Efficient Reduction among Oblivious Transfer Protocols based on New Self-Intersecting Codes

Claude Crépeau * Miklós Sántha †

Laboratoire de Recherche en Informatique
Université Paris-Sud
Bâtiment 490
91405 Orsay France

Abstract

A $\binom{2}{1}$–OT_2 (one-out-of-two Bit Oblivious Transfer) is a technique by which a party $\mathcal{S}$ owning two secret bits b_0, b_1, can transfer one of them b_c to another party $\mathcal{R}$, who chooses c. This is done in a way that does not release any bias about $b_{\bar{c}}$ to $\mathcal{R}$ nor any bias about c to $\mathcal{S}$. One interesting extension of this transfer is the $\binom{2}{1}$–OT_2^k (one-out-of-two String O.T.) in which the two secrets q_0, q_1 are elements of $GF^k(2)$ instead of bits. A reduction of $\binom{2}{1}$–OT_2^k to $\binom{2}{1}$–OT_2 presented in [BCR86] uses $O(k^{\log_2 3})$ calls to $\binom{2}{1}$–OT_2 and thus raises an interesting combinatorial question: how many calls to $\binom{2}{1}$–OT_2 are necessary and sufficient to achieve a $\binom{2}{1}$–OT_2^k?

In the current paper we answer this question quite precisely. We accomplish this reduction using $\Theta(k)$ calls to $\binom{2}{1}$–OT_2. First, we show by probabilistic methods how to obtain such a reduction with probability essentially 1 and second, we give a deterministic polynomial time construction based on the algebraic codes of Goppa [Gop81].

1 Introduction

The equivalence between cryptographic primitives has recently become a major research topic as evidenced by many papers on the topic: [BCR86], [Cré88],[CK88],[Kil88],[Cré89]. So far a large number of cryptographic protocols have been shown equivalent to one another. Nevertheless very few of those reductions accomplish *perfectly* the task they are designed for. Generally the reduction will fail to achieve its goal with a small probability. An exception to this situation is the main contribution of [BCR86]. As an introduction to our work, we first summarize their result.

A $\binom{2}{1}$–OT_2 (one-out-of-two Bit Oblivious Transfer) is a technique by which a party $\mathcal{S}$ owning two secret bits b_0, b_1, can transfer one of them b_c to another party $\mathcal{R}$, who chooses c. This is done in a way that does not release any bias about $b_{\bar{c}}$ to $\mathcal{R}$ nor any bias about c to $\mathcal{S}$. This primitive was first introduced in [EGL83] with application to contract signing protocols. A natural and interesting extension of this transfer is the $\binom{2}{1}$–OT_2^k (one-out-of-two String Oblivious Transfer, know as ANNBP in [BCR86]) in which the two secrets q_0, q_1 are elements of $GF^k(2)$ instead of bits. The core of [BCR86] is a reduction of $\binom{2}{1}$–OT_2^k to $\binom{2}{1}$–OT_2, i.e. an efficient two-party protocol to achieve $\binom{2}{1}$–OT_2^k based on the assumption of the existence of a protocol for $\binom{2}{1}$–OT_2. The problem is solved using functions known as (n,k)-*zigzags* for historical reasons. They give an efficient construction for a binary linear

*Supported in part by an NSERC Postdoctorate Scholarship. e-mail: crepeau@dmi.ens.fr
†Part of this research was supported by an Alexander von Humboldt Fellowship. e-mail: santha@dmi.ens.fr

$(k^{\log_2 3}, k)$-zigzag. This was the best known construction of an (n,k)-zigzag so far. This construction raises an interesting combinatorial question: Minimize n as a function of k.

As pointed out in [BCR86], the problem of finding linear (n,k)-zigzags is equivalent to finding linear binary (n,k,d) codes with some special intersecting properties. Recently we have also found out that such codes had been extensively studied in the past, and are known to code theorists as *self-intersecting codes* [CL85]. Indeed the construction of the binary linear $(k^{\log_2 3}, k)$-zigzag of [BCR86] was known to coding theorists since 1984 by the work of Miklós [Mik84].

