




Property-Preserving Hash Functions and Combinatorial Group Testing

Kazuhiko Minematsu   

NEC, Tokyo, Japan

Yokohama National University, Japan

Abstract

Property-preserving hash (PPH) function is a class of hash functions that allows an evaluation of the property of inputs from their hash values. Boyle et al. at ITC 2019 recently introduced it and considered the robustness of PPH against an adversary who accesses the internal randomness of PPH, and proposed two robust PPH constructions for a weak form of Hamming distance predicate. The second construction received attention for its short hash value, although it relies on an ad-hoc security assumption. The first construction, which is entirely hash-based and based on the classical collision-resistance assumption, has been largely overlooked. We study their first construction and discover its close connection to a seemingly different field of hash/MAC-based (adversarial) error detection using the theory of Combinatorial Group Testing (CGT). We show some consequences of this discovery. In particular, we show that some existing proposals in the field of CGT-based error detection can be converted into a PPH for the Hamming distance property, and they immediately improve and generalize Boyle et al.'s hash-based PPH proposal. We also show that the idea of Boyle et al. is useful in the context of a variant of CGT problem.

2012 ACM Subject Classification Security and privacy → Hash functions and message authentication codes

Keywords and phrases Hash function, Property-Preserving Hash, Combinatorial Group Testing, Provable Security

Digital Object Identifier 10.4230/LIPIcs.ITC.2022.2

Related Version *Previous Version:* <https://eprint.iacr.org/2022/478>

1 Introduction

Compressing a large amount of data into small digests while maintaining some of their properties is one of the fundamental goals in computer science. Popular algorithms such as Bloom filter [2] or Cuckoo hashing [31] offer such property-preserving hashing for approximate set membership property. Locality-sensitive hash (LSH) functions [22] allow for compressing two inputs independently and evaluating if they are close or not with respect to some metric from their digests. These algorithms are randomized and usually studied in the setting where inputs are independent of the internal random coin. In real-world use cases, this may not be enough, because we often need to consider an adversary who has an incentive to corrupt the algorithm by giving maliciously crafted inputs. Such an adversary would somehow try to learn the internal random coin, however, basic property-preserving hashing algorithms have no guarantee against such attacks.

Based on this motivation, Boyle et al. [4] (BLV19) initiated the study of *robust* property-preserving hash (PPH) functions that resist such attacks. They proposed two constructions for the Gap Hamming predicate, which is a weak form of Hamming distance predicate. The first construction was entirely hash-based and relied on the classical collision resistance of the hash function. The second was based on an ad-hoc assumption related to the hardness of decoding linear codes. The second construction has a much shorter hash size than the



© Kazuhiko Minematsu;

licensed under Creative Commons License CC-BY 4.0

3rd Conference on Information-Theoretic Cryptography (ITC 2022).

Editor: Dana Dachman-Soled; Article No. 2; pp. 2:1–2:14

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

first and stipulated research on pairing or lattice-based PPHs for (exact) Hamming distance predicate with similar hash size, as shown by Fleischhacker and Simkin [18] and Fleischhacker et al. [17]. In contrast, the first construction has been largely overlooked since the proposal, possibly because of its large hash size. However, its simplicity and computational efficiency, and high-security reliance may make it attractive in practical applications.

1.1 Our Contributions

The core idea of BLV19’s first construction (hereafter PPH1) is subsampling. That is, taking hash values for a predetermined set of input subsequences and concatenating them. We discover that this idea of PPH1 is closely related to a different application field of hash functions or message authentication codes (MACs), namely hash/MAC function with detection capability [19, 14, 25]. Such a hash/MAC function can be interpreted as an application of classical Combinatorial Group Testing (CGT) to the detection of data corruptions, where detection means to pinpoint the parts of the input data that have been corrupted. For convention, we say *CGT-based hash function* to mean this application. Our finding derives several interesting consequences.

In more detail, we show that known schemes in the aforementioned field of CGT-based hash can be interpreted as a robust PPH (in the sense of BLV19) for the Hamming distance predicate, with a much better compression rate, i. e. smaller hash size, than PPH1. Note that PPH1 only preserves a weaker Gap Hamming distance predicate. CGT-based hash functions also have smaller computation time for hashing and preserve more informative properties than plain Hamming distance predicate. The security (robustness) against the adversary is also proved with minor modifications to the original proofs for the security notions of CGT-based hash. Moreover, a MAC-based PPH is also naturally derived from the known CGT-based MACs [19, 25, 26]. As well as the hash-based counterpart, a MAC-based PPH preserves the Hamming distance property and fulfills a weak form of robustness defined by BLV19. Moreover, by using the technique of [25], the resulting scheme can significantly reduce the hashing time thanks to its special structure for the internal MAC function. Finally, we show that the construction of PPH1, which uses a bipartite expander graph, is also useful in building a test matrix for CGT that aims at estimating the number of defectives rather than detecting them [7]. Therefore, the connection is not one-way.

We came up with this finding while studying the existing CGT-based hash/MAC schemes. Our study did not present any new scheme. Once found, the connection may look rather obvious, while we are not aware in the literature and present a formal security analysis. We think bridging two seemingly different areas is important and will help understanding and development of both areas.

2 Preliminaries

2.1 Notations

Let $[i]$ denote $\{1, \dots, i\}$ for $i \geq 1$. The set of all binary strings is denoted by $\{0, 1\}^*$, which includes the empty string ε . We write the bit length of $X \in \{0, 1\}^*$ by $|X|$, where $|\varepsilon| = 0$. The Hamming weight of a binary string X is denoted by $\text{Hw}(X)$. For $X = (X_1, \dots, X_n) \in \mathcal{X}^n$ for some finite set \mathcal{X} and $V = (V_1, \dots, V_n) \in \{0, 1\}^n$, let $X_{|V}$ be the sub-sequence of X indexed by V . That is, $X_{|V}$ is $(X_{i_1}, \dots, X_{i_v})$, where $\text{Hw}(V) = v$ and $i_j \in [n]$ is the index of the j -th bit set at V . For $X, X' \in \mathcal{X}^n$, let $\mathcal{D}(X, X') := (\text{test}(X_1, X'_1), \dots, \text{test}(X_n, X'_n))$, where test is a test function (i. e., $\text{test}(A, B) = 1$ if $A = B$ and 0 otherwise). For $X, Y \in \{0, 1\}^n$, $X \vee Y$ denotes the bitwise Boolean sum (logical OR) of X and Y .

