How to Extract and Expand Randomness: A Summary and Explanation of Existing Results*

Yvonne Cliff, Colin Boyd, and Juan Gonzalez Nieto

Information Security Institute, Queensland University of Technology
GPO Box 2434, Brisbane Qld 4001, Australia.
y.cliff@isi.qut.edu.au, {c.boyd, j.gonzaleznieto}@qut.edu.au

Abstract. We examine the use of randomness extraction and expansion in key agreement (KA) protocols to generate uniformly random keys in the standard model. Although existing works provide the basic theorems necessary, they lack details or examples of appropriate cryptographic primitives and/or parameter sizes. This has lead to the large amount of min-entropy needed in the (non-uniform) shared secret being overlooked in proposals and efficiency comparisons of KA protocols. We therefore summarize existing work in the area and examine the security levels achieved with the use of various extractors and expanders for particular parameter sizes. The tables presented herein show that the shared secret needs a min-entropy of at least 292 bits (and even more with more realistic assumptions) to achieve an overall security level of 80 bits using the extractors and expanders we consider. The tables may be used to find the min-entropy required for various security levels and assumptions. We also find that when using the short exponent theorems of Gennaro et al., the short exponents may need to be much longer than they suggested.
Key words: randomness extraction, randomness expansion, key agreement, key exchange protocols, pseudorandom function (PRF), universal hash function, leftover hash lemma (LHL).

1 Introduction

In this paper we examine the techniques available for extracting and expanding randomness in the context of key agreement (KA) protocols. In such protocols, an agreed secret key is often a random member of a given group, and not a string of bits distributed uniformly at random. However, when the key is used, e.g. as the key of a symmetric encryption scheme, it is likely that a key consisting of bits distributed uniformly at random will be necessary, requiring the use of randomness extraction, and possibly randomness expansion techniques.

Informally, a randomness extractor is a family of functions keyed by a random but public value, where the input to each function is a value with high entropy, and the output is indistinguishable from a uniformly random bit string. Unfortunately, the number of bits of entropy in the input must usually be much larger than the number of bits in the output for practical security parameters.

A randomness expander, or pseudo-random function family (PRFF), is a family of functions keyed by secret, uniformly random strings, with each function taking as input any publicly known value and outputting a value indistinguishable from one distributed uniformly at random.

When only one relatively short uniformly random key is required of a KA protocol, the output of a randomness extractor may be used as the required key. However, it is more likely that the output of the extractor will be used to key a randomness expander, to provide a longer key or multiple keys, e.g. when one random group member is used to derive a MAC (message authentication code) key for use in the KA protocol, as well as the final agreed secret key.

This step of converting a randomly chosen group member to a uniformly random string or strings of bits is often not discussed in papers proposing KA protocols. However, if the key derivation function is not modelled with the random oracle model, this step has a significant impact on how large the security

* This is the full version of the paper that appears in ACNS 2009. Research funded by the Australian Research Council through Discovery Project DP0773348

parameter of the KA protocol needs to be to achieve proven security of a given level. As noted by Gennaro et al. [1, p.4] and Chevassut et al. [2, p.2], this point is often overlooked, particularly in protocol efficiency comparisons.

One reason randomness extraction and expansion and their effect on security parameter sizes is often overlooked may be the plethora of existing works that must be examined to obtain the necessary background knowledge, and the dearth numerical examples. Therefore, this paper provides:

- a summary of existing results on randomness extraction and expansion, including relevant definitions and theorems, and numerical examples,
- details of the short exponent discrete-log (DLSE) assumption and its use with randomness extraction and expansion (including numerical examples),
- an analysis of why assumptions made by Dodis et al. [3] in some justifications of the use of HMAC and cascade chaining (such as SHA) as randomness extractors are not realistic,
- a valuable resource for protocol designers and implementors to enable them to use security parameters of an appropriate size in efficiency comparisons and implementations, without having to examine all of the existing works,
- the observation, through the use of numerical examples for values of practical interest, that some of the theoretical results available are of limited practical value, due to the non-existence of underlying functions of an appropriate size or the availability of better methods,
- results for the standard model only; although use of a random oracle as a randomness extractor would mean that shorter parameters would be required in a protocol to achieve the same security level, making it more efficient, our aim is to describe solutions available for the standard model.

