# Separable Hash Functions

Sarang Aravamuthan

Ignite R&D Labs, Tata Consultancy Services, Chennai, India.

E-mail: sarang.aravamuthan@tcs.com

**Abstract.** We introduce a class of hash functions with the property that messages with the same hash are well separated in terms of their Hamming distance. We provide an example of such a function that uses cyclic codes and an elliptic curve group over a finite field.

A related problem is ensuring that the *consecutive distance* between messages with the same hash is as large as possible. We derive bounds on the c.d. separability factor of such hash functions.

**Keywords.** hash functions, separability, algebraic codes.

## 1 Introduction

In recent times, the security of the hashing algorithm SHA-1 has been brought under scrutiny following announcements by a group of researchers on successful attacks to uncover messages with the same hash [10, 11]. This is a critical issue for digital signature algorithms as their security against forgery depends on the robustness of the hash function against such attacks.

Given that hash functions are inherently many-to-one maps, it's inevitable that several messages will hash to the same value. A desirable requirement is that such messages be "well-separated" in terms of their Hamming distance. This prevents an attacker from attempting to find another message with the same hash by tweaking just a few bits of the original message.

We capture this notion through the concept of "separable hash functions". We show how one could construct such functions using algebraic codes and one-way permutations. We illustrate these concepts with a construction using a cyclic code and an elliptic curve group over a prime field.

We further introduce the notion of *consecutive distance* between messages. This is the minimum number of consecutive bits that may need to be changed in a message in order to derive another with the same hash.

The idea of consecutive distance captures practical scenarios where an attacker would want to change a few consecutive bits to derive another nearly identical message with the same hash. Bounds on the value of $t$ for a *$t$-c.d. separable* hash function are derived.

## 1.1 Preliminaries

Let $\mathcal{H}_N = \{0,1\}^N$ be the *Hamming space* of all binary vectors of length $N$. Addition of vectors in $\mathcal{H}_N$ is a component-wise x-or operation. The vector of all zeros is indicated by $\mathbf{0}$. The (Hamming) *distance* between two vectors $\mathbf{x}$ and $\mathbf{y}$, denoted $d(\mathbf{x},\mathbf{y})$, is the number of co-ordinates they differ in (we adopt the convention of using boldface font for vectors and non-boldface for scalars). The *weight* of a vector $\mathbf{x}$, $\mathrm{wt}(\mathbf{x})$, is the number of 1's in $\mathbf{x}$. One sees easily that $d(\mathbf{x},\mathbf{y}) = \mathrm{wt}(\mathbf{x}+\mathbf{y})$.

Let $\mathcal{H}_n$ and $\mathcal{H}_m$ be the *message space* and *hash space* respectively where $n$ is the message length and $m$ the hash length.

A *code* $\mathcal{C}$ is any non-empty subset of $\mathcal{H}_n$. It's elements are called *codewords*. The *size* of $\mathcal{C}$ is the number of codewords in $\mathcal{C}$. $n$ is the *length* of $\mathcal{C}$. The *minimum distance* of the code, denoted $d(\mathcal{C})$, is the minimum distance between all distinct pairs of codewords in $\mathcal{C}$; see [2] for a detailed background on coding theory.

An $\langle n,k\rangle$-code is a code of length $n$ and minimum distance $k$. A *maximal* $\langle n,k\rangle$-code is one that is not contained in any other $\langle n,k\rangle$-code. Unless specified otherwise, all codes introduced in this paper will be assumed to be maximal.

The *translate* of an $\langle n,k\rangle$-code $\mathcal{C}$ by a vector $\mathbf{y}$ is another $\langle n,k\rangle$-code

$$\mathcal{C}(\mathbf{y}) := \{\mathbf{x}+\mathbf{y} : \mathbf{x} \in \mathcal{C}\}.$$

Informally, $\mathcal{C}(\mathbf{y})$ is "$\mathcal{C}$ shifted by $\mathbf{y}$".

Let $B_n(\mathbf{x},R)$ be the *ball* of radius $R$ centred at $\mathbf{x} \in \mathcal{H}_n$, i.e.

$$B_n(\mathbf{x},R) = \{\mathbf{y} \in \mathcal{H}_n : d(\mathbf{x},\mathbf{y}) \le R\}.$$

The *volume* of this ball, $V_n(R)$, is the size of $B_n(\mathbf{x},R)$ and is independent of $\mathbf{x}$. Specifically

$$V_n(R) = \sum_{i=0}^{R} \binom{n}{i}.$$

We can bound $V_n(R)$ from above and below by

$$\left(\frac{n}{R}\right)^R \leq \binom{n}{R} \leq V_n(R) \leq \left(\frac{ne}{R}\right)^R \tag{1}$$

The upper bound is known as Sauer's Lemma and is a well known combinatorial identity (for a specific reference, see [9, Lemma 4.3]).