Denote by $zzl(n) = max\,\{k : \text{ there exists a binary linear } (n,k)\text{-zigzag}\}$. Katona and Srivastava [KS83] have shown that

n	$zzl(n)$
$3,4,5$	2
$6,7,8$	3
$9,10,11,12$	4
$13,14$	5

They also derived an upper bound on the asymptotic behavior of $\frac{zzl(n)}{n}$, whereas a lower bound was given by Komlós (reported in [Mik84]). These bounds are the following:

$$0.207 < \liminf_{n\to\infty} \frac{zzl(n)}{n} \leq \limsup_{n\to\infty} \frac{zzl(n)}{n} < 0.283.$$

They imply that binary linear (ck,k)-zigzags exist for $c > 4.81$ and clearly not for $c < 3.53$. Retter [Ret86] showed that most (classical) Goppa codes (see [MS77]) of rate less than 0.0817 are self-intersecting.

The current paper focuses on the polynomial time constructibility of such zigzags. First we show in section 3 that for $\alpha = \log_{4/3} 4$ and any $\epsilon > 0$, a random $k \times (1+\epsilon)\alpha k$ binary matrix (considered as a linear application) defines a $((1+\epsilon)\alpha k, k)$-zigzag with probability asymptotically 1, and a random $k \times (1-\epsilon)\alpha k$ binary matrix defines a $((1-\epsilon)\alpha k, k)$-zigzag with probability asymptotically 0. In both cases the convergence is exponentially fast in k. Actually the first part of this result was also stated by Cohen and Lempel [CL85]. This leads to an obvious efficient probabilistic algorithm for the generation of binary linear $((1+\epsilon)\alpha k, k)$-zigzags. Finally in section 4, we present a deterministic polynomial time construction for binary linear $(O(k), k)$-zigzags based on the algebraic codes of Goppa [Gop81].

2 Previous Work

Assume $\mathcal{S}$ and $\mathcal{R}$ have a mean of accomplishing $\binom{2}{1}$–OT_2 and that they wish to perform a $\binom{2}{1}$–OT_2^k over the two k-bit strings q_0, q_1. For any string x, let x^i denote the i^{th} bit of x. For a set of indices $I = \{i_1, i_2, ..., i_m\}$, we define x^I to be the concatenation $x^{i_1}x^{i_2}...x^{i_m}$, the indices taken in increasing order.

First observe that performing $\binom{2}{1}$–OT_2 on each pair q_0^i, q_1^i, $1 \leq i \leq k$, in order to implement a $\binom{2}{1}$–OT_2^k fails dramatically since a cheating $\mathcal{R}$ can get q_0^1 and q_1^2 for instance, i.e. partial information about q_0 **and** q_1. Assume instead that we find a function $f : GF^n(2) \to GF^k(2)$ with the nice property that for every two input strings x_0, x_1 and every disjoint sets I, J seeing the bits x_0^I and x_1^J releases information on at most one of $f(x_0)$ and $f(x_1)$. Let us be more precise about this.

Definition 2.1 A subset $I \subseteq \{1,2,...,n\}$ *biases* a function $f : GF^n(2) \to GF^k(2)$ if

$$\exists q_0, q_1, x \;\left[\#\{z | z^I = x^I,\, f(z) = q_0\} \neq \#\{z | z^I = x^I,\, f(z) = q_1\}\right] \tag{1}$$

such an x^I is said to *release information* about $f(x)$.

Definition 2.2 A (n,k)-*zigzag* is an easily computable function $f : GF^n(2) \rightarrow GF^k(2)$ for which random inverses can be easily computed for each input such that

$$\forall I \subseteq \{1,2,...,n\} \ \left[I \text{ or } \bar{I} \text{ does not bias } f \right] \tag{2}$$

The basic idea of [BCR86] is to find a zigzag f and use it as follows:

Protocol 2.1 ($\binom{2}{1}$–$\mathbf{OT}_2^k((q_0,q_1),c)$)

1: *$\mathcal{S}$ finds random x_0, x_1 such that $f(x_0) = q_0$ and $f(x_1) = q_1$.*

2: $\underset{i=1}{\overset{n}{\boldsymbol{DO}}}$ *$\mathcal{S}$ transfers x_c^i out-of x_0^i, x_1^i to $\mathcal{R}$ using $\binom{2}{1}$–OT_2.*

3: *$\mathcal{R}$ recovers q_c by computing $q_c = f(x_c)$.*

This protocol works since an honest $\mathcal{R}$ gets $x_c^{\{1,\ldots,n\}}$ and $x_{\bar{c}}^{\emptyset}$, leading him to q_c and no information about $q_{\bar{c}}$, while by the definition of f, a cheating $\mathcal{R}$ who gets x_0^I and $x_1^{\bar{I}}$ for some $I \neq \emptyset \neq \bar{I}$, only gets information about one of $f(x_0)$ or $f(x_1)$. Based on this protocol, our task is to build efficient f's.