For a keyed function $F : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$ for key space \mathcal{K} , we may write $F_K(X)$ instead of $F(K, X)$. Let $\text{negl}(\lambda)$ for a security parameter $\lambda \in \mathbb{N}$ be a negligible function. For a finite set \mathcal{X} we write $X \xleftarrow{\$} \mathcal{X}$ to mean X is uniformly sampled over \mathcal{X} . For a probabilistic polynomial-time (PPT) adversary A and the oracles $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_c, A^{\mathcal{O}_1, \dots, \mathcal{O}_c} \rightarrow Y$ denotes the event that A possibly adaptively queries to \mathcal{O}_i in an arbitrarily order and outputs Y . The set of all functions of domain \mathcal{X} and range \mathcal{Y} is written as $\text{Func}(\mathcal{X}, \mathcal{Y})$ and a uniformly random function (URF) $R : \mathcal{X} \rightarrow \mathcal{Y}$ is a random function that uniformly distributes over $\text{Func}(\mathcal{X}, \mathcal{Y})$.

2.2 Advantage Functions

Let $F, G : \mathcal{X} \rightarrow \mathcal{Y}$ be two (possibly randomized) functions. Let A be an adversary who tries to distinguish F from G using oracle access, and outputs a binary decision. We define

$$\text{Adv}_{F,G}^{\text{IND}}(A) := \Pr[A^F \rightarrow 1] - \Pr[A^G \rightarrow 1]$$

as the advantage of A in distinguishing F and G . Similarly, let $\text{Adv}_F^{\text{Event}}(A)$ denote the probability of an event Event invoked by A in the game.

2.3 Matrix Representation

Let \mathbb{M} be an $n \times m$ binary matrix. We write $\mathbb{M}_{i,*}$ to denote the i -th row, and $\mathbb{M}_{*,j}$ to denote the j -th column, and $\mathbb{M}_{i,j}$ to denote the entry at i -th row and j -th column. For simplicity we may abbreviate $\mathbb{M}_{i,*}$ to \mathbb{M}_i . The rows and columns of \mathbb{M} are interchangeably seen as sets, e.g. $\mathbb{M}_i = \{j \in [m] : \mathbb{M}_{i,j} = 1\}$, and $a \in \mathbb{M}_i$ means $\mathbb{M}_{i,a} = 1$. This may also apply to any binary string.

2.4 Property-Preserving Hash Function

Following BLV19, we describe the basics of property-preserving hash functions. For a finite set \mathcal{Z} , a \mathcal{Z} -valued property for a pair of messages in message space \mathcal{X} is a function $P : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{Z}$. When $|\mathcal{Z}| = 2$ it is called a predicate. A property is called *promise* if it is undefined for some inputs and otherwise called *total*. We only consider the latter case. This paper focuses on two-input properties, but generalization is possible.

► **Definition 1.** Let $\mathcal{X} = \mathcal{B}^n$ for some set \mathcal{B} and a positive integer n . let $\text{Hd}(X, X')$ for $X, X' \in \mathcal{X}$ be the generalized Hamming distance defined as

$$\text{Hd}(X, X') := \text{Hw}(\mathcal{D}(X, X')).$$

As we may see a binary string as a set, we also have $\text{Hd}(X, X') = |\mathcal{D}(X_i, X'_i)|$. The Hamming predicate with threshold $d \in [n]$ is defined as

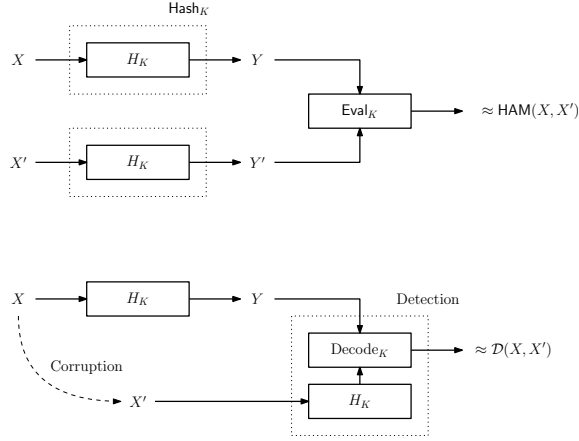
$$\text{HAM}^d(X, X') := \begin{cases} 1 & \text{if } \text{Hd}(X, X') \geq d \\ 0 & \text{otherwise} \end{cases}$$

Moreover, the (generalized) Gap-Hamming predicate with parameter (d, ϵ) for $\epsilon \in (0, 1)$ is defined as

$$\text{GapHAM}^{d,\epsilon}(X, X') := \begin{cases} 1 & \text{if } \text{Hd}(X, X') \geq d \cdot (1 + \epsilon) \\ 0 & \text{if } \text{Hd}(X, X') \leq d \cdot (1 - \epsilon) \\ \text{undefined} & \text{otherwise} \end{cases}$$

As shown above, Gap-Hamming is a promise predicate.

For simplicity, we assume $\mathcal{B} = \{0, 1\}^b$ for some $b \geq 1$. When $b = 1$, $\text{Hd}(*, *)$ corresponds to the classical Hamming distance between the bit strings.



■ **Figure 1** PPH (top) and CGT-based MAC/Hash scheme defined at Section 4 (bottom).

► **Definition 2** (Property-preserving Hash Function). Let $H : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$ be a keyed function. If H is a property-preserving hash (PPH) function for a property $P : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{Z}$, it requires the following algorithms:

- Sampling (key-generation) function $Sample(\lambda)$ for the security parameter λ that efficiently samples K over \mathcal{K} (we hereafter assume λ as a fixed constant and simply write $K \xleftarrow{\$} \mathcal{K}$ to represent this)
- Hash output evaluation function $Hash : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$ that returns $H(K, X)$ on input (K, X)
- Property evaluation function $Eval : \mathcal{K} \times \mathcal{Y} \times \mathcal{Y} \rightarrow \mathcal{Z}$

We say H is η -compressing if $\log |\mathcal{Y}| \leq \eta \log |\mathcal{X}|$ for $0 < \eta < 1$.

The role of $Eval$ is to give an estimate of $P(X_1, X_2)$ for some $X_1, X_2 \in \mathcal{X}$, using the key K and their corresponding hash values.

2.5 Robustness Notions for PPHs

Let $H : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$ be a PPH for a property $P : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{Z}$ and let $H.hash$ and $H.eval$ be the corresponding $Hash$ and $Eval$ functions. The goal of PPH has some similarities to the existing objects, such as LSH [22], which is a randomized hashing algorithm that preserves the distance between the inputs for some metric. However, the security against the adversary who may query the hashing and evaluation processes or even access the internal randomness has not been formally studied. Based on this observation, BLV19 introduced multiple robustness notions of PPH to capture such security, using the security parameter λ . Each robustness notion has a different adversary model, as shown below.