The *covering radius* of a code $\mathcal{C} \subseteq \mathcal{H}_n$ is the smallest integer $s$ such that every vector in $\mathcal{H}_n$ is within distance $s$ of some codeword in $\mathcal{C}$(see [2]). We observe that taking union of the balls of radius $s$ around each codeword in $\mathcal{C}$ covers all of $\mathcal{H}_n$, i.e.

$$\bigcup_{\mathbf{x} \in \mathcal{C}} B_n(\mathbf{x}, s) = \mathcal{H}_n. \tag{2}$$

It can be shown that the covering radius $s$ of a maximal $\langle n, t\rangle$-code satifies

$$\left\lfloor \frac{t}{2} \right\rfloor \leq s < t$$

## The Separable Hash Problem:

Given $m$ and $t$, determine $n$ and a hash function

$$f_t : \mathcal{H}_n \to \mathcal{H}_m$$

with the following properties

(i) *preimage resistance*: Given a hash value $\mathbf{y} \in \mathcal{H}_m$, determining a message $\mathbf{x} \in \mathcal{H}_n$ such that $f_t(\mathbf{x}) = \mathbf{y}$ is an intractable problem.

(ii) *2nd preimage resistance*: Given a message $\mathbf{x}$ with hash value $f_t(\mathbf{x}) = \mathbf{y}$, determining a different message with the same hash value is intractable.

(iii) *collision resistance*: Finding any two distinct messages $\mathbf{m}_1$ and $\mathbf{m}_2$ with $f_t(\mathbf{m}_1) = f_t(\mathbf{m}_2)$ is intractable.

(iv) *t-separability*: Any two messages with the same hash value are at least a distance $t$ apart.

We will call $t$, the *separability factor*. We note that properties (i), (ii) and (iii) are in increasing order of hardness. Thus, barring a few pathological cases (see [6, 9.20] for an example), collision resistance always implies preimage resistance$^\star$.

$^\star$ To see that this is usually true, take a random message and find its hash. It's very likely that the preimage of the hash is different from the message. So, if the preimage resistance property fails, then collision resistance must fail as well.

## 2 Satisfying $t$-separability through algebraic codes

The $t$-separability property implies that the collection of all messages with the same hash forms an $\langle n, t\rangle$-code. Thus one way of realizing a $t$-separable hash function would be to

– Partition $\mathcal{H}_n$ into $\langle n, t\rangle$-codes.
– Map each code to a distinct element of $\mathcal{H}_m$.

The number of codes in the partition must be at most $2^m$ to realize the map. This bounds the value of $m$.

### 2.1 Bounds for $m$ in terms of $t$ and $n$

**Lemma 1.** *If $f_t : \mathcal{H}_n \to \mathcal{H}_m$ is a $t$-separable function, then $m$ is bounded as*

$$V_n(t/2) \le 2^m \le V_n(t-1). \tag{3}$$

*Proof.* To see the lower bound, consider the ball $B_n(\mathbf{0}, t/2)$ of all vectors of weight $\le t/2$. We observe that any two vectors in this ball are within distance $t$ of each other and thus map to distinct hashes.

The upper bound is proved by constructing an explicit $t$-separable function. Let $\mathcal{C}$ be a maximal $\langle n, t\rangle$-code of covering radius $s < t$. By (2), $\mathcal{H}_n$ is covered by the union of balls of radius $s$ around each codeword.

Choose a codeword $\mathbf{x} \in \mathcal{C}$ and consider the collection of translates of $\mathcal{C}$

$$\mathbb{X} := \{\mathcal{C}(\mathbf{y}) : d(\mathbf{x}, \mathbf{y}) \le s\}. \tag{4}$$

By (2), these codes cover all of $\mathcal{H}_n$. If any vector in $\mathcal{H}_n$ appears in more than one code in $\mathbb{X}$, we may discard it from all but one code in $\mathbb{X}$. Thus we may assume that the codes $\mathcal{C}(\mathbf{y})$ form a partition of $\mathcal{H}_n$. As each $\mathcal{C}(\mathbf{y})$ is also an $\langle n, t\rangle$-code, by assigning a distinct hash value to each $\mathcal{C}(\mathbf{y})$, we attain a $t$-separable function. The number of codes in (4) is clearly $V_n(s) \le V_n(t-1)$ which is also the number of hash values. This proves the upper bound. $\square$

One can use the bounds for $V_n(t)$ given by (1) to estimate $m$ in (3). Specifically, taking logs in (3) and using (1), we obtain

$$\frac{t \log\left(\frac{2n}{t}\right)}{2} \le m \le (t-1)\log\left(\frac{ne}{t-1}\right) \tag{5}$$

Thus the hash length grows logarithmically with the message length.

