The Spammed Code Offset Method

Boris Škorić and Niels de Vreede

Abstract

Helper data schemes are a security primitive used for privacy-preserving biometric databases and Physical Unclonable Functions. One of the oldest known helper data schemes is the Code Offset Method (COM). We propose an extension of the COM: the helper data is accompanied by many instances of fake helper data that is drawn from the same distribution as the real one. While the adversary has no way to distinguish between them, the legitimate party has more information and can *see the difference. We use an LDPC code in order to improve the efficiency of the legitimate party's selection procedure.*
Our construction provides a new kind of trade-off: more effective use of the source entropy, at the price of increased helper data storage. We give a security analysis in terms of Shannon entropy and order-2 Rényi entropy.

1 Introduction

1.1 Helper Data Systems

The past decade has seen a lot of interest in a field that can be characterized as 'security with noisy data'. In several security applications it is necessary to *reproducibly* extract secret data from *noisy* measurements on a physical system. One such application is the privacy-preserving storage of biometric data. Analogously to password hashing, one can store biometric data in hashed form in order to prevent inside attackers from learning what the enrolled biometric features look like. Another application is read-proof storage of cryptographic keys using Physical Unclonable Functions (PUFs) [16, 17, 14]. Many types of digital memory can be considered insecure because of the large inbuilt redundancies needed to ensure reliable readout. PUFs provide an alternative way to store keys, namely in analog form, which allows the designer to exploit the inscrutability of analog physical behavior. Keys stored in this way are sometimes referred to as Physically Obfuscated Keys (POKs) [8].
In both the biometrics and the PUF/POK application, one faces the problem that some form of error correction has to be done, but under the constraint that the redundancy data, which is considered to be visible to attackers, does not reveal too much information about the secret extracted from the physical measurement. The problem is solved by a special security primitive, the *Helper Data System* (HDS).
A HDS in its most general form is shown in Fig. 1. The `Enroll` procedure takes as arguments a measurement X and (optionally) a random value R. It outputs a secret S and Helper Data W. The helper data is stored. In the reconstruction phase, a fresh measurement X' is obtained. Typically X' is a noisy version of X, i.e. close to X but not necessarily identical. The `Rec` (reconstruction) procedure takes X' and W as input. It outputs $\hat{S}$, an estimate of S. If X' is not too noisy then $\hat{S} = S$.

Two special cases of the general HDS are the Secure Sketch (SS) and the Fuzzy Extractor (FE) [6].

- The Secure Sketch has $S = X$ (and $\hat{S} = \hat{X}$, an estimator for X). If X is not uniformly distributed, then S is not uniform. The SS is suitable for privacy-preserving biometrics, where high entropy of S (given W) is required, but not uniformness.
- The Fuzzy Extractor has a (nearly) uniform S given W. The FE is typically used for extracting keys from PUFs and POKs.

There exists a generic construction to create a FE out of a SS: hashing the output of the SS using a Universal Hash Function (UHF) [3, 15, 11].

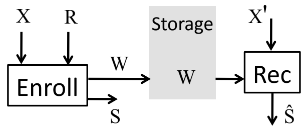

Figure 1: *Data flow in a generic Helper Data System.*

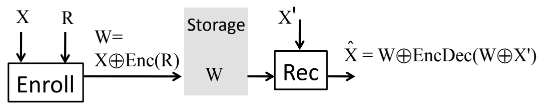

Figure 2: *The Code Offset Method employed as a Secure Sketch.*

1.2 The Code Offset Method

One of the oldest known SS constructions is the Code Offset Method (COM) [10, 6]. Here X is a binary string, say of length n, with probability distribution ρ. The construction uses a linear error-correcting code that encodes k-bit messages as n-bit codewords. The encoding and decoding operations are denoted as `Enc` and `Dec` respectively. In Fig. 1 we take R uniformly drawn from $\{0,1\}^k$ and

$$W = X \oplus \texttt{Enc}(R) \quad ; \quad \hat{X} = W \oplus \texttt{Enc}(\texttt{Dec}(W \oplus X')). \quad (1)$$

This is depicted in Fig. 2.
If X is uniformly distributed on $\{0,1\}^n$ then the scheme is not only a SS but in fact also a FE; this holds because a uniform X gives rise to helper data W that leaks nothing about R. The formulas for the FE are: $S = R$ and $W = X \oplus \texttt{Enc}(R)$; $\hat{R} = \texttt{Dec}(X' \oplus W)$. Note that for uniform X the W reveals the syndrome of X, but nothing about R. Hence R can then be used as a cryptographic key.
In this paper we study the case where X is *not* uniformly distributed. A non-uniform X appears naturally, e.g. in the

case of Coating PUFs [16], where Gray-coded capacitance measurements are concatenated to form X. Typically not all the Gray code words are represented, which leads to non-uniformity.
Similarly, a biometric feature vector is often split up into near-inepependent components which each yield a small number of non-uniform bits.

1.3 Zero Leakage

For some sources X it is possible to define helper data that reveals *nothing* about S. This is sometimes called Zero Secrecy Leakage (ZSL) helper data. The information contained in X is split into two *independent* parts, one of which serves for error correction, and one for making the secret S. As we saw above, the COM with uniform X has the ZSL property. Another example is the quantile partitioning scheme [18] of Verbitskiy et al. for *continuous* X, and its generalization to non-uniform S [5].