► **Definition 3** (Non-Robust PPH [4]). If H is a non-robust PPH for a property P , for any PPT adversary A ,

$$\begin{aligned} \mathbf{Adv}_H^{\text{NR-PPH}}(A) \\ := \Pr[K \xleftarrow{\$} \mathcal{K} : A \rightarrow (X, X'), Eval(K, H_K(X), H_K(X')) \neq P(X, X')] \leq \text{negl}(\lambda). \end{aligned}$$

► **Definition 4** (Evaluation-Oracle-Robust PPH [4]). If H is an Evaluation-Oracle (EO) robust PPH for a property P , for any PPT adversary A ,

$$\begin{aligned} \mathbf{Adv}_H^{\text{EO-PPH}}(A) \\ := \Pr[K \xleftarrow{\$} \mathcal{K}; A^{\mathcal{O}_{H.Eval}} \rightarrow (X, X') : Eval(K, H_K(X), H_K(X')) \neq P(X, X')] \leq \text{negl}(\lambda). \end{aligned}$$

► **Definition 5** (Double-Oracle-Robust PPH [4]). *If H is an Double-Oracle (DO) robust PPH for a property P , for any PPT adversary A ,*

$$\begin{aligned} & \mathbf{Adv}_H^{\text{DO-PPH}}(A) \\ & := \Pr[K \xleftarrow{\$} \mathcal{K}; A^{\mathcal{O}_{H.\text{Hash}}, \mathcal{O}_{H.\text{Eval}}} \rightarrow (X, X') : \text{Eval}(K, H_K(X), H_K(X')) \neq P(X, X')] \\ & \leq \text{negl}(\lambda). \end{aligned}$$

► **Definition 6** (Direct-Access Robust PPH [4]). *If H is a Direct-Access (DA) PPH for a property P , for any PPT adversary A ,*

$$\begin{aligned} & \mathbf{Adv}_H^{\text{DA-PPH}}(A) \\ & := \Pr[K \xleftarrow{\$} \mathcal{K}; (X, X') \leftarrow A(K) : \text{Eval}(K, H_K(X), H_K(X')) \neq P(X, X')] \leq \text{negl}(\lambda). \end{aligned}$$

A DA-robust PPH may be simply called a robust PPH.

In DA-PPH the adversary is given K , hence can also simulate $\mathcal{O}_{\text{Hash}}$ and $\mathcal{O}_{\text{Eval}}$.

For example, classical universal hash function [35] can be interpreted as a non-robust PPH for the collision property (i. e. $P(X, X') = 1$ iff $X = X'$). BLV19 provides in-depth discussions on these notions and relations to the existing ideas such as one-way communication protocols. They are not relevant to our work, so refer to BLV19 for more details.

2.6 Constructions of PPHs

As described in Introduction, BLV19 showed two constructions of PPH for Gap-Hamming predicate (Definition 1), which we call PPH1 and PPH2. PPH1 is entirely based on a collision-resistant hash function. The construction takes hash values for multiple subsets of the input, where the subsets are specified by a class of bipartite expander graph, and the output is the concatenation of these hash values (see Sections 4.2 and 5 for more details). While intuitive, its large hash size is problematic. PPH2 supports a smaller gap and better efficiency than PPH1 (although the compression rate is constant for both schemes) and is based on a new assumption related to the hardness of syndrome decoding of LDPC codes. BLV19 stipulated research on PPHs. Fleischhacker and Simkin [18] showed a PPH for the (exact, rather than Gap) Hamming distance predicate with a hash length for threshold t , which is much better than PPH1. The security is based on a new bilinear discrete-logarithm assumption in pairing friendly curves. Fleischhacker et al. [17] proposed a PPH for the Hamming distance predicate whose security is proved under a standard lattice hardness assumption while having a larger hash size than [18]. These studies have been done with PPH2 in mind.

Compared to PPH2, PPH1 has not been studied since the proposal, possibly for its larger hash size. However, hash-based constructions may be worth studying for their simplicity, computational efficiency, and classical security. The reduction to classical symmetric-key cryptographic assumption is also meaningful in the context of post-quantum cryptography. These observations made us turn our attention to PPH1. As a result, we discover a connection between the idea of PPH1 and the classical Combinatorial Group Testing.

3 Combinatorial Group Testing

We provide some basic ideas about Combinatorial Group Testing (CGT). It is a method to detect defectives among a set of items by using a set of *group tests*. A group test specifies a subset \mathcal{S} of the whole samples \mathcal{M} and returns a binary output indicating if \mathcal{S} contains a

defective. Originally Dorfman invented CGT in WWII [15] to effectively find blood samples infected by syphilis. We write $\mathcal{M} = [n]$ and $\mathcal{I} \subseteq \mathcal{M}$ to denote the (indices of) whole items and the defectives, and in the classical setting we know $|\mathcal{I}| \leq d$ for some $d \in [n]$. In the non-adaptive setting, which is relevant for our case, the set of k group tests is determined by a $k \times n$ binary matrix \mathbb{M} , where $\mathbb{M}_{i,j} = 1$ means j -th item is included in the i -th test.

The fundamental goal of CGT is to detect all the defectives, using as few tests as possible. In the case of non-adaptive CGT, we detect all the defectives when the test matrix satisfies a property called d -disjunct [15].

► **Definition 7.** A $k \times n$ binary matrix \mathbb{M} is d -disjunct if, for any $\mathcal{S} \subseteq [n]$ and $|\mathcal{S}| \in [d]$, $\mathbb{M}_{*,j} \not\subseteq \bigvee_{h \in \mathcal{S}} \mathbb{M}_{*,h}$ holds for any $j \notin \mathcal{S}$. That is, a sum of any distinct $i \leq d$ columns of \mathbb{M} does not cover any other column.

An $n \times n$ identity matrix is n -disjunct. When the test matrix is d -disjunct and $|\mathcal{I}| \leq d$, it is known that the so-called *naive decoder* can detect all the defectives. Algorithm 1 shows the naive decoder for n items, k tags using $k \times n$ test matrix \mathbb{M} , taking the test result $R = (R_1, \dots, R_k) \in \{0, 1\}^k$, where $R_i = 1$ denotes that i -th test is positive (indicating the test contains defectives).