We will begin by examining the suitability of various candidates as randomness expanders, which will tell us how large a key needs to be provided by the randomness extractor. We will then examine randomness extractors, and the amount of entropy required for their input in order to extract a long enough key for the randomness expander.

Prior work includes that of Dodis et al. [3], the first to attempt to justify the use of CBC-MAC, cascade chaining and HMAC as randomness extractors in the standard model, and that of Gennaro et al. [1] who examined the use of universal hash functions as randomness extractors in conjunction with the DDH (decisional Diffie-Hellman) assumption and short exponents. Chevassut et al. [2] made some brief but interesting observations on randomness extraction and expansion in general, before providing methods of randomness extraction which are more efficient than those studied here, but are only applicable for groups of points over an elliptic curve (EC), and the group of prime order q in $\mathbb{Z}_p^*$ where $p = 2q + 1$ and is prime. Their method for EC groups requires computations in the KA protocol to be carried out on an EC as well as its twist, instead of just on the curve, and so increases the number of computations required. However, the method may be advantageous as these computations on the EC and its twist will be in smaller groups than those necessary when using the methods studied in this paper in conjunction with computations on the EC only. Fouque et al. [4] showed that the lower order bits of a member of a subgroup of $\mathbb{Z}_p^*$ may be considered random in the right circumstances. Another work of Fouque et al. [5] examined the use of HMAC as a randomness extractor when the randomness is extracted from the HMAC key, and included an analysis of the cascade construction as a randomness extractor.

2 Notation and Basic Definitions

The notation mostly follows Dodis et al. [3] and Gennaro et al. [1]. For a probability distribution $\mathcal{X}$ over a set A, the notation $x \in_{\mathcal{X}} A$ indicates that x is chosen from A according to the distribution $\mathcal{X}$. The notation $x \in_{\mathrm{R}} A$ indicates that x is chosen from A according to the uniform distribution. $\Pr_{\mathcal{X}}[x]$ indicates the probability that distribution $\mathcal{X}$ assigns to the value $x \in A$. In some cases, definitions taken from other works have been modified to make the notation consistent.

This paper uses a concrete security approach, to allow determination of the size of the parameters needed in a protocol to achieve a given level of security. Following Gennaro et al. [1], we speak of circuits of size S

having a certain probability, ϵ of solving a particular problem. One may also think of a circuit of size S as a programme running in time t, where 'time' actually includes the length of the description of the programme (to avoid trivializing hard problems through the use of large precomputed tables), as well as the actual execution time of that programme [6].

We now introduce computational indistinguishability, a refinement of the notion of statistical distance (or variation distance) from probability theory. If two distributions are statistically close, they are computationally indistinguishable, although the converse is not true [7, Sect. 3.2.2].

Definition 1 ((S,ϵ)-indistinguishability [1, p.19]). *Let $\mathcal{X}, \mathcal{Y}$ be two probability distributions over A. Given a circuit D (called the distinguisher) consider the following quantities:*

$$\delta_{D,\mathcal{X}} = \Pr_{x\in\mathcal{X}}[D(x)=1] \quad \textit{and} \quad \delta_{D,\mathcal{Y}} = \Pr_{y\in\mathcal{Y}}[D(y)=1] \tag{1}$$

We say that the probability distributions $\mathcal{X}$ and $\mathcal{Y}$ are (S,ϵ)-indistinguishable if for every circuit D of size $\leq S$ we have that $|\delta_{D,\mathcal{X}} - \delta_{D,\mathcal{Y}}| \leq \epsilon$.