It will be instructive to see how the separability factor varies for some standard hash and message lengths. Let us choose $m = 256$. Using (5), we bound the values of $t$ for some standard message lengths in the table below.

| Message length $n$ (bits) | Range for $t$ |
|---|---|
| $2^{13} = 1\text{KB}$ | $28 \leq t \leq 64$ |
| $2^{16} = 8\text{KB}$ | $21 \leq t \leq 44$ |
| $2^{23} = 1\text{MB}$ | $14 \leq t \leq 26$ |
| $2^{33} = 1\text{GB}$ | $10 \leq t \leq 17$ |

**Table 1.** Bounds for $t$ for some standard message lengths when $m = 256$

We observe that as $n$ gets larger, the range for $t$ gets smaller. This is because the functions $t \log\left(\frac{2n}{t}\right)$ and $(t-1)\log\left(\frac{ne}{t-1}\right)$ increase with $n$ and $t$ (when $t \leq \frac{2n}{e}$). Thus, for a fixed $m$, as $n$ increases, $t$ must get smaller in order to satisfy (5).

The bounds for $t$ should be interpreted in the following manner: for a *fixed* message and hash length, if the goal is to maximize the separability factor, then its value lies in the range given in the table.

The upper (optimistic) bound for $t$ is based on the assumption that we can find codes such that every message vector is within distance $t/2$ of some codeword.

Table 1 illustrates the relationship between collision resistance and $t$-separability. For instance, when 1MB messages are hashed to 256 bit strings, we infer that there are always two messages with the same hash that differ from each other in at most 27 bits.

## 3 Building Blocks for Realizing $t$-separability

One way to realize the lower bound in Lemma 1 is to use *perfect codes* [5]. Then the balls around the codewords in $\mathcal{C}$ are disjoint and the lower bound in (3) is attained for $m$. For example, to construct 3-separable functions, one could use *Hamming codes* which are $(n = 2^m - 1, 2^m - m - 1, 3)$-codes. However, realizing hash functions through perfect codes is only possible for certain values of $t$ and such realizations do not satisfy the other properties desirable in a hash function (collision resistance...).

The construction in Lemma 1 allowed us to build a $t$-separable function from an $\langle n,t\rangle$-code $\mathcal{C}$, but this does not satisfy the other properties of a hash. For instance, these functions are not 2nd preimage-resistant; given a codeword, to find another that hashes to the same value, we simply choose another codeword from the same code. So it seems as if the code itself needs to be kept a secret.

A solution to this problem is to use *one-way bijections* to construct hashes. The idea is to replace the global hash by a collection of one-way bijections each centered around a codeword. Given a maximal $\langle n,t\rangle$-code $\mathcal{C}$ of covering radius $s$, and an $\mathbf{x} \in \mathcal{C}$, let

$$\pi_{\mathbf{x}} : B_n(\mathbf{x}, s) \to \mathcal{H}_m \tag{6}$$

be the one-way bijection that maps elements in $B_n(\mathbf{x}, s)$ to distinct hash values. The map $\mathbf{x} \to \pi_{\mathbf{x}}$ is defined in a pseudorandom manner using a one-way function (i.e. determining $\mathbf{x}$ from $\pi_{\mathbf{x}}$ should be computationally hard). As a result,

1. The map $\pi_{\mathbf{x}}$ depends on $\mathbf{x}$ and is thus different for each ball.
2. Such a map satisfies the first three requirements of the hash function but not the fourth. This is because, within each $B_n(\mathbf{x}, s)$, $\pi_{\mathbf{x}}$ distributes the hashes randomly. Thus, it's not necessarily true that vectors with the same hash value are a distance $t$ apart. However, as we show in Theorem 2, for appropriate choices of $\pi_{\mathbf{x}}$, the average distance between two vectors with the same hash value is at least $t$.
3. The balls $B_n(\mathbf{x}, s)$ are not necessarily disjoint. Thus some care has to be taken to define the maps $\pi_{\mathbf{x}}$.