1.4 Contributions and outline

In this paper we propose a simple modification of the Code Offset Method SS. The basic idea is to add a number (say $m-1$) of dummy helper data instances to the publicly stored enrollment data, and to randomly permute the list. All the instances are a priori indistinguishable from the point of view of the adversary, but the legitimate party possesses X', which allows for efficient selection of the correct helper data instance. This workload asymmetry improves the security. For small m, the attacker may simply try out all possibilities, which leads to an average attack effort of $(m+1)/2$ times the original one. For very large m this brute force attack is no longer feasible, and the attacker is forced to ignore the public data; in this way a new kind of 'zero leakage' is achieved, distinct from the ZSL of Section 1.3, namely public data that reveals practically nothing about X (as opposed to S).
The concept of 'spamming' the attacker in this way is very general and is applicable whenever there exists an efficient way of recognizing the correct W using X'. In this paper we show how the 'spamming' concept can be applied to the Code Offset Method. Our scheme requires the use of a linear error-correcting code with low-density parity check matrix (LDPC) in order to keep the legitimate party's workload low.
In Section 2 we introduce the notation and assumptions that we work with. In Section 3 we analyze the leakage of the ordinary Code Offset Method and briefly review the Leftover Hash Lemma. In Section 4 we present our new scheme, which we call the Spammed Code Offset Method (SCOM). Section 5 contains a security analysis of our scheme and a brief discussion of memory requirements. A discussion and conclusions are given in Section 6.

2 Notation and attacker model

Random variables are written with capitals, and their realizations in lower case. Vectors are in boldface; sets in calligraphic font. Concatenation is denoted as $||$. The notation $d_{\rm Hamm}(x,y)$ stands for the Hamming distance between x and y. The logarithm 'log' is defined in base 2. The natural logarithm is ln.
The Code Offset Method works with a *linear* code $\mathcal{C}$ that has n-bit code words and k-bit messages. The encoding and decoding algorithms associated with this code are denoted as `Enc` and `Dec` respectively. The algorithm for computing the syndrome is denoted as `Syn`.
We consider a POK whose output at enrollment is a bit string $X \in \{0,1\}^n$. The probability distribution of X is called ρ, and ρ is not necessarily uniform. The string R in Fig. 2 has length k. The helper data is called W. In the FE setting, the cryptographic key that is ultimately derived from the POK is denoted as $K \in \{0,1\}^\ell$.
We will use shorthand notation $p_{xw} = \Pr[X=x, W=w]$, $p_w = \Pr[W=w]$ and $p_{x|w} = \Pr[X=x|W=w]$, when it does not cause ambiguity. We define $q_z = \Pr[\texttt{Syn}(X)=z]$. The public data stored in nonvolatile memory is P.
The outcome of the POK measurement in the reconstruction phase is denoted as $X' \in \{0,1\}^n$. The X' is a noisy version of X, and in general does not have the same probability distribution as X. The estimator for K, derived from X' and the public data, is denoted as $\hat{K}$.
We will rely on a cryptographic hash function f. Furthermore we will use a Universal Hash Function $g(x,a)$, where the second argument is public auxiliary randomness.
The attacker model is summarized as follows. We distinguish between two scenarios:

1. Biometric database for authentication.
   The adversary can read but not manipulate the public data P. His aim is to learn as much about X as he can.[1]

2. Secure key storage with a POK.
   The adversary has access to the device which contains the POK. He cannot re-activate the device's enrollment mode of operation. The opacity of the POK, and the embedding of the POK in the device, prevent the adversary from reading out K from the POK. Furthermore, physical tampering with the POK is unerringly detected by the device at the reconstruction phase. The public data P is stored on the device in insecure nonvolatile memory. The adversary is able to read and to manipulate P. There is no Public Key Infrastructure that would allow the device to verify the authenticity of the public data. The adversary's main aim is to learn the POK key K. A secondary goal is to cause the device to accept a key other than K as the correct key.

In both scenarios the adversary is able to discern whether reconstruction is successful. No other side channels exist.

3 Analysis of the Code Offset Method

We consider the general case of a non-uniform distribution ρ, and review what is known about the leakage of the COM. We briefly discuss the required amount of compression in case one wants to build a FE based on the COM.

[1]He may exploit this knowledge in various ways: (i) Some part of X may reveal information about medical conditions. This is a privacy risk. (ii) Construct a fake biometric in order to pass authentication. This is a security risk. (iii) Cross-linking of people across different databases. This is a privacy risk.

Lemma 1 *The Code Offset Method has the following probabilities,*

$$\begin{aligned} p_{rw} &= 2^{-k}\rho(w\oplus \texttt{Enc}(r)) && (2)\\ p_w &= \frac{1}{2^k}\sum_{r\in\{0,1\}^k}\rho(w\oplus\texttt{Enc}(r)) = \frac{1}{2^k}\Pr[\texttt{Syn}X=\texttt{Syn}\,w] && (3)\\ p_{r|w} &= \frac{\rho(w\oplus\texttt{Enc}(r))}{\sum_{t\in\{0,1\}^k}\rho(w\oplus\texttt{Enc}(t))} && (4)\\ p_{xw} &= 2^{-k}\rho(x)\delta_{\texttt{Syn}(w),\texttt{Syn}(x)} && (5)\\ p_{x|w} &= \frac{\rho(x)\delta_{\texttt{Syn}(w),\texttt{Syn}(x)}}{\sum_{t\in\{0,1\}^k}\rho(w\oplus\texttt{Enc}(t))}. && (6)\end{aligned}$$

Proof: The R is drawn uniformly and thus each r has probability 2^{-k} of occurring. The probability $\Pr[W=w|R=r]$ is given by $\rho(w\oplus\texttt{Enc}(r))$. Multiplication of these two gives (2). Equation (3) follows by computing p_w as the marginal of p_{rw} by summing over r. Eq. (4) follows from $p_{r|w}=p_{rw}/p_w$. Finally, (5) and (6) follow from (2) and (4) by setting $x=w\oplus\texttt{Enc}(r)$, which is only possible if x and w have the same syndrome. □

Eq. (4) shows that in general $p_{r|w}\neq p_r$. Thus, W leaks information not only about $\texttt{Syn}(X)$, but also about R.