Definition 2 (Statistical Distance [8, p.131]). *The statistical distance between two probability distributions $\mathcal{X}$ and $\mathcal{Y}$ over a set A is defined to be*[1] $\Delta[\mathcal{X};\mathcal{Y}] = \frac{1}{2}\sum_{x\in A} |\Pr_{\mathcal{X}}[x] - \Pr_{\mathcal{Y}}[x]|$.

Lemma 1 ([3, p.500]). *If two distributions have statistical distance of (at most) ϵ, they are ϵ-close. Distributions that are ϵ-close cannot be distinguished with probability better than ϵ even by a computationally unbounded adversary.*

The following lemma has a proof [1] based on the triangle inequality or "hybrid argument."

Lemma 2 ([1, p.19]). *Let three probability distributions $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ over a set A be such that (i) $\mathcal{X}$ is (S_1,ϵ_1) indistinguishable from $\mathcal{Y}$ and (ii) $\mathcal{Y}$ is (S_2,ϵ_2) indistinguishable from $\mathcal{Z}$. Then $\mathcal{X}$ is (S,ϵ) indistinguishable from $\mathcal{Z}$ where $S = \min(S_1,S_2)$ and $\epsilon = \epsilon_1 + \epsilon_2$.*

We now focus on describing how much randomness is in a probability distribution, defining min-entropy and its computational analogue.

Definition 3 (Min-entropy [1, p.9]). *If $\mathcal{X}$ is a probability distribution over A, the min-entropy of $\mathcal{X}$ is* $\text{min-ent}(\mathcal{X}) = \min_{x\in A:\Pr_{\mathcal{X}}[x]\neq 0}(-log_2(\Pr_{\mathcal{X}}[x]))$ *. (Note that if $\mathcal{X}$ has min-entropy t then for all $x \in A$, $\Pr_{\mathcal{X}}[x] \leq 2^{-t}$.)*

Definition 4 (Computational entropy t [1, p.10]). *A probability distribution $\mathcal{Y}$ has (S,ϵ) computational entropy t if there exists a probability distribution $\mathcal{X}$ that is (S,ϵ) indistinguishable from $\mathcal{Y}$ and min-ent$(\mathcal{X}) \geq t$.*

Definition 5 (Function Family [6, adapted from full paper p.7]). *A function family $f : K \times D \to R$ (also denoted $\{f_\kappa\}_{\kappa\in K}$), where K is a non-empty set of keys, is a collection of functions, $f_\kappa(\cdot) \stackrel{\text{def}}{=} f(\kappa,\cdot)$ for $\kappa \in K$, from a domain, D, to a range, R. We call f a permutation family if $D = R$, and for each key $\kappa \in K$, f_κ is a permutation on D.*

Definition 6 (Truly Random Function (TRF) [5, 6]). *Denote the set of all functions from M to $\{0,1\}^L$ with* $\text{Rand}^{M\to 2^L}$ *(there are $2^{L|M|}$ such functions). A function chosen at random from* $\text{Rand}^{M\to 2^L}$ *is a truly random function (TRF) with input domain M and output domain $\{0,1\}^L$.*

A TRF may be implemented by an oracle that, for each new oracle query, generates an output selected at random from $\{0,1\}^L$, and for oracle queries that are not new, replies with the same output as previously given for that input.

[1] Gennaro et al.'s definition [1] is twice this value, but seems erroneous when compared with others [8, 3, 7] .

Definition 7 (Cascade Construction [5]). *The cascade construction (also known as keyed Merkle-Damgard cascade chaining) is the construction used for iterated hash functions. Let* $H : \{0,1\}^c \times \{0,1\}^* \to \{0,1\}^c$ *denote an iterated hash function, and let* $h : \{0,1\}^c \times \{0,1\}^b \to \{0,1\}^c$ *(the so-called compression function) be a family with key space* $\{0,1\}^c$*. The cascade construction of* h *is the function* $h^* : \{0,1\}^c \times \left(\{0,1\}^b\right)^* \to \{0,1\}^c$ *defined by:*

$$y_0 = a, y_i = h(y_{i-1}, x_i) \text{ and } h^*(a,x) = y_n$$

where $x = (x_1, \ldots, x_n)$ *is a* $n \cdot b$ *bit string and* $a \in \{0,1\}^c$*. To construct* H*, messages must be padded to an exact multiple of* b *bits. The padding, denoted* $\mathrm{pad}(|x|)$*, is a function of the input length,* $|x|$*. Let* $x_{pad} = x \parallel \mathrm{pad}(|x|)$*. Then* H *is defined by* $H(a,x) = h^*(a, x_{pad})$*.*