**Theorem 2.** *Assuming that the map in (6) distributes the hash values randomly and independently in each ball, and $n \geq 2t$, the expected distance between two vectors with the same hash is at least $t$.*

*Proof.* The value we want to estimate is equivalent to the expected distance between two randomly chosen vectors in balls of radius $s$ whose centers are a distance $t$ apart. WLOG assume that the two centers are $\mathbf{0}$ and $\mathbf{t}$ where $\mathbf{t}$ is a vector of weight $t$ with its first $t$ components equal to 1. Then

$$\begin{aligned} E(d(\mathbf{x}, \mathbf{t}+\mathbf{y}) \,|\, \mathbf{x}, \mathbf{y} \in B_n(\mathbf{0}, s)) &= E(\mathrm{wt}(\mathbf{t}+\mathbf{x}+\mathbf{y}) \,|\, \mathbf{x}, \mathbf{y} \in B_n(\mathbf{0}, s)) \\ &= E(\mathrm{wt}(\mathbf{t}+\mathbf{v}) \,|\, \mathbf{v} = \mathbf{x}+\mathbf{y},\ \mathbf{x}, \mathbf{y} \in B_n(\mathbf{0}, s)) \end{aligned} \tag{7}$$

Now consider a $\mathbf{v}$ of the form above such that $\mathrm{wt}(\mathbf{t}+\mathbf{v}) < t$. Such a $\mathbf{v}$ must have more 1s in the first $t$ components than in the last $n-t$ components. Let $\alpha$, $\beta$ and $\gamma$ be the number of 1s in the first $t$, the next $t$ and the last $(n-2t)$ components of $\mathbf{v}$. Then

$$\mathrm{wt}(\mathbf{t}+\mathbf{v}) = t - \alpha + \beta + \gamma.$$

We define a corresponding vector $\mathbf{v}'$ obtained from $\mathbf{v}$ by reflecting the first $2t$ components about the $t^{\text{th}}$ position (i.e. $\mathbf{v}'(i) = \mathbf{v}(2t+1-i)$ for $i = 1, \ldots, 2t$). Then $\mathbf{v}'$ has $\beta$, $\alpha$ and $\gamma$ 1's in the first $t$, next $t$ and last $(n-2t)$ components and

$$\mathrm{wt}(\mathbf{t}+\mathbf{v}') = t - \beta + \alpha + \gamma.$$

Hence $\mathrm{wt}(\mathbf{t}+\mathbf{v}) + \mathrm{wt}(\mathbf{t}+\mathbf{v}') \geq 2t$ and the average of this sum is at least $t$. As there are an equal number of ways of expressing $\mathbf{v}$ or $\mathbf{v}'$ as the sum of two vectors in $B_n(\mathbf{0}, s)$, we infer that the expected value in (7) is at least $t$. □

We show that one-way bijections can be derived from one-way permutations (a one-way bijection from a set to itself), a concept well studied in literature (see for example [3]). To create a hash function using the method outlined above, one performs the following steps. Let

$$\mathcal{F} := \{\phi : \mathcal{H}_m \to \mathcal{H}_m | \phi \text{ is a one-way permutation}\}$$

be a family of one-way permutations on $\mathcal{H}_m$.

Given a message $\mathbf{m} \in \mathcal{H}_n$, an $\langle n, t\rangle$-code $\mathcal{C}$ of covering radius $s$ with $V_n(s) \leq 2^m$, a *1-way generation* function

$$\alpha : \mathcal{C} \to \mathcal{F}$$

that derives the one-way permutation for each ball centered around a codeword and a one-to-one *enumeration* function

$$\beta : B_n(\mathbf{0}, s) \to \mathcal{H}_m$$

that represents the vectors in $B_n(\mathbf{0}, s)$ as $m$-bit strings, we determine the hash of a message $\mathbf{m} \in \mathcal{H}_n$ in the following manner.

1. Locate the codeword $\mathbf{x} \in \mathcal{C}$ nearest to $\mathbf{m}$. For instance, if we assume $\mathbf{m}$ to be a message received over a noisy channel, then we can use decoding techniques to recover the nearest codeword $\mathbf{x}$. It follows that $\mathbf{m} \in B_n(\mathbf{x}, s)$.

2. Determine the one-way permutation $\alpha(\mathbf{x})$ corresponding to $\mathbf{x}$.
3. Determine the input to the one-way permutation $\mathbf{y} := \beta(\mathbf{m}+\mathbf{x}) \in \mathcal{H}_m$. Note that $\mathbf{m}+\mathbf{x} \in B_n(\mathbf{0}, s)$ since $\mathbf{m} \in B_n(\mathbf{x}, s)$.
4. Output the hash $\alpha(\mathbf{x})(\mathbf{y}) \in \mathcal{H}_m$.