■ **Algorithm 1** Naive Decoder.

```

1: procedure DECODE( $\mathbb{M}, R$ )           ▷  $k \times n$  binary matrix  $\mathbb{M}$ , test result  $R \in \{0, 1\}^k$ 
2:    $\mathcal{P} \leftarrow [n]$ 
3:   for  $i = 1, \dots, k$  do
4:     if  $R_i = 1$  then
5:        $\mathcal{P} \leftarrow \mathcal{P} \setminus \mathbb{M}_i$ 
6:   return  $\mathcal{P}$ 

```

Naturally, we want to minimize the number of rows (k) of a d -disjunct matrices given d and n . Disjunct matrix has extensively been studied from the viewpoint of combinatorics or designs or codes; refer to Du and Hwang [15] for classical constructions and the bounds. Most importantly it is known that $k = O(d^2 \log n)$. Porat and Rothchild (PR11) [32], and Cheraghchi and Nakos (CN20) [9] presented polynomial-time constructions that achieve this bound, i. e., these constructions are order-optimal.

Since the inception, the most typical application of CGT is biology, such as DNA screening [28]. One can also find a recent surge of research on applying CGT to COVID-19 PCR testing (e. g. [27] and there are lots of many others). Moreover, CGT has many other applications in the field of computer science, such as image compression [21], streaming computation [10], digital forensics [19] and (aggregate) MAC with detection capability [25, 26, 20, 30].

4 Constructing PPHs from CGT

4.1 CGT-Hash as Direct-Access Robust PPH

4.1.1 Defining CGT-Hash

We describe a class of CGT-based hash function (recall that this is to detect data corruptions rather than telling the binary verification result). For $\mathcal{X} = \mathcal{B}^n$, $\mathcal{B} = \{0, 1\}^b$ for some fixed b , let $G : \mathcal{K} \times [k] \times \mathcal{X} \rightarrow \mathcal{T}$ for $\mathcal{T} = \{0, 1\}^t$ be the atomic (keyed) hash function, and let \mathbb{M} be a $k \times n$ binary matrix. Taking G and \mathbb{M} as parameters, $\text{CGT-Hash}[G, \mathbb{M}] : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$ for $\mathcal{Y} = \mathcal{T}^k$ is defined as

$$\text{CGT-Hash}[G, \mathbb{M}](K, X) = (Y_1, \dots, Y_k), \quad Y_i = G(K, i, X_{|\mathbb{M}_i})$$

Note that \mathbb{M} is a fixed, public matrix. This could be used to detect corruptions on $X \in \mathcal{X}$: first we take $Y = \text{CGT-Hash}[G, \mathbb{M}](K, X)$ with key K and store Y to a tamper-free storage. Later, we take $Y' = \text{CGT-Hash}[G, \mathbb{M}](K, X')$ for X' which is a possibly corrupted version of X by the adversary, and try to detect the locations of corruptions (namely, $\mathcal{D}(X, X')$) via Y and Y' . CGT-Hash is an application of CGT described in Section 3. From the definition of d -disjunct matrix, if X consists of n items and the adversary can corrupt at most d items, by using d -disjunct matrix \mathbb{M} and the naive decoder (Alg. 1), we can detect $\mathcal{D}(X, X')$ without an error, if G is secure. CGT-Hash will bring more useful information than just taking one hash for the entire X . Moreover, it significantly reduces the size of tamper-free storage than taking hash values for each item. In other words, it allows a trade-off between the size of hash values and the resolution of detection.

Effectively the same idea has been seen in *Corruption-localizing hashing* by Crescenzo et al. [14] and follow-up works [11, 3], where different attack models and different decoders are used. From a practical viewpoint, a basic form of hash-based corruption detection (say by taking hash values for all the files in a server) has been extensively used by major commercial integrity protection/management products, such as TripWire¹ or Splunk². In principle, the use of CGT-Hash will reduce the size of tamper-free storage in these applications.

4.1.2 Connection to PPH

We can use CGT-Hash as a PPH for Hamming distance property. By writing $H(K, *)$ to denote $\text{CGT-Hash}[G, \mathbb{M}](K, *)$, Hash procedure just performs $H_K(X)$ for input X , and the Eval procedure taking (K, Y, Y') s.t. $Y = H(K, X)$ and $Y' = H(K, X')$ for some $X \neq X' \in \mathcal{X}$ first obtains $\mathcal{D}(Y, Y')$ and performs the naive decoder. If decoder output is larger than d , Eval returns 1 and 0 otherwise. Formally, the following theorem shows that CGT-Hash is a DA-robust PPH for the Hamming distance predicate if G (given the random key K) is collision-resistant.

► **Theorem 8.** *In the aforementioned problem setting, if \mathbb{M} is d -disjunct and G is collision-resistant, $\text{CGT-Hash}[G, \mathbb{M}]$ is a DA-robust PPH for the (generalized) Hamming predicate with threshold d (HAM^d) over \mathcal{B}^n .*

Proof. In the game of DA-Robust PPH, suppose the adversary \mathbf{A} uses q Hash queries, with total σ blocks and τ time. Note that Eval oracle does not need the key, hence this can be accessed for free. The i -th Hash query and the corresponding tag vector are denoted by $X^{(i)}$ and $T^{(i)} = (T_1^{(i)}, \dots, T_k^{(i)})$. Let Cons (for consistency) be the event that $[\forall i, j \in [q], i \neq j, \forall h \in [k] : X_{|\mathbb{M}_h}^{(i)} \neq X_{|\mathbb{M}_h}^{(j)} \Leftrightarrow T_h^{(i)} \neq T_h^{(j)}]$.

It is clear that to invoke

$$T_h^{(i)} = T_h^{(j)}$$

for some distinct $i, j \in [q]$ and $h \in [k]$ s.t. $X_{|\mathbb{M}_h}^{(i)} \neq X_{|\mathbb{M}_h}^{(j)}$, \mathbf{A} needs to invoke a collision on G . If $T_h^{(i)} \neq T_h^{(j)}$ we must have $X_{|\mathbb{M}_h}^{(i)} \neq X_{|\mathbb{M}_h}^{(j)}$ for any G . Thus, invalidation of Cons is equivalent to finding a non-trivial collision on G .