Working under the assumption that $f(x)$ is a linear binary function, i.e. that $f(x) = Mx$ for a given $k \times n$ binary matrix M (we say that M *defines* f), [BCR86] shows that the characterization (2) of f

$$\forall I \subseteq \{1,2,...,n\} \ \left[I \text{ or } \bar{I} \text{ does not bias } Mx \right] \tag{3}$$

is equivalent to the two following characterizations (4) and (5).

Define for a matrix M, M^i to be the i^{th} column of M. For $I = \{i_1, i_2, ..., i_m\}$, we define M^I to be the matrix obtained by the concatenation $M^{i_1}M^{i_2}...M^{i_m}$. We say that two line vectors $\vec{v}_0$ and $\vec{v}_1$ of dimension n *intersect* if $\vec{v}_0 \times \vec{v}_1 \neq (0,0,...,0) = \vec{0^n}$, where the product $\times$ is performed component by component.

Proposition 2.3 *M defines a linear zigzag if and only if*

$$\forall I \subseteq \{1,2,...,n\} \ \left[M^I \textit{ or } M^{\bar{I}} \textit{ has rank } k \right] \tag{4}$$

or

$$\forall \vec{a}, \vec{b} \in GF^k(2) - \left\{ \vec{0^k} \right\} \ \left[\vec{a}M \textit{ and } \vec{b}M \textit{ intersect} \right]. \tag{5}$$

Observations & Remarks

This last characterization turns out to be the most useful. Consider M as the generator matrix of a binary linear code. M defines an (n,k)-zigzag exactly if the (n,k,d) linear code generated by M does not contain two non-intersecting codewords $\vec{c_1}, \vec{c_2}$. A number of observations can be made about the code generated by M. The most important are:

Observation 2.1: If M defines a linear (n,k)-zigzag then it defines a linear (n,k,k) code. (see [BCR86])

Observation 2.2: If M defines a linear $(n,k,\frac{n}{2}+1)$ code then it defines a linear (n,k)-zigzag. By a pigeon hole argument any two codewords of such a code must intersect.

Remark 2.1: Observation 2.2 combined with Plotkin's bound [MS77] implies that linear $(n,k,\frac{n}{2}+1)$ codes cannot exist over $GF(2)$ when $n \in O(k)$. Nevertheless this observation will be crucial to the elaboration of the deterministic technique of section 4.

3 Random matrices for linear zigzags

In this section we determine very precisely the size of random matrices which define linear zigzags. If k denotes the number of lines, then there is a threshold function $t(k)$, which is linear in k, such that a random matrix with more than $t(k)$ columns defines a linear zigzag with high probability, whereas a random matrix with less than $t(k)$ columns does not define a linear zigzag, also with high probability. Komlós has already observed [Mik84] that some matrices with more than $t(k)$ columns define a linear zigzag.

Theorem 3.1 *Set $\alpha = \log_{4/3} 4 \approx 4.81$, and let M be a random $k \times n$ binary matrix. Then for every constant $\epsilon > 0$, there exists a constant $0 < \eta < 1$ such that we have the following two propositions:*

1. *If $n \geq (1+\epsilon)\alpha k$, then $Pr[M\ does\ not\ define\ a\ linear\ zigzag] < \eta^k$,*
2. *If $n \leq (1-\epsilon)\alpha k$, then $Pr[M\ defines\ a\ linear\ zigzag] < \eta^k$.*

Sketch of Proof: Let $\vec{a}$ and $\vec{b}$ be two non-zero binary line vectors of length k, and set $S = (\vec{a}, \vec{b})$. For every S, let D_S be the event that $\vec{a}M$ and $\vec{b}M$ do not intersect, and let X_S be the associated indicator random variable. As every bit in $\vec{a}M$ and $\vec{b}M$ is 1 with probability 1/2, we have

$$E[X_S] = Pr[D_S] = \begin{cases} (\frac{3}{4})^n & \text{if } \vec{a} \neq \vec{b} \\ (\frac{1}{2})^n & \text{if } \vec{a} = \vec{b}. \end{cases}$$

Let us define the random variable $X = \sum_S X_S$, where the summation is done over all couples $S = (\vec{a}, \vec{b})$. Then M defines a linear zigzag if and only if $X = 0$. By the linearity of the expectation, $E[X] \approx 2^{2k}(\frac{3}{4})^n$.