We discuss below, the security of this method and its implications.

**Handling variable length messages:** Our construction fixes the message length to be $n$. Hash functions in general map variable length messages to a fixed length hash value.

A possible solution is to fix $n$ at a large value. Given a message of length less than $n$, we prepend it with the sequence $0\ldots01$ to derive an $n$-bit string and then apply the transformation described above. For example, the message 0101 is transformed to the $n$-bit string $0\ldots010101$. One sees easily that this transformation is one-to-one, i.e. distinct messages map to different $n$-bit strings.

Messages of length $n$ bits or longer are split into $(n-1)$ bit strings and a hash is constructed for each string.

A small value of $n$ will lead to efficiency in computation of hash but lead to the splitting of many messages. However, too large a value of $n$ may lead to an inefficient hash computation. The choice of $n$ is also determined by the presence of codes with rapid decoding techniques as well as the separability factor. As shown in Table 1, the separability factor decreases with $n$ and for large separability factors, $n$ should be small.

**Choice of $\mathcal{C}$:** The code $\mathcal{C}$ should be chosen to allow for efficient (in space and time) decoding of messages. As both $n$ and $|\mathcal{C}|$ are large, typical decoding techniques using *syndromes* (see [8]) may prove inefficient (the syndrome table would have a size of order $2^m$ which is too large for realistic hash lengths).

**Preimage resistance:** Given a hash value, the one-way property of $\mathcal{F}$ ensures that finding its preimage is an intractable problem.

**2nd preimage and collision resistance:** We note that $\alpha$ is not one-to-one if $|\mathcal{C}| > |\mathcal{F}|$. To break 2nd preimage or collision resistance, one would need to find two codewords $\mathbf{x}_1$ and $\mathbf{x}_2$ such that $\alpha(\mathbf{x}_1) = \alpha(\mathbf{x}_2)$. Then the permutation function $\alpha(\mathbf{x})$ within the balls centered at $\mathbf{x}_1$ and $\mathbf{x}_2$ would be identical.

To provide for collision resistance we require that

1. $\alpha$ be one-way.
2. $|\mathcal{F}|$ be large. This is because, using the birthday attack (see [6, 9.7.1] for a description of the attack) to break collision resistance (on $\alpha$) requires $O(|\mathcal{F}|^{1/2})$ operations.

Even if two such codewords are found, our mapping ensures that the messages are sufficiently spaced apart.

**Constructing one-way permutations on $\boldsymbol{\mathcal{H}_m}$:** One-way permutations are functions that map a set to itself and are easy to evaluate but computationally hard to invert; see [3] for a precise definition. The intractability of some public key cryptosystems is based on the existence of such functions. These include the discrete log problem in elliptic curves, the RSA algorithm and the discrete log problem in the multiplicative group modulo a prime $p$.

Here's one way of constructing the family $\mathcal{F}$. Let $\mathbb{G}$ be a cyclic group of prime order $p \approx 2^m$ (with $p \leq 2^m$) such that

– The discrete log problem in $\mathbb{G}$ is intractable.
– There is a natural ordering of the elements of $\mathbb{G}$, i.e. elements of $\mathbb{G}$ can be mapped to $m$-bit vectors.

An example of such a group is the elliptic curve group of prime order over a finite field. Using point compression techniques, a point $(\mathbf{x}, \mathbf{y})$ on the curve can be represented as a vector $(\mathbf{x}, b)$ where $b$ is 0 or 1; see [4] for an introduction to elliptic curves and [1, IV.4] for point compression techniques.

If $g \in \mathbb{G}$ is not the identity element, then the map

$$\phi_g : \mathbb{Z}_p \to \mathbb{G}, \qquad \phi_g(y) = y.g$$

is a one-way bijection. Composing this map with point compression yields $|\mathbb{G}| \approx 2^m$ one-way permutations. Constructions using RSA or the discrete log problem in multiplicative groups are described in [3].

As we observed earlier, the security of this scheme is directly related to the size of $\mathcal{F}$. Estimating the number of one-way permutations on $\mathcal{H}_m$ is an interesting problem. Since the number of permutations on $\mathcal{H}_m$ is $2^m! \gg 2^m$, it's likely that $\mathcal{F}$ could be made much larger.