We also observe that, as long as Cons holds, the problem exactly matches with non-adaptive CGT with *error-free* tests. Moreover, whenever the difference is larger than d , Eval will always detect this fact because the decoder output is always larger than d from the

¹ <https://www.tripwire.com/>

² <https://www.splunk.com/>

property of the naive decoder (i. e., it never evicts non-corrupted items). A group testing scheme with this property is called a strict group testing [13], and any non-adaptive CGT scheme with a naive decoder and a disjunct test matrix is a strict group testing. This means that HAM^d property is always preserved. Therefore,

$$\text{Adv}_{\text{CGT-Hash}[G, \mathbb{M}]}^{\text{DA-PPH}}(\mathbf{A}) \leq \text{Adv}_G^{\text{Coll}}(\mathbf{B})$$

holds for some collision adversary \mathbf{B} against G using qt queries with total σ blocks, $\tau' = \tau + O(\sigma)$ time. \blacktriangleleft

By using order-optimal d -disjunct matrices derived by PR11 or CN20, the compression rate of $\text{CGT-Hash}[G, \mathbb{M}]$ is kt/bn for $k = O(d^2 \log n)$. The evaluation Eval runs in time $O(nk)$ relying upon the naive decoder. We note that, the evaluation complexity can be greatly improved if *efficient* decoder is available for the test matrix. For example, the disjunct matrix construction by Indyk et al. [23] achieves $k = O(d^2 \log n)$ tests with $O(d^4 \log n)$ decoding time, and that by Ngo et al. [29] achieves $k = O(d^2 \log n + d \log n \cdot \log \log_d n)$ tests with $O(k \log^2 n)$ decoding time. The construction by CN20 also has an efficient decoder. More precisely, it presented a construction achieving $k = O(d^2 \min\{\log n, (\log_d n)^2\})$ tests with $O(k + d \log^2(n/d))$ decoding time.

4.2 Beyond Predicate

Theorem 8 is easily extended to a more informative property defined below:

$$\text{DETECT}^d(X, X') := \begin{cases} \mathcal{D}(X, X') & \text{if } \text{Hd}(X, X') \leq d \\ 0 & \text{otherwise.} \end{cases}$$

The DETECT^d property tells exact indices of different components when the difference is smaller than the threshold, and otherwise tells that the difference is indeed larger than d . Such a property will facilitate further investigation and is useful for some applications, e. g., biometrics or digital forensics.

4.2.1 Comparison with PPH1

BLV19's first construction (PPH1) is a PPH for the Gap-Hamming property with parameter (d, ϵ) , taking $b = 1$. It utilizes a bipartite expander graph with a certain regularity. Using a hash G of t -bit output, its output is $k \cdot t$ bits, where $k = O(n/\log n)$ [4, Lemma 47], $d = O(n/t)$. Direct comparison is not possible, however, CGT-Hash (for the same input, using the same G) implements a robust PPH for the exact Hamming distance predicate with $k \cdot t$ -bit output, where $k = O(d^2 \log n)$. Thus, the size reduces by a factor of about $n/(d^2 \log^2 n)$, which is non-negligible when we focus on the case $d \ll n$.

We remark that PPH for relatively small d is still usable: it allows to check if two inputs are close even the adversary has full knowledge of the hashing scheme, which is a typical and intuitive application of PPH to (e. g.) biometrics.

4.3 CGT-MAC as Double-Oracle Robust PPH

We consider a MAC-based counterpart to CGT-Hash . Namely, using a variable-input-length pseudorandom function (PRF) $F : \mathcal{K} \times [k] \times \mathcal{X} \rightarrow \mathcal{T}$,

$$\text{CGT-MAC}[F, \mathbb{M}](K, X) = (Y_1, \dots, Y_k), \quad Y_i = F(K, i, X_{[\mathbb{M}_i]})$$

■ **Table 1** Comparison of Hash-based PPHs. We assume that the baseline hash G runs in $O(n)$ time for n -bit/item input.

	Property	Output	Hash time	Eval time
PPH1 [4]	GapHAM $^{d,\epsilon}$	$O(\frac{nt}{\log n})$	$O(\frac{n^2}{\log n})$	$O(\frac{n}{\log n})$
CGT-Hash	HAM d , DETECT d	$O(d^2 t \log n)$	$O(d^2 n \log n)$	$O(d^2 n \log n)$ ¹⁾

1) Improved to $\text{poly}(d^2 \log n)$ if efficient decoders are available (see Section 4.1)

as in the same manner to CGT-Hash. Eval is identical to that of CGT-Hash and does not involve the key. We cannot give K to the adversary, as there is no security guarantee for PRF once the key is known. Thus Robust PPH is impossible in the first place. The next strong robustness notion is DO-PPH. Assuming F is a monolithic, black-box PRF, proving DO-PPH security for the above CGT-MAC is almost identical to the DA-PPH security proof of CGT-Hash, hence we omit it here. Such an instantiation of CGT-MAC scheme is the same as the core component of what Goodrich et al. [19] proposed for data forensics applications.

To achieve a further efficiency improvement from the above scheme, we take a CGT-based MAC scheme proposed by Minematsu [25], called GTM. It is a blockcipher-based scheme, however, instead of a conventional variable-input-length PRF (or MAC) such as CMAC, GTM uses a variant of vector-input PRF [34] that accepts an empty string as a component of an input vector, where a vector consists of (a fixed number of) bit strings. Namely, if the i -th row of the test matrix is $(1, 0, 1)$, the input to the underlying vector-input PRF is (X_1, ε, X_3) , rather than conventional (X_1, X_3) , together with additional index i . Such a PRF is easily built on any variable-input-length PRF via an input encoding, but if we adopt a structure similar to PMAC [1, 33] and simply skip the computation for j -th component if it is an empty string, it allows a significant computational improvement in our application while maintaining appropriate security. Concretely, CGT-MAC uses F which is defined as

$$F(K, i, X_{\mathbb{M}_i}) = E' \left(K_2, i, \sum_{j \in \mathbb{M}_i} E(K_1, j, X_j) \right). \quad (1)$$

To learn why this F can be interpreted as a vector-input PRF, please refer to [25]. The above F is based on two PRFs, $E : \mathcal{K}_1 \times [k] \times \mathcal{X} \rightarrow \mathcal{C}$ for $\mathcal{C} = \{0, 1\}^c$ for some positive c , and $E' : \mathcal{K}_2 \times [k] \times \mathcal{C} \rightarrow \mathcal{T}$. The key is $K = (K_1, K_2)$. As mentioned above, it is closely related to PMAC. Let $\text{GTM}[F, \mathbb{M}]$ denote the PPH using F and the test matrix \mathbb{M} that is d -disjunct. The Hash and Eval procedures are defined as in the same manner to CGT-Hash. We show that $\text{GTM}[F, \mathbb{M}]$ is a DO-PPH for HAM d .

► **Theorem 9.** *Let F be a keyed function built on E, E' as Eq. (1). If \mathbb{M} is d -disjunct and E, E' are PRFs, $\text{GTM}[F, \mathbb{M}]$ is a DO-robust PPH for the (generalized) Hamming predicate with threshold d (HAM d) over \mathcal{B}^n .*

Proof. Since Eval is a keyless function, the proof approach is almost identical to that of CGT-Hash, and the proof is similar to that of [25]. We first analyze the information-theoretic setting. Let R_F be a function having the same domain and range as F , and uses two b -bit URFs R, R' instead of E and E' . Using a similar argument as the proof of Theorem 8, we observe that breaking the DO-PPH security implies the invalidation of Cons event. Let A be an adversary using q queries to $\text{GTM}[R_F, \mathbb{M}].\text{Hash}$ oracle and infinite computation time (note that queries to the key-less $\text{GTM}[R_F, \mathbb{M}].\text{Eval}$ oracle are simulatable hence we omit).