Let $1 \leq c' \leq c$ *be an integer and let* $\mathrm{msb}_{c'}(\cdot)$ *denote the* c' *most significant bits of a bit string. For any function* H *with range* $\{0,1\}^c$*, we define for every input* x *the truncated iterated hash function* $\tilde{H}(x) = \mathrm{msb}_{c'}(H(x))$*; e.g. SHA-384 has* $c' = 384$ *and* $c = 512$*.*

Definition 8 (NMAC [5]). $\mathrm{Nmac} : \{0,1\}^c \times \{0,1\}^c \times \{0,1\}^* \to \{0,1\}^{c'}$ *is a hash function family constructed from a (possibly truncated) iterated hash function* $\mathrm{Hash} : \{0,1\}^c \times \{0,1\}^* \to \{0,1\}^{c'}$*. If* $(k_1, k_2) \in (\{0,1\}^c)^2$ *is a couple of keys and* $x \in \{0,1\}^*$ *is the input, the definition of NMAC is* $\mathrm{Nmac}^{\mathrm{Hash}}(k_1, k_2, x) = \mathrm{Hash}(k_2, \mathrm{Hash}(k_1, x))$*.*

Definition 9 (HMAC [5]). *HMAC is a hash function from* $\{0,1\}^* \times \{0,1\}^*$ *to* $\{0,1\}^{c'}$*. Let ipad and opad be two* b*-bit strings and* IV *be a* c*-bit string. Let* $Hash : \{0,1\}^c \times \{0,1\}^* \to \{0,1\}^{c'}$ *be the (possibly truncated) iterated hash function with compression function* $h : \{0,1\}^c \times \{0,1\}^b \to \{0,1\}^c$*. If the key* k *is a bit string from* $\{0,1\}^b$*, then*

$$\begin{aligned}\mathrm{Hmac}_{IV}^{\mathrm{Hash}}(ipad, opad; k, x) &= \mathrm{Hash}\left(IV, [k \oplus opad] \parallel \mathrm{Hash}\left(IV, [k \oplus ipad] \parallel x\right)\right) \\ &= \mathrm{Nmac}^{\mathrm{Hash}}(h(IV, k \oplus ipad), h(IV, k \oplus opad), x).\end{aligned}$$

If the key k *is smaller than* b *bits, then it is first padded with '0' bits to form a* b*-bit string, and this string is used as the key. If the key* k *is larger than* b *bits, it is first hashed using* Hash *to obtain a* c'*-bit digest, then padded with* $b - c'$ *'0' bits to obtain a* b*-bit string, which is then used as the key.*

3 Randomness Expansion

To ascertain the minimum output length required from the randomness extractor used, we begin by examining the randomness expander—also known as a pseudorandom function (PRF) family, or PRFF—to be used, since the output of the randomness extractor will be used as the key to the PRFF.

Definition 10 (Pseudorandom Function Family [9, 6]). *A function family* $f = \{f_\kappa\}_{\kappa \in K}$ *is a* (S, q, ϵ) *pseudorandom function family (PRFF) if a circuit,* $\mathcal{A}$*, of size* S *which is given oracle access to either* f_κ *for* $\kappa \in_R K$ *or a TRF with the same domain and range as the functions in* f*, and makes at most* q *queries to this oracle, has advantage at most* ϵ *in distinguishing whether it has access to a random member of* f *or a TRF; i.e.:*