Lemma 2 *In the Code Offset Method it holds that*

$$\begin{aligned} \mathsf{H}(W) &= k+\mathsf{H}(\texttt{Syn}\,X) && (7)\\ \mathsf{H}(X,W) &= \mathsf{H}(X)+k && (8)\\ I(X;W) &= \mathsf{H}(\texttt{Syn}\,X). && (9)\end{aligned}$$

Proof: $\mathsf{H}(W)$ follows directly from (3), and $\mathsf{H}(X,W)$ from (5). The $I(X;W)$ is computed as $\mathsf{H}(X)+\mathsf{H}(W)-\mathsf{H}(X,W)$. □

In order to obtain a nearly uniform key K from X (or, equivalently, from R), one has to hash down to a smaller size (say ℓ): $K=g(X,A)\in\{0,1\}^\ell$. Here A is *public* auxiliary randomness that serves as a 'catalyst' for the UHF g.

Let U be a uniform variable on $\{0,1\}^\ell$. The relation between ℓ and the uniformity of K is given by the Leftover Hash Lemma (LHL) [9] and can be formulated as

$$\ell\le L_\varepsilon(X,W) \implies \mathbb{E}_w[\Delta(U;K|W=w)]\le\varepsilon \quad (10)$$

$$\text{with } L_\varepsilon(X,W)=\mathsf{H}_2(X|W)+1-2\log\frac{1}{\varepsilon}. \quad (11)$$

Eq. (10) states that the non-uniformity of K given W does not exceed ε as long as X has been sufficiently hashed down. The ℓ must not exceed the 'ε-extractable randomness' L_ε. The notation H_2 in (11) stands for the conditional Rényi entropy of order two and is defined as [7]

$$\begin{aligned} \mathsf{H}_2(X|W) &= -2\log Q_2(X|W)\\ Q_2(X|W) &= \mathbb{E}_w\sqrt{\textstyle\sum_x p_{x|w}^2} = \sum_w\sqrt{\textstyle\sum_x p_{xw}^2}, && (12)\end{aligned}$$

where $\mathbb{E}_w$ stands for the expectation value over W.

Note that the 'penalty' term $2\log\frac{1}{\varepsilon}$ in (11) depends only on ε, i.e. it depends not on the *improvement* of the uniformity but on the *final* uniformity. Because of this fact, the approach using UHFs can be quite wasteful.

Remark: Under some conditions [1] the factor 2 in the penalty term can be replaced by 1. Furthermore, the LHL can be sharpened a bit by considering *smooth* Rényi entropy [12, 13, 19]. Such details are beyond the scope of the current paper.

4 Our construction: the Spammed Code Offset Method

We first show a naive spamming approach, without efficient de-spamming at the reconstruction phase. Then we propose an efficient scheme, in two variants: one in the privacy-preserving biometrics context, the other in the secure key storage context.

The efficient scheme requires a linear block code with a *low-density* parity check matrix. The security and the storage requirements are analyzed in Section 5.

4.1 Naive approach

Algorithm E0: Enrollment, the naive way

```
1. Measure $X\in\{0,1\}^n$.
2. Draw $R\in\{0,1\}^k$ uniformly at random.
3. Compute helper data $W=X\oplus\texttt{Enc}(R)$.
4. For $j\in\{1,\ldots,m-1\}$ do:
   (a) Uniformly draw $\Sigma_j\in\{0,1\}^k$.
   (b) Draw $D_j\in\{0,1\}^n$ from the distribution $\rho$.
   (c) Compute $\Omega_j=D_j\oplus\texttt{Enc}(\Sigma_j)$.
5. Draw a random permutation $\pi$.
6. Construct a vector $\mathbf{\Omega}=\pi(\Omega_1,\cdots,\Omega_{m-1},W)$.
7. Compute $G=f(\mathbf{\Omega}||X)$.
8. Store public data $P=(\mathbf{\Omega},G)$.
```

Algorithm R0: Reconstruction, the naive way

```
1. Read $P'=(\mathbf{\Omega}',G')$.
2. Measure $X'$.
3. Set $\mathcal{L}_1=\emptyset$. For $j\in\{1,\ldots,m\}$ do:
   (a) Try to compute $R_j=\texttt{Dec}(X'\oplus\Omega'_j)$.
   (b) If the decoding succeeds then add $j$ to $\mathcal{L}_1$.
4. If $\mathcal{L}_1=\emptyset$ then abort.
5. Set $\mathcal{L}_2=\emptyset$. For $i\in\mathcal{L}_1$ do:
   (a) $\hat{X}_i=\Omega'_i\oplus\texttt{Enc}(R_i)$
   (b) Compute $G_i=f(\mathbf{\Omega}'||\hat{X}_i)$.
   (c) If $G_i=G'$ then add $i$ to the list $\mathcal{L}_2$.
6. If $|\mathcal{L}_2|\neq 1$ then abort; else $\hat{X}=X_{\mathcal{L}_2}$.
```

Enrollment steps 2 and 3 represent the construction of the ordinary COM helper data W. The D_j in step 4b are decoy measurements. They follow the same distribution as X and are therefore statistically indistinguishable from a real POK measurement. The Ω_j is the COM helper data associated with the decoy D_j. The purpose of the random permutation in step 6 is to hide from the adversary which entry of $\mathbf{\Omega}$ is the actual helper data. Here it is crucial that the adversary cannot 'see inside' the hash G. Since the entries of $\mathbf{\Omega}$ are distributed exactly in the way real helper data should be, his knowledge about the statistics of W does not help him to decide which entry is the real one. This intuitive statement is made more precise in Section 5.