## 4 A Hash Function using Cyclic Codes and an Elliptic Curve Group

We illustrate the ideas presented in the previous section with a practical scheme using cyclic codes and an elliptic curve group over a prime field. For ease of computation, our method associates a hypercube (instead of a ball) with each codeword and a somewhat different decoding technique.

A *linear code* is invariant under addition of codewords, i.e. if $\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{C}$, then $\mathbf{c}_1 + \mathbf{c}_2 \in \mathcal{C}$. Thus a linear code can be viewed as a vector space over $\mathbb{F}_2$. It's *dimension* is the dimension of this vector space.

A *cyclic code* $\mathcal{C}$ is a linear code that is invariant under cyclic shifts (i.e. if $\mathbf{c} = (c_0, \dots, c_{n-1}) \in \mathcal{C}$ then $(c_1, \dots, c_{n-1}, c_0) \in \mathcal{C}$). See [5] for an introduction to linear and cyclic codes.

There is a natural association between (binary) polynomials of degree $< n$ and elements of $\mathcal{H}_n$. With every vector

$$\mathbf{a} = (a_0, a_1, \dots, a_{n-1}) \in \mathcal{H}_n$$

we associate the polynomial

$$a(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}$$

and vice versa. Thus we will use these notations interchangeably.

We consider an $\langle n, t \rangle$ cyclic code $\mathcal{C}$ of dimension $(n - m)$ defined by a *generator polynomial* $g(x)$ of degree $m$ with $g(x)\,|x^n + 1$. $g$ is chosen to maximize $t$. The code is then given by (see [5] for a proof)

$$\mathcal{C} = \{q(x)g(x) \,|\, \deg(q(x)) < n - m\}.$$

This associates codewords with binary strings of length $(n - m)$; given a codeword $\mathbf{a} = q(x)g(x)$, the corresponding string is $\mathbf{q}$.

For each $\mathbf{x} \in \mathcal{C}$, we define the *area around* $\mathbf{x}$ to be

$$\mathcal{A}(\mathbf{x}) := \{\mathbf{x} + h(x) \,|\, \deg(h(x)) < m\}.$$

We note that

1. $\mathcal{A}(\mathbf{x})$ defines a hypercube of size $2^m$ with $\mathbf{x}$ being one of the vertices.
2. $\mathcal{A}(\mathbf{x})$ contains no other codeword from $\mathcal{C}$.
3. The collection $\{\mathcal{A}(\mathbf{x}) | \mathbf{x} \in \mathcal{C}\}$ forms a partition of $\mathcal{H}_n$ into $2^{n-m}$ regions each of size $2^m$.

4. A message $\mathbf{m}$ is "decoded" to the codeword $\mathbf{x}$ if $\mathbf{m} \in \mathcal{A}(\mathbf{x})$. We observe that this does not correspond to *minimum distance decoding* as other codewords may be nearer to $\mathbf{m}$. However, the decoding algorithm is efficient; given $\mathbf{m}$, divide $m(x)$ by $g(x)$ to give the remainder $r(x)$ of degree $< m$. Then, $m(x) - r(x)$ is the decoded codeword.
5. The remainder on dividing a message $m(x)$ by $g(x)$ allows us to associate messages with $m$-bit strings.

We now fix $m = 255$. We assume a one-way function

$$\gamma : \mathcal{H}_{n-m} \rightarrow \mathcal{H}_m$$

(more specifically, $\gamma : \mathcal{C} \rightarrow \mathcal{H}_m$). For instance, $\gamma$ could be the SHA-256 hash [7] restricted to 255 bits.

The elliptic curve we choose is one of the named curves recommended by NIST, curve P-256. This is defined over a 256 bit prime field and generates a cyclic group of prime order $< 2^{256}$; see [4] for the curve parameter values.

Let $G$ be the base point of this group. Given a message $\mathbf{m}$ that decodes to $\mathbf{x}$, and gives a remainder $r(x)$, our hash value is

$$\texttt{compress}(r(x) \cdot (\gamma(\mathbf{x}) \cdot G))$$

where `compress` is the point compression function [1, I, IV.4] and $\cdot$ is the point multiplication operation. The compression operation expands the x-xoordinate of the product by a single bit, resulting in a hash length of 257 bits.