Proposition *1.*: $n \geq (1+\epsilon)\alpha k$. Then we have:

$$Pr[X > 0] \leq E[X] \approx 2^{2k}\left(\frac{3}{4}\right)^n \leq \left(\frac{1}{4}\right)^{\epsilon k}.$$

Proposition *2.*: $n \leq (1-\epsilon)\alpha k$. Then using Chebishev's inequality, we have:

$$Pr[X = 0] \leq Pr[|X - E[X]| \geq E[X]] \leq \frac{Var(X)}{E[X]^2}.$$

We will show that $Var(X)$ is exponentially small compared to $E[X]^2$. We set $N = E[X]^2$. Let $\vec{a_1}, \vec{b_1}, \vec{a_2}$ and $\vec{b_2}$ be non-zero line vectors of length k, and set $S_1 = (\vec{a_1}, \vec{b_1}), S_2 = (\vec{a_2}, \vec{b_2})$. Since X_{S_1} and X_{S_2} are 0-1 random variables, we have:

$$Var(X) = \sum_{S_1,S_2} cov(X_{S_1}, X_{S_2}) = \sum_{S_1,S_2} E[X_{S_1}](E[X_{S_2}|X_{S_1} = 1] - E[X_{S_2}]).$$

If the random variables X_{S_2} and X_{S_1} are not independent, then we will bound $cov(X_{S_1}, X_{S_2})$ from above by $E[X_{S_1}]E[X_{S_2}|X_{S_1} = 1]$. If X_{S_2} and X_{S_1} are independent, then $cov(X_{S_1}, X_{S_2}) = 0$. The proof works out because for most S_1 and S_2 they are indeed independent. We will prove this with the help of the following Lemma:

Lemma: *If for some $m > 0$, the family of vectors $\{\vec{a_1}, \ldots \vec{a_m}\}$ is linearly independent then $\vec{a_1}M, \ldots, \vec{a_m}M$ are independent random variables.*
Proof of Lemma: It follows from Vazirani's Parity Lemma [Vaz87] when applied to the uniform

distribution. $\square$

It follows from the Lemma that the random variables X_{S_1} and X_{S_2} are independent when the family of four vectors $\{\vec{a_1}, \vec{b_1}, \vec{a_2}, \vec{b_2}\}$ is linearly independent. Thus we have to consider the covariances of only those random variables X_{S_1} and X_{S_2} where there is some linear dependence among the (not necessarily distinct) vectors of the family $\mathcal{U} = \{\vec{a_1}, \vec{b_1}, \vec{a_2}, \vec{b_2}\}$. We will distinguish several cases according to the rank of $\mathcal{U}$ and the possible dependencies. The analysis of all the cases is similar, we analyze here two of them.

Case 1: $rank(\mathcal{U}) = 3$ and $\vec{a_2} = \vec{a_1}$. The number of such families is less than 2^{3k}. Let us suppose that $\vec{a_1}M$ and $\vec{b_1}M$ do not intersect and let us compute the probability that $\vec{a_2}M$ and $\vec{b_2}M$ do not intersect either. Let a_1 and b_1 be the first bit of respectively $\vec{a_1}M$ and $\vec{b_1}M$. Similarly, let a_2 and b_2 be the first bit of $\vec{a_2}M$ and $\vec{b_2}M$. The string a_1b_1 with probability $\frac{1}{3}$ takes each of the values 00, 01 and 10, since $\vec{a_1}M$ and $\vec{b_1}M$ do not intersect. Our hypotheses about the family $\mathcal{U}$ imply that $a_2 = a_1$ and b_2 is 1 with probability $\frac{1}{2}$, independently from the value of a_1b_1. Thus we have

$$Pr[a_2b_2 = 11 | a_1b_1 \neq 11] = \frac{1}{3} \times 0 + \frac{1}{3} \times 0 + \frac{1}{3} \times \frac{1}{2} = \frac{1}{6}.$$