Note that in step 7 the hash is computed over the *whole* vector $\mathbf{\Omega}$. This ensures that any manipulation of the public data will be detected, be it in the hash, in W or in the decoys. This way of protecting the helper data against manipulation was introduced by [2]. Alternatively, one may use a Message Authentication Code with Key Manipulation Security [4].

At reconstruction the public data may have been altered, which is why we use the notation P', $\mathbf{\Omega}'$, G' in step 1 of algorithm R0. A list $\mathcal{L}_1$ is made of $\mathbf{\Omega}'$ entries that lead to successful decoding. The whole set $\mathcal{L}_1$ has to taken into account, since some of the decoys may by chance decode, and the order of the entries is random. The list of candidates is further narrowed down to a list $\mathcal{L}_2$ of entries whose X_j generates the correct hash. If $P' = P$ and $X' \approx X$ then typically there is only one candidate left in $\mathcal{L}_2$. If $P' \neq P$ or X' is too far away from X to be error-corrected, then typically $\mathcal{L}_2 = \emptyset$. (Algorithm R0 continues the search after having found its first match. Alternatively, we could stop.)

The main idea behind the scheme is that the adversary cannot distinguish between the true helper data and the decoys, while the device's knowledge of X' helps it to see the difference.

It may happen that the choice of system parameters is such that R0 has a long running time. For instance, the processing time in step 3 is linear in m, where m may be a large number. Furthermore, the choice of n, k, and m may give rise to a long list $\mathcal{L}_1$, and the number of hashes that has to be computed in step 5 is linear in $|\mathcal{L}_1|$. In the schemes below we aim to reduce the running time of the reconstruction algorithm.

4.2 Scheme #1: Secure Sketch for biometrics database

Below we show a more efficient pair of algorithms, in the biometrics scenario. The main difference with the naive approach is the use of the syndromes $\mathbf{\Phi}$. Note that $\texttt{Syn}(X) = \texttt{Syn}(W)$. Hence revealing F conveys no extra information to the adversary; he could already compute F from W unaided.

The idea behind scheme #1 is that comparing $\texttt{Syn}(X')$ to $\texttt{Syn}(X)$ and the other syndromes allows the device to heuristically re-order $\mathbf{\Omega}'$ in such a way that the most likely candidates are tried out first. Here it is crucial that the parity check matrix of the code has low density: then a small Hamming distance between X and X' leads to a small Hamming distance between $\texttt{Syn}(X)$ and $\texttt{Syn}(X')$.

The running time of the reconstruction algorithm R1 is practically independent of the number of dummies, except for steps 4–6 which are lightweight. Most importantly, due to the sorting R1 does not have to compute many decodings and hashes.

Algorithm E1:
enrollment for biometrics database

```
1. Measure the biometric $X \in \{0,1\}^n$.
2. Draw $R \in \{0,1\}^k$ uniformly at random.
3. Compute the syndrome $F = \texttt{Syn}(X)$.
   Compute helper data $W = X \oplus \texttt{Enc}(R)$.
4. For $j \in \{1,\dots,m-1\}$ do:
   (a) Uniformly draw $\Sigma_j \in \{0,1\}^k$.
   (b) Draw $D_j \in \{0,1\}^n$ from the distribution $\rho$.
   (c) Compute $\Phi_j = \texttt{Syn}(D_j)$.
       Compute $\Omega_j = D_j \oplus \texttt{Enc}(\Sigma_j)$.
5. Choose a random permutation $\pi$.
6. Construct a vector $\mathbf{\Phi} = \pi(\Phi_1,\cdots,\Phi_{m-1},F)$.
   Construct a vector $\mathbf{\Omega} = \pi(\Omega_1,\cdots,\Omega_{m-1},W)$.
7. Compute $G = f(\mathbf{\Phi}||\mathbf{\Omega}||X)$.
8. Store public data $P = (\mathbf{\Phi},\mathbf{\Omega},G)$.
```

Algorithm R1: efficient biometric verification

```
1. Read $P' = (\mathbf{\Phi}',\mathbf{\Omega}',G')$.
2. Measure the fresh biometric $X'$.
3. Compute $F' = \texttt{Syn}(X')$.
4. For $j \in \{1,\dots,m\}$ do: $d_j = d_{\text{Hamm}}(F',\Phi'_j)$.
5. Make a permutation $\lambda$ that sorts $(d_j)_{j=1}^m$ in ascending order.
6. Let $\tilde{\mathbf{\Omega}} = \lambda(\mathbf{\Omega}')$.
7. Let $j = 0$.
8. Increase $j$. If $j = m+1$ then abort.
9. Try to compute $R_j = \texttt{Dec}(X' \oplus \tilde{\Omega}_j)$.
   If the decoding fails then goto 8.
10. $\hat{X}_j = \tilde{\Omega}_j \oplus \texttt{Enc}(R_j)$.
11. If $G' \neq f(\mathbf{\Phi}'||\mathbf{\Omega}'||\hat{X}_j)$ then goto 8.
12. Accept.
```

Remark 1: In step 7 of E1, the $\mathbf{\Phi}$ and $\mathbf{\Omega}$ also serve as salt for the hashing of X.