We have

$$\mathbf{Adv}_{\text{GTM}[R_F, \mathbb{M}]}^{\text{DO-PPH}}(\mathbf{A}) \leq \mathbf{Adv}_{\text{GTM}[R_F, \mathbb{M}].\text{Hash}}^{\text{Cons}}(\mathbf{A}'),$$

for some adversary \mathbf{A}' using q Hash queries. Let \mathbf{Ideal} be the idealized version of $\text{GTM}[R_F, \mathbb{M}]$, namely using an independent URF instead of $F(K, i, *)$, for each $i \in [k]$. Using the standard hybrid argument, we have

$$\mathbf{Adv}_{\text{GTM}[R_F, \mathbb{M}]}^{\text{Cons}}(\mathbf{A}') \leq \mathbf{Adv}_{\text{GTM}[R_F, \mathbb{M}].\text{Hash}, \mathbf{Ideal}.\text{Hash}}^{\text{IND}}(\mathbf{B}) + \mathbf{Adv}_{\mathbf{Ideal}.\text{Hash}}^{\text{Cons}}(\mathbf{A}'') \quad (2)$$

Since each component function of \mathbf{Ideal} is an independent random function, as in the case of CGT-Hash, the last term of the right hand side of Eq. (2) is reduced to the collision probability (in the same component function; we do not have to care about collision between different components). Let $\mathbf{Adv}_{\mathbf{Ideal}.\text{hash}}^{\text{Coll}}(\mathbf{A})$ denote such a collision probability by \mathbf{A} . We have

$$\mathbf{Adv}_{\mathbf{Ideal}.\text{hash}}^{\text{Cons}}(\mathbf{A}'') \leq \mathbf{Adv}_{\mathbf{Ideal}.\text{hash}}^{\text{Coll}}(\mathbf{A}'') \leq \frac{q^2}{2^{t+1}}, \quad (3)$$

where the last equation follows from the standard collision analysis. For the first term of the r.h.s. of Eq. (2), define $X^{(i)}$ and $T^{(i)} = (T_1^{(i)}, \dots, T_k^{(i)})$ as the i -th Hash query and its response, and let $S^{(i)} = (S_1^{(i)}, \dots, S_k^{(i)}) \in (\{0, 1\}^c)^k$ be the inputs to R' , i.e., $S_j^{(i)} = \sum_{h \in \mathbb{M}_j} R(K_1, h, X_h^{(i)})$. Based on the analysis [25] (which is also a variant of PMAC's security proof [33]), we show that the indistinguishability is reduced to the collision between S , namely the first term of the r.h.s. of Eq. (2) is bounded by

$$\binom{q}{2} \cdot \Pr[\exists h, h' \in [q], i \in [k] : X_{|\mathbb{M}_i}^{(h)} \neq X_{|\mathbb{M}_i}^{(h')}, S_i^{(h)} = S_i^{(h')}] \leq \binom{q}{2} \cdot \frac{1}{2^c} \leq \frac{q^2}{2^{c+1}}, \quad (4)$$

where the probability is defined by \mathbf{A}'' and $\text{GTM}[R_F, \mathbb{M}]$. The second inequality follows from the observation that each component of keyed message hashing procedure, $Y \leftarrow \sum_{j \in \mathbb{M}_i} R(j, X_j)$, is XOR-universal, that is, for any $X^{(h)}$ and $X^{(h')}$ and $i \in [k]$ such that $X_{|\mathbb{M}_i}^{(h)} \neq X_{|\mathbb{M}_i}^{(h')}$, the sum $S_i^{(h)} \oplus S_i^{(h')}$ is uniform. Note that we do not have to count collisions between different component functions thanks to the “finalization” by R' taking the index of component as a part of input (often called a domain separation). From Eqs. (2), (3), and (4), we have

$$\mathbf{Adv}_{\text{GTM}[R_F, \mathbb{M}]}^{\text{DO-PPH}}(\mathbf{A}) \leq \frac{q^2}{2^{c+1}} + \frac{q^2}{2^{t+1}}. \quad (5)$$

Eq. (5) immediately tells that the computational security of $\text{GTM}[F, \mathbb{M}]$ assuming that E and E' are PRFs. More precisely, $\mathbf{Adv}_{\text{GTM}[F, \mathbb{M}]}^{\text{DO-PPH}}(\mathbf{A})$ for any \mathbf{A} with q Hash queries with t time is at most the sum of PRF advantages of E and E' (i.e., the advantage in distinguishing them from the URF), and the bound shown above. This bound is shown via the standard information-theoretic-to-computational hybrid. \blacktriangleleft

4.3.1 Reduced Hashing Time

A significant difference from the generic CGT-MAC is the time complexity for hashing: assuming F runs in $O(n)$ for n -block input (where a block is b -bit), which holds for the most of popular constructions such as HMAC or CMAC, $\text{CGT-MAC}[F, \mathbb{M}]$ needs $O(kn)$ time for hashing an input. In contrast, the hashing of GTM needs n calls of E and k calls of E' – hence $O(k + n)$ time – by caching the outputs of E (See [25]). If we use the parameters

of Table 1, hashing time is reduced from $O(d^2 n \log n)$ to $O((d^2 \log n) + n)$, assuming the MAC runs in $O(n)$ for n input items. Typically $k \ll n$, as otherwise, the hashing does not compress well, hence this simple trick of [25] allows to reduce the cost of hashing to that of single MAC/PRF computation³. We also note that this computation cost does not depend on the contents of \mathbb{M} .

4.4 XOR-GTM

Minematsu and Kamiya [26] (MK19) proposed a new approach to CGT-based MAC, dubbed XOR-GTM. It enables to detect d corruptions among n data items using a significantly smaller number of MAC tags than $O(d^2 \log n)$, namely what CGT-MAC or GTM can achieve with a disjunct matrix.

We only briefly describe the scheme. The scheme has almost the same procedure as GTM, using a test matrix \mathbb{M} and PRF E , except that the final E' is a tweakable block cipher [24] taking $i \in [k]$ as a tweak. Here, a tweakable block cipher is an extension of a conventional block cipher that takes a chosen public value, called tweak, as a part of input in addition to the message block. On the detection of corruptions, it first decrypts the received tags using the inverse of E' . Next, it takes linear combinations of the decrypted tags, following the “extended” test matrix \mathbb{M}^R , which is a submatrix of row span of \mathbb{M} . It also takes the same combinations from the received message and compares the linear combinations. MK19 showed that if \mathbb{M}^R is d -disjunct, this scheme allows detecting of up to d corrupted items. Since the communication cost (number of tags) only depends on \mathbb{M} , not \mathbb{M}^R , this implies the communication cost can be possibly smaller than the limit of the number of rows of d -disjunct matrix. In other words, what is needed here is a disjunct matrix of small rank for \mathbb{M}^R . MK19 presented several such instances using error-correcting codes derived by finite geometry.