The analysis is identical in the other columns of the matrix, therefore the total contribution to the variance of these families is less than

$$2^{3k} \left(\frac{3}{4}\right)^n \left(\frac{5}{6}\right)^n \leq (0.83)^k N.$$

Case 2: $rank(\mathcal{U}) = 3$ and $\vec{a_1} + \vec{a_2} + \vec{b_1} + \vec{b_2} = \vec{0}$. The number of such families is once again less than 2^{3k}. We will use the same notation as previously. Then $a_2 = a_1 + z$ and $b_2 = b_1 + z$, where $z = a_1 + a_2 = b_1 + b_2$ is a random bit. Clearly, $a_2b_2 = 11$ if and only if $a_1b_1 = 00$ and $z = 1$. This happens with probability $\frac{1}{3} \times \frac{1}{2} = \frac{1}{6}$, and the total contribution of these families is less than $(0.83)^k N$. $\square$

4 Deterministic Linear Size Construction

Although the result of section 3 implies that binary linear $(O(k), k)$-zigzags exist and can be obtained easily by picking one at random, it does not provide an efficient way of building a *guaranteed* zigzag of linear size. The problem of checking if a random matrix defines a zigzag seems rather hard (probably as hard as finding the minimal weight of a code, which is considered a hard problem [BET78]).

The solution we present now, is based on a new approach: using the theory of linear codes over $GF(q)$, for $q > 2$, in order to perform the reduction of $\binom{2}{1}$–OT_q^k to $\binom{2}{1}$–OT_q in the larger field $GF(q)$ ($\binom{2}{1}$–OT_q is the q-ary version of $\binom{2}{1}$–OT_2). We then use the construction of [BCR86]/[Mik84] to perform the reduction of $\binom{2}{1}$–OT_q to $\binom{2}{1}$–OT_2.

4.1 Reduction over $GF(q)$

Assume $\mathcal{S}$ and $\mathcal{R}$ have a mean of accomplishing $\binom{2}{1}$–OT_q for some fixed $q > 2$ and that they wish to perform a $\binom{2}{1}$–OT_q^k with the two (q-ary) strings q_0, q_1 of length k. As before, q_x^i defines the i^{th} (q-ary) entry of q_x. In the same manner, for $I \subseteq \{1, 2, ..., n\}$ we define q_x^I as before but with respect to q-ary strings instead of binary. We also say that a subset I *biases* a function $f : GF^n(q) \to GF^k(q)$ exactly if the condition (1) is satisfied as in the case $q = 2$.

The basic idea of [BCR86], to find a function f with property (2) remains our approach to performing the reduction of $\binom{2}{1}$–OT_q^k to $\binom{2}{1}$–OT_q. Based on the assumption that f is a linear function (with operations in $GF(q)$ instead of $GF(2)$) it is possible to show that the characterizations

(4) and (5) are also equivalent to (3) when interpreted over $GF(q)$ (the details are left out of this abstract). Therefore, in order to solve the problem over $GF(q)$, one possible approach is to find linear $(n, \Theta(n), \frac{n}{2}+1)$ codes over $GF(q)$. Due to the work of Goppa [Gop81], Tsfasman, Vlădut and Zink [TVZ82], Katsman, Tsfasman and Vlădut [KTV84] and Manin and Vlădut [MV84] we actually get the codes that we need over $GF(q)$ where q is greater than 9 and is an even power of a prime.

More precisely, their result is as follows:

Proposition 4.1 ([MV84]) *If $q = p^{2m}$ for some prime p, it is possible to construct in polynomial time linear (n, k, d) codes over $GF(q)$ with parameters spreading over the line of equation*

$$\frac{k}{n} + \frac{d}{n} = 1 - \frac{1}{\sqrt{q}-1}.$$

We use these codes to achieve our goal as follows.

4.2 Construction of a Linear zigzag over $GF(q)$

Let $q = p^{2m} > 9$ for some prime p, and let $\Gamma_q = \frac{1}{2} - \frac{1}{\sqrt{q}-1}$. According to proposition 4.1 it is possible to build in polynomial time a linear $(n, \Gamma_q n - 1, \frac{n}{2}+1)$ code over $GF(q)$ for all sufficiently large n. This leads to the construction of a linear $(n, \Gamma_q n - 1)$-zigzag over $GF(q)$, i.e.