Remark 2: There are many alternative ways to organize the steps in (R1,E1). For instance, in step 6 of R1 the vector $\mathbf{\Omega}'$ does not have to be physically permuted; permutation of the indices $\{1,\cdots,m\}$ is more efficient.

4.3 Scheme #2: Fuzzy Extractor

Below we present the Fuzzy Extractor version of algorithms (E1, R1), in the POK scenario.

```
Algorithm E2: enrollment for POK
  1. Measure the POK output $X \in \{0,1\}^n$.
  2. Generate random $A$. Compute $K = g(X, A)$.
  3. Draw $S \in \{0,1\}^k$ uniformly at random.
  4. Compute the syndrome $F = \mathtt{Syn}(X)$.
     Compute helper data $W = X \oplus \mathtt{Enc}(S)$.
  5. For $j \in \{1, \ldots, m-1\}$ do:
     (a) Uniformly draw $\Sigma_j \in \{0,1\}^k$.
     (b) Draw $D_j \in \{0,1\}^n$ from the distribution $\rho$.
     (c) Compute $\Phi_j = \mathtt{Syn}(D_j)$.
         Compute $\Omega_j = D_j \oplus \mathtt{Enc}(\Sigma_j)$.
  6. Choose a random permutation $\pi$.
  7. Construct a vector $\mathbf{\Phi} = \pi(\Phi_1, \cdots, \Phi_{m-1}, F)$.
     Construct a vector $\mathbf{\Omega} = \pi(\Omega_1, \cdots, \Omega_{m-1}, W)$.
  8. Compute $G = f(\mathbf{\Phi}||\mathbf{\Omega}||A||X)$.
  9. Store public data $P = (\mathbf{\Phi}, \mathbf{\Omega}, A, G)$.
```

```
Algorithm R2: efficient reconstruction of POK
  1. Read $P' = (\mathbf{\Phi}', \mathbf{\Omega}', A', G')$.
  2. Measure the POK output $X'$.
  3. Compute $F' = \mathtt{Syn}(X')$.
  4. For $j \in \{1, \ldots, m\}$ do: $d_j = d_{\text{Hamm}}(F', \Phi'_j)$.
  5. Make a permutation $\lambda$ that sorts $(d_j)_{j=1}^m$ in ascending order.
  6. Let $\tilde{\mathbf{\Omega}} = \lambda(\mathbf{\Omega}')$.
  7. Let $j = 0$.
  8. Increase $j$. If $j = m+1$ then abort.
  9. Try to compute $R_j = \mathtt{Dec}(X' \oplus \tilde{\Omega}_j)$.
     If the decoding fails then goto 8.
  10. $\hat{X}_j = \tilde{\Omega}_j \oplus \mathtt{Enc}(R_j)$.
  11. If $G' \neq f(\mathbf{\Phi}'||\mathbf{\Omega}'||A'||\hat{X}_j)$ then goto 8.
  12. $\hat{K} = g(\hat{X}_j, A')$.
```

The only difference with the biometrics scenario (E1,R1) is the use of the auxiliary randomness A and the computation of K and $\hat{K}$.

5 Analysis of the SCOM

We investigate how much information about X is revealed to the adversary by showing him $\mathbf{\Omega}$. In principle we should be looking at the leakage from the whole public data P, but there one hits a snag: information-theoretically there is no such a thing as a one-way function. The hash G hides its input *in practice*, but information theory gives $I(X;P) = I(X;W)$. The leakage from $\mathbf{\Omega}$ is a better way to represent the adversary's actual workload.
In the biometrics scenario, the relevant quantity to look at is Shannon entropy. (One might argue that min-entropy is more important, but since we do not have the stringent requirements that cryptographic keys have to satisfy[2] , we will stick to Shannon entropy.) The relevant quantity in the POK scenario is the Rényi entropy H_2, which features in the ε-extractable randomness (11). We show results for both scenarios.
In Section 5.3 we also briefly look at memory requirements.

5.1 Leakage in terms of Shannon entropy

We first present two lemmas that allow us to relate the leakage $I(X;\mathbf{\Omega})$ to the adversary's ignorance about the permutation Π. Then we present a result for small m and for large m.

Lemma 3 *The adversary's ignorance about X given $\mathbf{\Omega}$ can be written as*

$$\mathsf{H}(X|\mathbf{\Omega}) = \mathsf{H}(X|W) + I(\Pi; X\mathbf{\Omega}). \tag{13}$$

Proof: We write $\mathsf{H}(X|\mathbf{\Omega}\Pi)$ in two ways: as $\mathsf{H}(X|W)$ and as $\mathsf{H}(X\Pi|\mathbf{\Omega}) - \mathsf{H}(\Pi|\mathbf{\Omega}) = \mathsf{H}(X\Pi|\mathbf{\Omega}) - \mathsf{H}(\Pi) = \mathsf{H}(X|\mathbf{\Omega}) + \mathsf{H}(\Pi|X\mathbf{\Omega}) - \mathsf{H}(\Pi)$. Equating the two different expressions yields (13). □