We observe that

1. Two distinct messages that decode to the same codeword $\mathbf{x}$ will have different hash values. This is because they will have different remainders (say $r_1(x)$ and $r_2(x)$) and as a result, the points $r_1(x) \cdot (\gamma(\mathbf{x}) \cdot G)$ and $r_2(x) \cdot (\gamma(\mathbf{x}) \cdot G)$ will be different. Note that we have restricted the size of $\mathcal{A}(\mathbf{x})$ to $2^{255}$ which is less than the order of $\gamma(\mathbf{x}) \cdot G$.
2. While the hash length is 257, the number of possible hash values is the order of the curve $\approx 2^{256}$.
3. This map provides collision resistance provided $\gamma$ is collision resistant. Finding two messages with the same hash value is equivalent to determining two codewords $\mathbf{x}_1$ and $\mathbf{x}_2$ with $\gamma(\mathbf{x}_1) = \gamma(\mathbf{x}_2)$.

Now we estimate the minimum distance between two messages with the same hash. We assume that the point multiplication operation distributes the hashes randomly and independently in each hypercube. The minimum distance will be attained when the corresponding codewords are a distance $t$ apart. Thus we may restrict ourselves to the regions $\mathcal{A}(\mathbf{0})$ and $\mathcal{A}(\mathbf{t})$ where $\mathbf{t}$ is a codeword of (minimum possible) hamming weight $t$. We estimate the expected value of the distance between two randomly chosen vectors in $\mathcal{A}(\mathbf{0})$ and $\mathcal{A}(\mathbf{t})$ as

$$\begin{aligned} E(d(\mathbf{v}_1, \mathbf{v}_2) |\, \mathbf{v}_1 \in \mathcal{A}(\mathbf{0}), \mathbf{v}_2 \in \mathcal{A}(\mathbf{t})) &= E(d(\mathbf{v}_1, \mathbf{t} + \mathbf{v}_2) |\, \mathbf{v}_1, \mathbf{v}_2 \in \mathcal{A}(\mathbf{0})) \\ = E(\mathrm{wt}(\mathbf{t} + \mathbf{v}_1 + \mathbf{v}_2) |\, \mathbf{v}_1, \mathbf{v}_2 \in \mathcal{A}(\mathbf{0})) &\geq E(\mathrm{wt}(\mathbf{v}_1 + \mathbf{v}_2) - t |\, \mathbf{v}_1, \mathbf{v}_2 \in \mathcal{A}(\mathbf{0})) \\ &= m/2 - t. \end{aligned}$$

Thus, vectors with the same hash are sufficiently spaced apart.

## 5 The Consecutive Distance Problem

Another problem of practical interest is the detection of messages with the same hash that differ at some consecutive bits. For example, to change the message “pay one 1,000 dollars” to “pay 1,000,000 dollars” requires altering five consecutive bytes. We therefore reformulate the hashing problem by introducing the notion of consecutive distance.

Define the *consecutive distance* between two messages $\mathbf{m}_1$ and $\mathbf{m}_2$ (abbreviated $\mathrm{CD}(\mathbf{m}_1, \mathbf{m}_2)$) as the minimum number of consecutive bits that must possibly be altered in $\mathbf{m}_1$ to arrive at $\mathbf{m}_2$. In other words, if $\mathbf{m}_1$ and $\mathbf{m}_2$ differ at positions $i_1 < i_2 < \cdots < i_j$, then $\mathrm{CD}(\mathbf{m}_1, \mathbf{m}_2) = i_j - i_1$.

A *$t$-c.d. separable* hash function is a map $\mathcal{H}_n \to \mathcal{H}_m$, such that for any two messages $\mathbf{m}_1, \mathbf{m}_2$ with the same hash, $\mathrm{CD}(\mathbf{m}_1, \mathbf{m}_2) \geq t$. We call $t$, the *c.d. separability factor.*

The following argument shows that for a $t$-c.d. separable hash function, $t \leq m$.

Consider the collection of $2^m$ messages that agree on all but the first $m$ bits. If any two of these messages have the same hash, then the consecutive distance between them is $< m$ and we are done. Otherwise, consider a message $\mathbf{m}$ that differs on the $(m+1)^{\text{th}}$ bit from this collection. $\mathbf{m}$ must have the same hash as some message $\mathbf{m}_1$ in this collection and $\mathrm{CD}(\mathbf{m}, \mathbf{m}_1) \leq m$.

