriculum reform; rather, it is simply a collective effort at science popularization.

The Megamath Project of the U.S. Los Alamos National Labs intends to influence classroom practice, by making schoolwork more like the experience one has in a good science museum. That is, the goal is to bring to children in the classroom a live experience of mathematical science as something in which they can actively participate. Thus, the Megamath Project is looking into such things as (1) mathematics research problems accessible to children, (2) possible forums for children to present the results of their mathematical investigations, (3) extended projects for classroom investigation, (4) the classroom use of personal mathematics journals, and (5) opportunities for children to communicate with larger mathematical communities.

The three initiatives in discrete mathematics and computer science described above join other efforts involving research scientists in elementary education. These include the Mathematicians and Education Reform Network sponsored by the AMS and the NSF, and the Scientists in the Schools program of the Sandia U.S. National Research Laboratories. Many scientists are now looking for more direct ways to work with children and stimulate grade school educational reform. This seems to be an idea whose time has come.

Besides fitting in well with these initiatives, a program of classroom activity centered around Kid Krypto is an ideal way to implement the Curriculum Standards of the National Council of Teachers of Mathematics [NCTM 1989]. These standards, which stress the importance of mathematical thinking, problem-solving, communication, and connections between mathematics and the world, are a radical departure from earlier curriculum standards. (In the past, the mathematical curriculum was defined simply to consist of a list of topics.)

Moreover, the idea of presenting the mathematics of computers (without machines!) has proved to be attractive to organizations interested in promoting opportunities for women and minorities in science and technology, particularly in situations where funds for education are severely limited. One of the sponsoring organizations of the Los Alamos Megamath Project is the *American Association of Historically Black Colleges*. The *Kovalevskaia Fund* (a foundation for women in science in developing countries) has organized lectures and demonstrations on discrete mathematics in the classroom (including Kid Krypto) at universities in the Third World.

We believe that the cryptographic community has an important role to play in the ambitious curriculum reform projects articulated by the NCTM and other organizations. Kid Krypto includes a tremendous wealth of vivid, accessible, applicable, engaging and active mathematics in its treasury of ideas. The involvement of cryptologists and theoretical computer scientists in elementary education will have several effects — first and foremost in helping to clarify what computer science is about. Like any science, it is about *ideas*; it is not a Cargo Cult.

One of the purposes of this paper is to encourage the reader to become involved in developing Kid Krypto and related materials for children. This can be done, for example, through the SIGACT compendium project. Contributions to this effort can be communicated to the first author. Even rough ideas in rough form are solicited, and will be credited in the compendium publication.

References

1. Crépeau, C., Kilian, J.: Discreet solitary games. Manuscript, April 1992.
2. Kari, J.: Cryptosystems based on reversible cellular automata. Manuscript, August 1992.
3. Kratochvíl, J.: Perfect codes in general graphs. Monograph, Czechoslovakian National Academy of Sciences, Prague, 1991.
4. Kierstead, H., Trotter, T.: Planar graph coloring with an uncooperative partner. Manuscript, April 1992.
5. Siromoney, R., Jeyanthi, A., Do Long Van, Subramanian, K. G.: Public key cryptosystems based on word problem. ICOMIDC Symposium on the Mathematics of Computation, Ho Chi Minh City, April 1988.