4.4.1 Robustness of XOR-GTM

It is natural to expect DO-PPH security for XOR-GTM in the same manner as GTM. However, unlike GTM, the Evaluation oracle of XOR-GTM involves the key. This fact makes the security proof for DO-PPH not directly derived from the original proof of XOR-GTM. Most importantly, the original security proof of XOR-GTM requires that the forward evaluation of E' is pseudorandom, but does not require the pseudorandomness of the inverse. Technically speaking, MK19 requires Tweakable Pseudorandom Permutation (TPRP, for CPA-secure TBC) but not Tweakable Strong PRP (TSPRP, for CCA-secure TBC). Proving DO-PPH security appears to require the latter. This difference comes from the fact that the adversary in the game defined for XOR-GTM (called Decoder Unforgeability, DUF) in MK19 does not have freedom in choosing the tag given to the decoder, while the adversary in the game of DO-PPH notion has no restriction on the choice of tags given to the evaluation oracle. Considering that XOR-GTM significantly reduces the number of tags beyond $O(d^2 \log n)$, proving DO-PPH for XOR-GTM assuming TSPRP for E' is an interesting open question.

³ A more detailed comparison of GTM and CGT-MAC is possible by instantiating F as a variant of PMAC using b -bit block cipher, and letting $c = b$ and E and E' be the same b -bit block cipher. CGT-MAC needs $w = O(kn)$ block cipher calls, where w denotes the weight of \mathbb{M} , and GTM needs $(k + n)$ calls.

5 Expander Graph-based Constructions for PPH and CGT

As stated earlier, PPH1 relies on a bipartite expander graph. The graph has node sets $L = [n]$ and $R = [k]$, by defining the i -th test as its neighbors in L for $i \in R$. We find the relationship between this expander-based PPH with a variant of CGT problem that aims at determining the number of defectives rather than identifying them. This problem has been actively studied for its practical and theoretical importance, say [8, 6, 16, 12] for example.

Among these studies, a CGT scheme proposal by Bshouty and Haddad-Zaknoon [7] (BH21) has an interesting overlap with BLV19's PPH. In detail, they first proposed the construction of matrix using a bipartite expander matrix in the same manner as BLV19. Let $\mathcal{M} = [n]$ be the item set and $\mathcal{I} \subseteq \mathcal{M}$ be the set of defectives. The proposed matrix allows a distinguisher $A(\ell, \Delta)$ who uses $m(\ell, \Delta)$ tests to distinguish whether $|\mathcal{I}| < \ell/\Delta^2$ (A returns 0) or $|\mathcal{I}| \geq \ell/\Delta$ (A returns 1) without error, for certain parameters ℓ and $\Delta > 1$. Given the maximum value for $|\mathcal{I}|$, $|\mathcal{I}| < D$, the proposed test (for n items and the defective set \mathcal{I}) runs $A(D/\Delta^i, \Delta)$ for $i = 0, 1, \dots, \lceil \log D/\log \Delta \rceil$, and outputs $\hat{d} = D/\Delta^{i+1}$ for the smallest i such that $A(D/\Delta^i, \Delta) \rightarrow 1$. The \hat{d} guarantees $|\mathcal{I}|/\Delta \leq \hat{d} \leq |\mathcal{I}|\Delta$. Using a known construction of bipartite expander graphs, the number of tests is $(D/\Delta^2) \cdot 2^{O(\log^3(\log n))}$. This construction of $A(\ell, \Delta)$ in BH21 is essentially identical to the idea of PPH1. We note that the problem setting of BH21 requires D as prior information on $|\mathcal{I}|$ and if D is not known there is no non-trivial testing to estimate $|\mathcal{I}|$ [5]. BH21 can be a variant of hash-based PPH whose goal is to give an estimate of the generalized Hamming distance with a predetermined margin, however, such a variant is not covered by the original definition of PPH by BLV19. Therefore, we cannot expect a direct translation from CGT scheme into a PPH as we did in the previous sections. It may be interesting to further explore the relationships.

6 Conclusions

This paper has shown the connection between the property-preserving hash (PPH) functions and the adversarial error detection schemes combining the classical Combinatorial Group Testing (CGT) and hash/MAC functions. Our findings brought several implications and improvements to the hash/MAC-based PPHs, which has been initially proposed by Boyle et al. [4] but largely overlooked since the proposal. PPH is still in its infancy as a research field, and we believe that the connection we have discovered will be useful for developing PPH. Moreover, we hope that our results will also encourage the CGT research community to look into research on PPH.

References

- 1 John Black and Phillip Rogaway. A block-cipher mode of operation for parallelizable message authentication. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 384–397. Springer, Heidelberg, April / May 2002. doi:10.1007/3-540-46035-7_25.
- 2 Burton H. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Commun. ACM*, 13(7):422–426, 1970.
- 3 Annalisa De Bonis and Giovanni Di Crescenzo. Combinatorial Group Testing for Corruption Localizing Hashing. In *COCOON*, volume 6842 of *Lecture Notes in Computer Science*, pages 579–591. Springer, 2011.
- 4 Elette Boyle, Rio LaVigne, and Vinod Vaikuntanathan. Adversarially robust property-preserving hash functions. In Avrim Blum, editor, *ITCS 2019*, volume 124, pages 16:1–16:20. LIPIcs, January 2019. doi:10.4230/LIPIcs.ITCS.2019.16.