Lower bounds for the c.d. separability factor is achieved through explicit constructions. Since on an average, $2^{n-m}$ messages must map to a single hash, we first find a code $\mathcal{C}$ of size $2^{n-m}$ such that the consecutive distance between any two codewords in $\mathcal{C}$ is at least $m$. $\mathcal{C}$ is defined in the following manner.

$$\begin{aligned} \mathcal{C} &:= \{(x_{n-m-1}, \ldots, x_0, y_{m-1}, \ldots, y_0) \in \mathcal{H}_n | x_i = 0 \text{ or } 1\} \text{ where} \\ y_i &= x_i \oplus x_{i+m} \oplus \cdots \oplus x_{i+lm} = \bigoplus_{j=0}^{l} x_{i+jm} \text{ for } i = 0, \ldots, m-1 \end{aligned}$$

and $l = \lfloor (n-m-1-i)/m \rfloor$. In other words, to construct $\mathcal{C}$, we take all possible $0, 1$ combinations for the first $(n-m)$ components (giving $2^{n-m}$ vectors). The last $m$ components are defined by taking x-or of every $m^{\text{th}}$ component in $(x_{n-m-1}, \ldots, x_0)$.

When $n < 2m$, some of the x-or terms will have an empty sum. We fix the corresponding $y_i$ to 1.

We claim that the minimum consecutive distance of $\mathcal{C}$ is at least $m$. Given two distinct codewords $\mathbf{a}, \mathbf{b} \in \mathcal{C}$, if the consecutive distance between them in the first $(n-m)$ components is at least $m$, then we are done. Otherwise, there exists a component $x_i$ that takes values 0 and 1 in $\mathbf{a}$ and $\mathbf{b}$. As a result, the corresponding component $y_i$ must also be different in $\mathbf{a}$ and $\mathbf{b}$. Since $x_i$ and $y_i$ are at least $m$ apart, the consecutive distance between $\mathbf{a}$ and $\mathbf{b}$ is at least $m$ proving our claim.

Next we define $2^m$ such codes each of size $2^{n-m}$ that partition $\mathcal{H}_n$. These are simply the translates $\mathcal{C}((\mathbf{0}, \mathbf{z}))$ for each $\mathbf{z} \in \mathcal{H}_m$ (i.e. the $\mathbf{y}$ component of each codeword in $\mathcal{C}$ is x-ored by $\mathbf{z}$). As a result these codes also have minimum consecutive distance $m$.

Thus for a hash function, the c.d. separability factor may be much higher than the separability factor. The challenge here is constructing a *t-c.d. separable* hash function that satisfies the desirable properties of a hash (collision resistance ...).

## 6 Conclusions

The notion of separability is desirable if the goal is to disallow an attacker from altering a few bits in a message to generate another with the same hash. A related notion is that of *consecutive distance* where messages with the same hash differ in bits that are spaced far apart.

Both of these concepts were introduced and we showed how separable hash functions could be constructed using algebraic codes and one-way permutations. An explicit construction using cyclic codes and point multiplication over the elliptic curve P-256 was realized. Finally bounds on the c.d. separability factor were derived.

**Acknowledgement.** The author thanks M. Vidyasagar for raising the consecutive distance problem and for his feedback on the earlier drafts of this paper.

## References

1. I. Blake, G. Seroussi, and N. P. Smart, Eds. *Advances in Elliptic Curve Cryptography*, London Mathematical Society Lecture Note Series **317**, Cambridge University Press, 2005.
2. G.D. Cohen, S.N. Litsyn, A.C. Lobstein and H.F. Mattson Jr. Covering radius 1985–1994. *Applicable Algebra in Engineering Communication and Computing*, 8:173–239, 1997.
3. O. Goldreich, L.A. Levin and N. Nisan, "On Constructing 1-1 One-Way Functions", *Electronic Colloquium on Computational Complexity* (ECCC) 1995. Available online at *ftp://theory.lcs.mit.edu/pub/people/oded/gln.ps.*
4. D. Johnson and A. Menezes, "The Elliptic Curve Digital Signature Algorithm (ECDSA)", Technical Report CORR 99-34, Dept. of C&O, University of Waterloo, 1999.
5. F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, Amsterdam, 1993.
6. A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997. Available online at *http://www.cacr.math.uwaterloo.ca/hac/.*
7. "SHA-256 Cryptography Software", *http://www.cryptosys.net/sha256.html*
8. J. H. Van Lint, *Introduction to Coding Theory*, GTM **86** (2nd ed.), Springer-Verlag, 1992.
9. M. Vidyasagar, *Learning and Generalization with Applications to Neural Networks*, Springer, Second Edition, 2002.
10. X. Wang, H. Yu and Y.L. Yin, "Efficient Collision Search Attacks on SHA-0", *CRYPTO 2005*, 1–16.
11. X. Wang, Y.L. Yin and H. Yu, "Finding Collisions in the Full SHA-1", *CRYPTO 2005*, 17–36.