$$\epsilon \geq \mathbf{Adv}_f^{\mathrm{prf}}(q, S) \stackrel{\mathrm{def}}{=} \max_{\mathcal{A}} \left\{ \mathbf{Adv}_f^{\mathrm{prf}}(\mathcal{A}) \right\} \tag{2}$$

$$\mathbf{Adv}_f^{\mathrm{prf}}(\mathcal{A}) \stackrel{\mathrm{def}}{=} \left| \Pr[\mathcal{A}^{\mathcal{O}(\cdot)} = 1 | \mathcal{O}(\cdot) \in_{\mathrm{R}} f] - \Pr[\mathcal{A}^{\mathcal{O}(\cdot)} = 1 | \mathcal{O}(\cdot) \in_{\mathrm{R}} \mathrm{Rand}] \right| \tag{3}$$

The values $\mathbf{Adv}_f^{\mathrm{prp}}(q, S)$ and $\mathbf{Adv}_f^{\mathrm{prp}}(\mathcal{A})$, may be defined similarly for an adversary $\mathcal{A}$ against a pseudorandom permutation family, except that $\mathcal{A}$ attempts to tell the difference between the permutation family and a truly random permutation, rather than a TRF.

When one PRFF is used with various different keys (e.g. each party from a number of parties may use its own key to produce pseudorandom values from the PRFF), there is a linear decrease in security. Furthermore, the key to a PRFF may be only computationally indistinguishable from random, in which

case the level of security of the PRFF and the level computational indistinguishability must be combined. Theorems 10 and 11 in Appendix B formally state and prove this.

Function families widely believed to be pseudorandom include CBC-MAC used in conjunction with a block cipher, HMAC or the HMAC variant NMAC, and cryptographic hash functions such as SHA-1 or SHA-256 based on the cascade construction, but with the fixed IV (initialization vector) replaced with a random key. Appendix C discusses the merits of each of these options in turn. Here we overview the security levels provided by each option. Some assumptions (described in the appendix) must be made on the security level of the underlying block ciphers or compression functions to arrive at the below concrete security levels.

3.1 CBC-MAC

Bellare et al. [6] have proved that CBC-MAC is a secure PRFF if the underlying block cipher is a secure pseudorandom permutation family and the input length is constant. The level of security provided depends on the block length, number of queries, q, and number of blocks of input, l. When ql is small (e.g. 2), the security level is about $k = b - 3$ bits. Otherwise, if we have $ql \leq 2^k$ (which we are assuming when we consider a security level of 2^k sufficient), then we will require $k \leq (b-2)/2$. If the block cipher to be used with CBC-MAC is AES-128, AES-192, or AES-256, then the block length, b, will be 128 bits for each of these ciphers [10]. Therefore, the level of security provided by CBC-MAC when used in conjunction with any of these ciphers will be no greater than 125 bits, and will be less for values of q and l larger than 1. Hence, CBC-MAC is likely to be an acceptable choice of randomness expander for security levels of 80 bits if the number of queries to randomness expander with a single key is small and the length of each query is also small, but inadequate for security levels of 128 bits and higher. If an unlimited number of queries or queries with a very large length are able to be made by the adversary to the randomness expander with a single key, the security level will only be $(b-2)/2 = 63$ bits when $b = 128$.

3.2 HMAC

Bellare [11] has proven that HMAC is a secure pseudorandom function if the compression function of the underlying hash function is a pseudorandom function. The analysis assumes that the key provided to HMAC is the same length as a block for the underlying hash function (i.e. b bits). To achieve a shorter key of only $2c$ bits (where c is the length of the output of the compression function), NMAC may be used, which is similar to HMAC but differs in its use of keying material. However, NMAC is generally used for analysis of HMAC only, so availability of an existing implementation is unlikely. Any implementation of NMAC will require access to the compression function underlying the hash function to be used, which may be difficult to acquire.