Lemma 4 *Let $t(x, \boldsymbol{\omega})$ denote the number of entries in $\boldsymbol{\omega}$ that are consistent with x, i.e. $t(x, \boldsymbol{\omega}) = |\{j : \mathtt{Syn}(\omega_j) = \mathtt{Syn}(x)\}|$. Then*

$$\mathsf{H}(X|\mathbf{\Omega}) = \mathsf{H}(X|W) + \log m - \mathbb{E}_{x\boldsymbol{\omega}} \log t(x, \boldsymbol{\omega}). \tag{14}$$

Proof: We write $I(\Pi; X\mathbf{\Omega}) = \mathsf{H}(\Pi) - \mathsf{H}(\Pi|X\mathbf{\Omega}) = \log m! - \mathsf{H}(\Pi|X\mathbf{\Omega})$. For a given $t(x, \boldsymbol{\omega})$ there are t possible ways to place w in $\boldsymbol{\omega}$; furthermore there are $m-1$ further entries to be permuted, which can be done in $(m-1)!$ ways. The total number of permutations π consistent with x and $\boldsymbol{\omega}$ is $t \cdot (m-1)!$. They are all equiprobable from the point of view of the adversary. Hence $\mathsf{H}(\Pi|X = x, \mathbf{\Omega} = \boldsymbol{\omega}) = \log[t \cdot (m-1)!]$. It follows that $\mathsf{H}(\Pi|X\mathbf{\Omega}) = \mathbb{E}_{x\boldsymbol{\omega}} \log[t \cdot (m-1)!] = \log(m-1)! + \mathbb{E}_{x\boldsymbol{\omega}} \log t$ and $I(\Pi; X\mathbf{\Omega}) = \log \frac{m!}{(m-1)!} - \mathbb{E}_{x\boldsymbol{\omega}} \log t$. Finally we substitute this expression for $I(\Pi; X\mathbf{\Omega})$ into Lemma 3. □

Theorem 1 *The conditional entropy $\mathsf{H}(X|\mathbf{\Omega})$ can be bounded from below as*

$$\mathsf{H}(X|\mathbf{\Omega}) \geq \mathsf{H}(X|W) + \log m - \frac{m-1}{\ln 2} \mathbb{E}_x q_{\mathtt{Syn}(x)}. \tag{15}$$

Proof: We write $t(x, \boldsymbol{\omega}) = 1 + u(x, \boldsymbol{\omega})$ and use $\ln(1+u) \leq u$. This gives $\mathbb{E}_{x\boldsymbol{\omega}} \log t(x, \boldsymbol{\omega}) \leq \frac{1}{\ln 2} \mathbb{E}_{x\boldsymbol{\omega}} u(x, \boldsymbol{\omega})$. For given x, the u is binomial-distributed with parameters $m-1$ and $q_{\mathtt{Syn}(x)}$. (See Section 2 for the notation q.) Thus we have $\mathbb{E}_{x\boldsymbol{\omega}} u(x, \boldsymbol{\omega}) = \mathbb{E}_x (m-1) q_{\mathtt{Syn}(x)}$. □
At first sight (15) might seem to contradict the well known principle 'conditioning reduces Shannon entropy'. However, it should be borne in mind that $\mathbf{\Omega}$ is not just W plus decoys; $\mathbf{\Omega}$ results from a *function* Π applied to W and the decoys. The function Π reduces the leaking effect of W.

[2]Remember that most biometrics cannot be kept secret, since it is possible to measure them surreptitiously.

The probability $q_{\mathtt{Syn}(x)}$ is typically of the order $1/2^{n-k}$ if X is not too strangely distributed. Hence the last term in (15) is a small correction term if $m < 2^{n-k}$. Eq. (15) confirms the intuitive idea that the attacker's effort increases by a factor $\approx m/2$. Note that the bound in Theorem 1 is far from tight when m is large. For large m we have the following result.

Theorem 2 *The conditional entropy* $\mathsf{H}(X|\mathbf{\Omega})$ *can be bounded from below as*

$$\mathsf{H}(X|\mathbf{\Omega}) \geq \mathsf{H}(X) - \frac{1}{m}\cdot\frac{2^{n-k}-1}{\ln 2}. \qquad (16)$$

Proof: As in the proof of Theorem 1, we write $t = 1 + u$. Furthermore we split u into its expectation value (at fixed x) and a deviation: $u = (m-1)q_{\mathtt{Syn}(x)} + \delta$, where $\mathbb{E}_\delta \delta = 0$. This gives

$$\begin{aligned}\mathbb{E}_{x\boldsymbol{\omega}} \log t &= \mathbb{E}_x \log[m q_{\mathtt{Syn}(x)}] + \mathbb{E}_x \log[1 + \frac{1-q_{\mathtt{Syn}(x)}}{m q_{\mathtt{Syn}(x)}}] \\ &\quad + \mathbb{E}_{x\delta} \log(1 + \frac{\delta}{1 + [m-1]q_{\mathtt{Syn}(x)}}). \qquad (17)\end{aligned}$$

Next we use $\ln(1+z) \leq z$ twice, and $\mathbb{E}_\delta \delta = 0$, to get

$$\begin{aligned}\mathbb{E}_{x\boldsymbol{\omega}} \log t &\leq \log m + \mathbb{E}_x \log q_{\mathtt{Syn}(x)} + \frac{1}{\ln 2}\mathbb{E}_x \frac{1-q_{\mathtt{Syn}(x)}}{m q_{\mathtt{Syn}(x)}} \\ &= \log m + \sum_{z\in\{0,1\}^{n-k}} q_z \log q_z + \frac{1}{m \ln 2}(-1 + \sum_z \frac{q_z}{q_z}) \\ &= \log m - \mathsf{H}(\mathtt{Syn}X) + \frac{2^{n-k}-1}{m \ln 2}. \qquad (18)\end{aligned}$$

Substitution of (18) into Lemma 4 finishes the proof. □

If m is of order 2^{n-k} or larger, then the $\frac{1}{m}$ term in Theorem 2 is a small correction term; we see that $\mathbf{\Omega}$ then hardly leaks anything about X, as we expected intuitively.