- 5 Nader H. Bshouty, Vivian E. Bshouty-Hurani, George Haddad, Thomas Hashem, Fadi Khoury, and Omar Sharafy. Adaptive Group Testing Algorithms to Estimate the Number of Defectives. In *ALT*, volume 83 of *Proceedings of Machine Learning Research*, pages 93–110. PMLR, 2018.
- 6 Nader H. Bshouty, George Haddad, and Catherine A. Haddad-Zaknoon. Bounds for the Number of Tests in Non-adaptive Randomized Algorithms for Group Testing. In *SOFSEM*, volume 12011 of *Lecture Notes in Computer Science*, pages 101–112. Springer, 2020.
- 7 Nader H. Bshouty and Catherine A. Haddad-Zaknoon. Optimal deterministic group testing algorithms to estimate the number of defectives. *Theor. Comput. Sci.*, 874:46–58, 2021.
- 8 Chao L. Chen and William H. Swallow. Using Group Testing to Estimate a Proportion, and to Test the Binomial Model. *Biometrics*, 46(4):1035–1046, 1990.
- 9 Mahdi Cheraghchi and Vasileios Nakos. Combinatorial group testing and sparse recovery schemes with near-optimal decoding time. In *61st FOCS*, pages 1203–1213. IEEE Computer Society Press, November 2020. doi:10.1109/FOCS46700.2020.00115.
- 10 Graham Cormode and S. Muthukrishnan. What’s hot and what’s not: tracking most frequent items dynamically. *ACM Trans. Database Syst.*, 30(1):249–278, 2005.
- 11 Giovanni Di Crescenzo and Gonzalo R. Arce. Data Forensics Constructions from Cryptographic Hashing and Coding. In *IWDW*, volume 7128 of *Lecture Notes in Computer Science*, pages 494–509. Springer, 2011.
- 12 Peter Damaschke and Azam Sheikh Muhammad. Competitive Group Testing and Learning Hidden Vertex Covers with Minimum Adaptivity. *Discret. Math. Algorithms Appl.*, 2(3):291–312, 2010.
- 13 Peter Damaschke, Azam Sheikh Muhammad, and Gábor Wiener. Strict group testing and the set basis problem. *J. Comb. Theory, Ser. A*, 126:70–91, 2014.
- 14 Giovanni Di Crescenzo, Shaoquan Jiang, and Reihaneh Safavi-Naini. Corruption-localizing hashing. In Michael Backes and Peng Ning, editors, *ESORICS 2009*, volume 5789 of *LNCS*, pages 489–504. Springer, Heidelberg, September 2009. doi:10.1007/978-3-642-04444-1_30.
- 15 D. Du and F. Hwang. *Combinatorial Group Testing and Its Applications*. Applied Mathematics. World Scientific, 2000.
- 16 Moein Falahatgar, Ashkan Jafarpour, Alon Orlitsky, Venkatadheeraj Pichapati, and Ananda Theertha Suresh. Estimating the number of defectives with group testing. In *ISIT*, pages 1376–1380. IEEE, 2016.
- 17 Nils Fleischhacker, Kasper Green Larsen, and Mark Simkin. Property-Preserving Hash Functions from Standard Assumptions. *CoRR*, abs/2106.06453, 2021. arXiv:2106.06453.
- 18 Nils Fleischhacker and Mark Simkin. Robust property-preserving hash functions for hamming distance and more. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part III*, volume 12698 of *LNCS*, pages 311–337. Springer, Heidelberg, October 2021. doi:10.1007/978-3-030-77883-5_11.
- 19 Michael T. Goodrich, Mikhail J. Atallah, and Roberto Tamassia. Indexing information for data forensics. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *ACNS 05*, volume 3531 of *LNCS*, pages 206–221. Springer, Heidelberg, June 2005. doi:10.1007/11496137_15.
- 20 Shoichi Hirose and Junji Shikata. Aggregate Message Authentication Code Capable of Non-Adaptive Group-Testing. *IEEE Access*, 8:216116–216126, 2020.
- 21 Edwin S. Hong and Richard E. Ladner. Group testing for image compression. *IEEE Trans. Image Process.*, 11(8):901–911, 2002.
- 22 Piotr Indyk and Rajeev Motwani. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *STOC*, pages 604–613. ACM, 1998.
- 23 Piotr Indyk, Hung Q. Ngo, and Atri Rudra. Efficiently Decodable Non-adaptive Group Testing. In *SODA*, pages 1126–1142. SIAM, 2010.
- 24 Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable block ciphers. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 31–46. Springer, Heidelberg, August 2002. doi:10.1007/3-540-45708-9_3.

- 25 Kazuhiko Minematsu. Efficient message authentication codes with combinatorial group testing. In Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl, editors, *ESORICS 2015, Part I*, volume 9326 of *LNCS*, pages 185–202. Springer, Heidelberg, September 2015. doi:10.1007/978-3-319-24174-6_10.
- 26 Kazuhiko Minematsu and Norifumi Kamiya. Symmetric-key corruption detection: When XOR-MACs meet combinatorial group testing. In Kazue Sako, Steve Schneider, and Peter Y. A. Ryan, editors, *ESORICS 2019, Part I*, volume 11735 of *LNCS*, pages 595–615. Springer, Heidelberg, September 2019. doi:10.1007/978-3-030-29959-0_29.
- 27 Leon Mutesa, Pacifique Ndishimye, Yvan Butera, Jacob Souopgui, Annette Uwineza, Robert Rutayisire, Ella Larissa Nduricimpaye, Emile Musoni, Nadine Rujeni, Thierry Nyatanyi, Edouard Ntagwabira, Muhammed Semakula, Clarisse Musanabaganwa, Daniel Nyamwasa, Maurice Ndashimye, Eva Ujeneza, Ivan Emile Mwikarago, Claude Mambo Muvunyi, Jean Baptiste Mazarati, Sabin Nsanzimana, Neil Turok, and Wilfred Ndifon. A pooled testing strategy for identifying sars-cov-2 at low prevalence. *Nature*, 589(7841):276–280, January 2021. doi:10.1038/s41586-020-2885-5.
- 28 Hung Q. Ngo and Ding-Zhu Du. A Survey on Combinatorial Group Testing Algorithms with Applications to DNA Library Screening. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 2000.
- 29 Hung Q. Ngo, Ely Porat, and Atri Rudra. Efficiently Decodable Error-Correcting List Disjunct Matrices and Applications - (Extended Abstract). In *ICALP (1)*, volume 6755 of *Lecture Notes in Computer Science*, pages 557–568. Springer, 2011.
- 30 Yoshinori Ogawa, Shingo Sato, Junji Shikata, and Hideki Imai. Aggregate message authentication codes with detecting functionality from biorthogonal codes. In *ISIT*, pages 868–873. IEEE, 2020.
- 31 Rasmus Pagh and Flemming Friche Rodler. Cuckoo hashing. *J. Algorithms*, 51(2):122–144, 2004.
- 32 Ely Porat and Amir Rothschild. Explicit Nonadaptive Combinatorial Group Testing Schemes. *IEEE Trans. Inf. Theory*, 57(12):7982–7989, 2011.
- 33 Phillip Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In Pil Joong Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 16–31. Springer, Heidelberg, December 2004. doi:10.1007/978-3-540-30539-2_2.
- 34 Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 373–390. Springer, Heidelberg, May / June 2006. doi:10.1007/11761679_23.
- 35 Mark N. Wegman and Larry Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22:265–279, 1981.