Hash functions likely to be used with HMAC include MD5 [12], RIPEMD-160 [13], SHA-1, SHA-256, SHA-384 and SHA-512 [14]. Table 1 shows the block size (b), compression function key and output length (c), hash function output length (c') and HMAC security level for each of these algorithms, where q is the number of queries using the same key and l is the number of blocks per query. The traditional security level is $c/2$ bits, due to the birthday based forgery attacks against iterated MACs [15] that require $2^{c/2}$ oracle queries.

3.3 Cascade Construction

Bellare, Canetti and Krawczyk [9] have provided a proof of pseudorandom function family security for cryptographic hash functions such as SHA-1 or SHA-256 based on the cascade construction, but with the fixed IV (initialization vector) replaced with a random key, provided the input is prefix-free and the underlying compression function used by the hash function is a pseudorandom function family. (It is possible to remove the prefix-free requirement by using extra keying material, but it is unlikely to be necessary in our setting. Belare et al. provided another construction to improve security using randomization, but if the extra randomness is counted as part of the key, more keying material than HMAC is required for a similar security level.)

Algorithm	b	c'	c	Security level $(q, l \leq 2)$		Security level (q is large)		
				for Hash $(c-2)$	for HMAC $(c-4)$	max. for Hash $(\frac{c-20}{2})$	conservative Hash $(\frac{c-40}{3})$	HMAC $(\frac{c-2}{2})$
MD5	512	128	128	126	124	54	29	63
RIPEMD-160	512	160	160	158	156	70	40	79
SHA-1	512	160	160	158	156	70	40	79
SHA-224	512	256	224	254	252	118	72	127
SHA-256	512	256	256	254	252	118	72	127
SHA-384	1024	512	384	510	508	246	157	255
SHA-512	1024	512	512	510	508	246	157	255

Table 1. Block and key size, output length, and hash and HMAC security level

Security level (bits)	Key Length				
	CBC-MAC	Casc. min.	Casc. consrv.	NMAC	HMAC
29			128		
40			160		
54		128			
63	128			256	512
70		160			
72			256		

Security level (bits)	Key Length			
	Casc. min.	Casc. consrv.	NMAC	HMAC
79			320	512
118	256			
127			512	512
157		512		
246	512			
255			1024	1024

Table 2. Summary of required key lengths for a given security level when q large

Table 1 shows the security level of the cascade construction using the same notation as for HMAC. Assumptions made to obtain the security levels are described in the appendix. The difference between the maximum and conservative security levels for large q is due to different assumptions concerning the efficiency of the best attack against the underlying compression function.

3.4 Key Length Summary

In summary, when $q \leq 2$, a minimum of 128 bits will be needed to key the randomness expander, e.g. using CBC-MAC or the cascade construction, achieving a security level around 125 bits. In this case, the cascade construction allows the use of a key about two bits longer than the required security level, and requires fewer key bits than using NMAC or HMAC for the same security level.

When there is no restriction on q, the cascade construction provides the lowest key length for a given security level when we take the security level as being $\frac{c-20}{2}$. However, if the more conservative security level of $\frac{c-40}{3}$ bits is used, then NMAC may be better, depending on the level of security required. Table 2 summarizes the results.

4 Randomness Extraction

Let us consider a KA protocol that allows the participating parties to agree upon a secret value, called the pre-secret, that an adversary cannot distinguish from a value drawn uniformly at random from a particular distribution, e.g. from a group in which the DDH (Decisional Diffie-Hellman) assumption holds. Furthermore, assume a randomness extractor and expander are used to derive a final key from the pre-secret, such that the final key is indistinguishable from a uniformly random bit string. As will be seen in this section, when using the techniques of randomness extraction and expansion considered in this paper, the entropy of the pre-secret must be much larger than the security level required of the final key. Therefore, if the pre-secret is from a suitable group, it may seem desirable to use the discrete-log short-exponent (DLSE) assumption to enable calculations required by the KA protocol to be more efficient, by using exponents shorter than the

group order. In addition, if the KA protocol is Diffie-Hellman (DH) based, it may be desirable to use the t-DDH assumption (a relaxation of the DDH assumption) to allow the use of groups with non-prime order with the protocol. These assumptions and theorems are therefore provided in Appendix A. Note that two theorems of Gennaro et al. [1] regarding use of the DLSE assumption are incorrect in their original paper and have been corrected in the appendix according to details supplied by Gennaro in a personal communication.