5.2 Leakage in terms of Rényi entropy

We present a bound on $\mathsf{H}_2(X|\mathbf{\Omega})$ that is useful for large m. We observe that for the adversary each entry in $\boldsymbol{\omega}$ is equally likely to be the correct one. Thus, his knowledge about x can be parametrized as a probability distribution that is conditioned on each of the entries ω_j with equal probability $1/m$. This gives

$$p_{x|\boldsymbol{\omega}} = \frac{1}{m}\sum_{j=1}^{m} p_{x|\omega_j}. \qquad (19)$$

Based on (19) we obtain the following result.

Theorem 3 *The conditional Rényi entropy* $\mathsf{H}_2(X|\mathbf{\Omega})$ *can be bounded from below as*

$$\mathsf{H}_2(X|\mathbf{\Omega}) \geq \mathsf{H}_2(X) - \frac{1}{m \ln 2}\left[\frac{\mathbb{E}_w \sum_x p^2_{x|w}}{\sum_x p_x^2} - 1\right]. \qquad (20)$$

Proof:

$$\begin{aligned}\mathsf{H}_2(X|\mathbf{\Omega}) &= -2 \log \mathbb{E}_{\boldsymbol{\omega}} \sqrt{\textstyle\sum_x p^2_{x|\boldsymbol{\omega}}} \qquad (21)\\ &\geq -2 \log \sqrt{\mathbb{E}_{\boldsymbol{\omega}} \textstyle\sum_x p^2_{x|\boldsymbol{\omega}}} \qquad (22)\\ &= -\log \mathbb{E}_{\boldsymbol{\omega}} \sum_x \frac{1}{m^2}\sum_{a,b=1}^{\ell} p_{x|\omega_a} p_{x|\omega_b} \qquad (23)\\ &= -\log \frac{1}{m^2}\sum_x\left[\sum_a \mathbb{E}_{\boldsymbol{\omega}} p^2_{x|\omega_a} + \sum_{a,b:a\neq b} \mathbb{E}_{\boldsymbol{\omega}} p_{x|\omega_a} p_{x|\omega_b}\right] \qquad (24)\\ &= -\log \sum_x\left[\frac{1}{m}\mathbb{E}_w p^2_{x|w} + [1-\frac{1}{m}]p_x^2\right] \qquad (25)\\ &= -\log\left[(\sum_x p_x^2)(1 + \frac{\mathbb{E}_w \sum_x p^2_{x|w} - \sum_x p_x^2}{m \sum_x p_x^2})\right] \qquad (26)\\ &= \mathsf{H}_2(X) - \log(1 + \frac{\mathbb{E}_w \sum_x p^2_{x|w} - \sum_x p_x^2}{m \sum_x p_x^2}). \qquad (27)\end{aligned}$$

In (22) we used Jensen's inequality. In (23) we substituted (19). Finally (20) is obtained from (27) by using $\log(1+z) = \ln(1+z)/\ln 2 \leq z/\ln 2$. □

Remark: If X is not too far from uniform, then the $\frac{1}{m}$-term in Theorem 3 is of order $2^{n-k}/m$, i.e. the same order of magnitude as the $\frac{1}{m}$-term in Theorem 2.

When spammed helper data $\mathbf{\Omega}$ is used instead of W, the entropy $\mathsf{H}_2(X|W)$ in the extractable randomness formula (11) can be replaced by the (much) larger number $\mathsf{H}_2(X|\mathbf{\Omega})$.

5.3 Storage requirements

In the biometrics scenario there is a large amount of storage space per enrolled person, since the public data P is usually stored in a dedicated database. Blowing up the database by a factor m could be feasible. Furthermore, the original W is a very small thing to start with.

In the POK scenario, however, the public data is usually stored on the device that contains the POK. This device has to be cheap; hence nonvolatile memory may become an issue.

Below we tabulate some numerical estimates. The $k = 128$ case corresponds to a POK with a 128-bit key. The $k = 64$ case represents the biometrics scenario. (We could even have chosen k a little smaller.) The 'err' is the number of errors that the LDPC code can correct. Under 'Mem' we list the space required to store $\mathbf{\Omega}$ and $\mathbf{\Phi}$, namely $m \cdot (2n-k)$ bits. The values of n are approximate and are meant only to give orders of magnitude. For m we list two values: $m = 2^{(n-k)/2}$, which reduces the leakage from W by approximately half the bits, and $m = 2^{n-k}$ which almost completely cancels the leakage.