Proposition 4.2 *If $q = p^{2m} > 9$ for some prime p, it is possible to build in polynomial time a linear $(\frac{k+1}{\Gamma_q}, k)$-zigzag over $GF(q)$.*

Remark 4.1: An analysis similar to that of section 3 for the case over $GF(q)$ shows that for any $\epsilon > 0$, a random $GF(q)$-matrix of size $k \times (2+\epsilon)k$ has asymptotically probability 1 of being a zigzag, as $q \to \infty$.

4.3 Construction of a Linear zigzag over $GF(2)$

Using the previous technique it is possible, for $m > 1$, to build a $\binom{2}{1}$–$\mathrm{OT}^k_{2^{2m}}$ based on the existence of a $\binom{2}{1}$–$\mathrm{OT}_{2^{2m}}$. First observe that a $\binom{2}{1}$–$\mathrm{OT}_{2^{2m}}$ and a $\binom{2}{1}$–OT^{2m}_2 are interchangeable since the base sets of the two structures are the same. On the other hand, building a $\binom{2}{1}$–OT^{2m}_2 is very simple using the general construction of [BCR86]. Their technique is used purely to bootstrap our more efficient technique and is used in an ad-hoc fashion to solve the problem for this fixed length $2m$ over $GF(2)$. This technique is nothing else than building a concatenation code from the Goppa code and a fixed smaller binary linear code. The concatenation of these two codes will be a linear binary code.

Example 4.1 Set $q = 256$. Let q_0, q_1 be two binary strings of length k. Consider each of these two strings as $\frac{k}{8}$ bytes (i.e. elements of $GF(256)$). By proposition 4.2 one can build a linear $(\frac{k}{3}, \frac{k}{8})$-zigzag over $GF(256)$. Let M be the byte-matrix of size $\frac{k}{8} \times \frac{k}{3}$ that defines such a zigzag. Let z_0, z_1 be two random byte-strings of length $\frac{k}{3}$ such that $q_0 = Mz_0$ and $q_1 = Mz_1$ where the operations $+$ and $\times$ of $GF(256)$ are performed on bytes. Now let $\mathcal{M}$ be the 8×27 binary-matrix that defines the binary linear $(27, 8)$-zigzag of [BCR86]. For each byte z_0^i of z_0 and z_1^i of z_1, let y_0^i and y_1^i be random binary strings of length 27 such that $z_0^i = \mathcal{M}y_0^i$ and $z_1^i = \mathcal{M}y_1^i$ with the operations performed in $GF(2)$. Name $y_0 = y_0^1 y_0^2 ... y_0^{\frac{k}{3}}$ and $y_1 = y_1^1 y_1^2 ... y_1^{\frac{k}{3}}$ the two resulting binary strings of length $9k$.

Define $f : GF^{9k}(2) \to GF^k(2)$ to be

$$f(x) = G\mathcal{G}x$$

where $\mathcal{G}$ is the $\frac{8k}{3} \times 9k$ binary matrix defined from $\mathcal{M}$ as follows:

$$\mathcal{G} = \begin{pmatrix} \mathcal{M} & \begin{matrix} 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{matrix} & \dots & \begin{matrix} 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{matrix} \\ \begin{matrix} 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{matrix} & \mathcal{M} & \dots & \begin{matrix} 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{matrix} \\ \vdots & \vdots & \ddots & \vdots \\ \begin{matrix} 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{matrix} & \begin{matrix} 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{matrix} & \dots & \mathcal{M} \end{pmatrix}$$

and G is the $k \times \frac{8k}{3}$ binary matrix constructed by replacing each line M_i of M ($\frac{k}{3}$ bytes per line) by 8 lines of $\frac{8k}{3}$ bits obtained by multiplying (in $GF(256)$) each component of M_i with the byte constants 00000001, 00000010, ..., 10000000 and interpreting the result as binary strings, i.e.

$$G = \begin{pmatrix} M_1 \\ 00000010 \times M_1 \\ \vdots \\ 10000000 \times M_1 \\ \vdots \\ M_{\frac{k}{8}} \\ 00000010 \times M_{\frac{k}{8}} \\ \vdots \\ 10000000 \times M_{\frac{k}{8}} \end{pmatrix}$$

We then have that $q_0 = f(y_0)$, $q_1 = f(y_1)$ and the following proposition:

Proposition 4.3 *The above f is a binary linear $(9k, k)$-zigzag.*

Similar constructions can be obtained for any large enough field. For larger fields more efficient techniques than that of [BCR86]/[Mik84] should be used. Nevertheless, if their technique is used to bootstrap the process, then the field $GF(256)$ will lead to the family of zigzags with smallest expansion factor.