The most common existing randomness extractor definition is of a strong randomness extractor:

Definition 11 (Strong randomness extractor [16]). *A family of efficiently computable hash functions $H = \{h_\kappa : \{0,1\}^n \rightarrow \{0,1\}^c | \kappa \in \{0,1\}^d\}$ is called a (t, ϵ) strong randomness extractor, if for any random variable X over $\{0,1\}^n$ that has min-entropy at least t, if κ is chosen uniformly at random from $\{0,1\}^d$ and R is chosen uniformly at random from $\{0,1\}^c$, the following two distributions are within statistical distance ϵ from each other: $(\kappa, h_\kappa(X)) \cong_\epsilon (\kappa, R)$.*

By Lemma 1, the above distributions are also computationally indistinguishable from each other. Notice that the definition means that the key to the randomness extractor, κ, may be made public, yet the output of the randomness extractor, given a secret input with sufficient min-entropy, is indistinguishable from a string of bits distributed uniformly at random.

Since it is likely that X will only have computational entropy (not min-entropy) of a certain level, we introduce the following definition (which is similar to a recent definition of Fouque et al. [5] in an independent work).

Definition 12 (Strong computational randomness extractor). *A family of efficiently computable functions $H = \{h_\kappa : A \rightarrow \{0,1\}^c | \kappa \in \{0,1\}^d\}$ is a $(t, S, \epsilon, S', \epsilon')$ strong computational randomness extractor if given any probability distribution $\mathcal{X}$ over A such that $\mathcal{X}$ has (S, ϵ) computational entropy at least t, the following two probability distributions are (S', ϵ')-indistinguishable:*

$$\mathcal{H} = \{(\kappa, h_\kappa(x)) \text{ for } \kappa \in_{\mathrm{R}} \{0,1\}^d \text{ and } x \in_{\mathcal{X}} A\} \tag{4}$$

$$\mathcal{R}^{\mathrm{h}} = \{(\kappa, r) \text{ for } \kappa \in_{\mathrm{R}} \{0,1\}^d \text{ and } r \in_{\mathrm{R}} \{0,1\}^c\} \tag{5}$$

It is possible to show that a strong randomness extractor is also a strong computational randomness extractor (see Theorem 12 in Appendix B). However, the converse is not necessarily true.

The above definitions assume that the key to the randomness extractor, κ, is generated afresh for each use of the randomness extractor. This may be appropriate in some protocols, where parties may have exchanged nonces with each other and can use these values to generate the key. However, it is imperative that any such nonces be authenticated (i.e. unable to be influenced by the adversary) and not subject to replay by the adversary. Otherwise, a key derived from these nonces may not be distributed uniformly at random over $\{0,1\}^d$ as required for these extractors.

When parties are unable to generate a new key, κ, each time they use a randomness extractor, the key κ may be fixed as part of the system parameters. However, this requires multiple uses of the randomness extractor with the one key. It turns out that the security of the randomness extractor decreases linearly with the number of queries to it using the same key. Protocols using this approach may be proven secure in one of two ways. As part of the proof of security of the protocol, one often focuses on the security of one particular session chosen at random from all sessions. In the proof, it may be possible to use the above definitions to prove the security of the protocol. The total number of sessions will appear as a factor in the security reduction (due to focusing on one session chosen at random from all sessions), and this will cater for the reduction in security due to multiple uses of the extractor with only one key. The other way to justify the use of a single key to the randomness extractor is via Theorems 13, 17 and 18 in Appendix B.