	$k = 64$			$k = 128$		
err	n	$\log m$	Mem	n	$\log m$	Mem
1	72	4	0.2KB	138	5	0.6KB
		8	2.5KB		10	19KB
2	78	7	1.4KB	146	9	10KB
		14	0.2MB		18	5.1MB
3	85	10.5	19KB	154	13	0.2MB
		21	27MB		26	1.4GB

Table 1: *Memory required to store* $\mathbf{\Omega}$ *and* $\mathbf{\Phi}$*, listed as a function of* k*, the number of errors to be corrected and* $\log m$*.*

6 Discussion

We have proposed the Spammed Code Offset Method, in which the adversary gets spammed with bogus helper data. For small spam factor m, the security is increased by roughly $\log m$ bits. For large m, of the order 2^{n-k} or larger, the leakage $I(X;W)$ is practically eliminated. These statements are quantified in Theorems 1, 2 and 3. While the workload of the adversary is increased, the workload of the legitimate party (counting only calls to `Dec` and the hash function f) stays almost constant as a function of m. This is achieved by using Hamming distance in syndrome space as a fast candidate selection criterion, where the use of an LDPC code makes sure that a small distance between X and X' translates to a small distance in syndrome space.

In the POK scenario it depends on various system parameters whether it makes sense to use the SCOM. If the available nonvolatile memory in the device is limited and there is ample entropy in X, then the ordinary COM suffices. Table 1 illustrates that for a 128-bit key and $\log m$ comparable to $n-k$, the size of the public data rapidly becomes infeasibly large as the number of errors increases. However, it should be borne in mind that a even a small m already yields a (modest) security improvement. The SCOM provides a new kind of trade-off: a more effective use of source entropy is achieved at the price of digital memory usage.

In the biometrics case it is especially important to eliminate the leakage $I(X;W)$, since the entropy of X is usually rather low and has to be maximally exploited. Fortunately it is easier to meet the memory requirements in this scenario.

As future work we will do experiments with various LDPC codes in order to optimize the search in algorithms R1 and R2, and in order to get more precise numbers in Table 1. Another interesting issue to look at is the cross-linkability between biometric templates in different databases. When lots of decoy entries are added to the templates, it becomes much harder for an adversary to decide if templates in different databases belong to the same person, since the decoys are likely to cause false matches.

Acknowledgements
We thank Ruud Pellikaan, Ludo Tolhuizen, Joep de Groot and Jean-Paul Linnartz for useful discussions.

References

[1] B. Barak, Y. Dodis, H. Krawczyk, O. Pereira, K. Pietrzak, F.-X. Standaert, and Y. Yu. Leftover Hash Lemma, revisited. In *CRYPTO*, volume 6841 of *LNCS*, pages 1–20. Springer, 2011.

[2] X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith. Secure remote authentication using biometric data. In *Eurocrypt 2005*, volume 3494 of *LNCS*, pages 147–163. Springer-Verlag, 2005.

[3] J.L. Carter and M.N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979.

[4] R. Cramer, Y. Dodis, S. Fehr, C. Padró, and D. Wichs. Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 471–488, 2008.

[5] J.A. de Groot, B. Škorić, N. de Vreede, and J.-P. Linnartz. Information leakage of continuous-source Zero Secrecy Leakage Helper Data Schemes. `http://eprint.iacr.org/2012/566`, 2012.

[6] Y. Dodis, M. Reyzin, and A. Smith. Fuzzy Extractors: How to generate strong keys from biometrics and other noisy data. In *Eurocrypt 2004*, volume 3027 of *LNCS*, pages 523–540. Springer-Verlag, 2004.

[7] S. Fehr and S. Berens. The conditional Rényi entropy, 2013.

[8] B. Gassend. Physical Random Functions. Master's thesis, Massachusetts Institute of Technology, 2003.

[9] J. Håstad, R. Impagliazzo, L.A. Levin, and M. Luby. Construction of pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[10] A. Juels and M. Wattenberg. A fuzzy commitment scheme. In *ACM Conference on Computer and Communications Security (CCS) 1999*, pages 28–36, 1999.

[11] J.-P. Kaps, K. Yüksel, and B. Sunar. Energy scalable universal hashing. *IEEE Trans. Computers*, 54(12):1484–1495, 2005.

[12] R. Renner and S. Wolf. Smooth Rényi entropy and applications. In *IEEE International Symposium on Information Theory (ISIT) 2004*, page 233, 2004.

[13] R. Renner and S. Wolf. Simple and tight bounds for information reconciliation and privacy amplification. In *Asiacrypt 2005*, volume 3788 of *LNCS*, pages 199–216. Springer-Verlag, 2005.

[14] A.-R. Sadeghi and D. Naccache, editors. *Towards hardware-intrinsic security.* Springer, 2010.

[15] D.R. Stinson. Universal hashing and authentication codes. *Designs, Codes, and Cryptography*, 4:369–380, 1994.

[16] P. Tuyls, G.-J. Schrijen, B. Škorić, J. van Geloven, R. Verhaegh, and R. Wolters. Read-proof hardware from protective coatings. In *Cryptographic Hardware and Embedded Systems (CHES) 2006*, volume 4249 of *LNCS*, pages 369–383. Springer-Verlag, 2006.

[17] P. Tuyls, B. Škorić, and T. Kevenaar. *Security with Noisy Data: Private Biometrics, Secure Key Storage and Anti-Counterfeiting.* Springer, London, 2007.

[18] E.A. Verbitskiy, P. Tuyls, C. Obi, B. Schoenmakers, and B. Škorić. Key extraction from general nondiscrete signals. *IEEE Transactions on Information Forensics and Security*, 5(2):269–279, 2010.

[19] B. Škorić, C. Obi, E. Verbitskiy, and B. Schoenmakers. Sharp lower bounds on the extractable randomness from non-uniform sources. *Information and Computation*, 209:1184–1196, 2011.