5 Conclusion and Open Problems

We have shown how to construct binary linear size zigzag functions both by probabilistic and deterministic polynomial time methods. The exact complexity of the decision problem "Given a matrix M, is it the generator of a zigzag?" still has to be determined (the best we can say is that it is in Co-NP). Another problem still open is to construct deterministically over $GF(q)$ some zigzag that will do better than the asymptotic bounds of section 3 and remark 4.1 . Finally we ask if non-linear functions can generate smaller zigzags than linear functions.

Acknowledgments

We would like to thank Paul Barbaroux, Gilles Brassard, Johannes Buchmann, W. Fernandez De La Vega, Antoine Joux, Joe Kilian, Jean-Marc Robert, and Jacques Stern for their help, comments, and support. Special thanks to Oded Goldreich and Silvio Micali for suggesting characterization (4) (c.f. [BCR86]). Claude wishes to thank Shafi Goldwasser for introducing him to Goppa codes, Peter Elias for providing an extensive bibliography on the topic and Gérard Cohen for extremely valuable information about his earlier research on intersecting codes. We are very greatful to Ivan Damgård for pointing out to us that our $(9k, k)$-zigzag was indeed linear.

References

[BCR86] G. Brassard, C. Crépeau, and J.–M. Robert. Information theoretic reductions among disclosure problems. In 27^{th} *Symp. of Found. of Computer Sci.*, pages 168–173, IEEE, 1986.

[BET78] E.R. Berlekamp, R.J. Mc Eliece, and H.C.A. Van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transaction on Information Theory*, 384–386, 1978.

[CK88] C. Crépeau and J. Kilian. Achieving oblivious transfer using weakened security assumptions. In 28^{th} *Symp. on Found. of Computer Sci.*, pages 42–52, IEEE, 1988.

[CL85] G. Cohen and A. Lempel. Linear intersecting codes. *Discrete Mathematics*, 56:35–43, 1985.

[Cré88] C. Crépeau. Equivalence between two flavours of oblivious transfers (abstract). In C. Pomerance, editor, *Advances in Cryptology: Proceedings of Crypto '87*, pages 350–354, Springer-Verlag, 1988.

[Cré89] C. Crépeau. Verifiable disclosure of secrets and application. In *Advances in Cryptology: Proceedings of Eurocrypt '89*, Springer-Verlag, 1989.

[EGL83] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. In R. L. Rivest, A. Sherman, and D. Chaum, editors, *Proceedings CRYPTO 82*, pages 205–210, Plenum Press, New York, 1983.

[Gop81] V.D. Goppa. Codes on algebraic curves. *Soviet Mathematical Dokl.*, 24(1):170–172, 1981.

[Kil88] J. Kilian. Founding cryptography on oblivious transfer. In *Proc. 20th ACM Symposium on Theory of Computing*, pages 20–31, ACM, Chicago, 1988.

[KS83] G. Katona and J. Srivastava. Minimal 2-coverings of finite affine spaces based on $GF(2)$. *Journal of Statist. Plann. Inference*, 8:375–388, 1983.

[KTV84] G.L. Katsman, M.A. Tsfasman, and S.G. Vlădut. Modular curves and codes with a polynomial construction. *IEEE Transaction on Information Theory*, IT-30(2):353–355, 1984.

[Mik84] D. Miklós. Linear binary codes with intersecting properties. *Discrete Applied Mathematics*, 9(2):187–196, 1984.

[MS77] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977.

[MV84] Yu.I. Manin and S.G. Vlădut. Linear codes and modular curves (in Russian). *Sovr. Prob. Mat.*, VINITI, 1984.

[Ret86] C.T. Retter. Intersecting goppa codes. 1986. manuscript.

[TVZ82] M.A. Tsfasman, S.G. Vlădut, and Th. Zink. Modular curves, Shimura curves, and Goppa codes, better than Varshamov-Gilbert bound. *Math. Nachr.*, 109:21–28, 1982.

[Vaz87] U. Vazirani. Efficiency considerations in using semi-random sources. In *Proc. 19th ACM Symposium on Theory of Computing*, pages 160–168, ACM, New York City, 1987.