4.1 Combining Extraction and Expansion

To ascertain the security of the overall key derivation function consisting of randomness extraction and expansion, all of the relevant theorems must be combined (e.g. extractor reuse, Diffie-Hellman assumption,

For $\mathcal{EEDH}$ and $\mathcal{EER}$ indistinguishable:

- $\frac{S_5}{\epsilon_5} \geq 2^{k+1}$
- $e \geq k + 2 + \log_2(q_1)$
- t-DDH assumptions: $\left(2^{k+4}q_1 + q_1 + q_2, \frac{1}{2}\right)$ and $\left(q_1 + q_2 + 1, \frac{1}{2^{k+3}q_1}\right)$
- s-DLSE assumptions: $\left(2^{i-1}s\ln(2s)(Y + 2Z), \frac{1}{2}\right)$ and $\left(Y^i s\ln(s)(Z+1), \frac{1}{Y}\right)$ where $Y \stackrel{\text{def}}{=} (\log_2(m) - s)\,2^{k+5}q_1$, $Z \stackrel{\text{def}}{=} S_3 + q_1 + q_2$.

For $\mathcal{EEDH}^*$ and $\mathcal{EER}^*$ indistinguishable:

- $\frac{S_5 - S_8}{\epsilon_5} \geq 2^{k+1}q_1$ where $S_8 \approx (q_1 - 1)q_2$
- $e \geq k + 2 + 2\log_2(q_1)$
- t-DDH assumptions: $\left(2^{k+4}q_1^2 + q_1q_2, \frac{1}{2}\right)$ and $\left(q_1q_2 + 1, \frac{1}{2^{k+3}q_1^2}\right)$
- s-DLSE assumptions: $\left(2^{i-1}s\ln(2s)(Y + 2Z), \frac{1}{2}\right)$ and $\left(Y^i s\ln(s)(Z+1), \frac{1}{Y}\right)$ where $Y \stackrel{\text{def}}{=} (\log_2(m) - s)\,2^{k+5}q_1^2$, $Z \stackrel{\text{def}}{=} S_3 + q_1q_2$.

In both cases $i = 3$ unless $\log_2(m) > 2s - \log_2(\epsilon_1)$, in which case $i = 2$ and the smallest sensible value for ϵ_1 is $\frac{1}{Y}$. S_3 is the cost of a multi-exponentiation in G.

Table 3. Requirements for the two cases to be indistinguishable from random

short exponent theorems, expander reuse etc.). As a useful example, we combine Theorems 10, 11, 9 and 17 in Theorem 19 in Appendix B. A summary of the results is as follows.

Let $H = \{h_\kappa : \{0,1\}^n \to \{0,1\}^c | \kappa \in \{0,1\}^d\}$ be a $(t, 2^{-e})$ strong randomness extractor, with a maximum of q_1 queries per (publicly known) randomness extractor key κ, and let $f = \{f_\lambda\}_{\lambda \in K}$ be a (S_5, q_2, ϵ_5) PRFF, with a maximum of q_2 queries per (secret) key λ. Suppose a security level of k bits is desired for the final key(s) output by f. Let G be a cyclic group of order m generated by g, such that m is odd, or $m/2$ is odd. We assume there are q_1 publicly known pairs g^{a_i}, g^{b_i} for $1 \leq i \leq q_1$, and that the $g^{a_ib_i}$ are the inputs to $h_\kappa(\cdot)$.

We consider two cases. For the first, we require $q_1 - 1$ outputs of H to be indistinguishable from random, use the other output of H to key f, and require the q_2 outputs of f using this key to be indistinguishable from random. The indistinguishable distributions are labelled $\mathcal{EEDH}$ and $\mathcal{EER}$.

In the second case, all q_1 outputs of H are used to key f, giving a total of q_1q_2 outputs of f, and all of these outputs must be indistinguishable from random. The indistinguishable distributions are labelled $\mathcal{EEDH}^*$ and $\mathcal{EER}^*$. Which of these cases is appropriate will depend upon the protocol in question and its proof of security. Table 3 shows the requirements in each case, where the distributions are to be indistinguishable with a security level of k bits.

As an example putting it all together, suppose that a security level of $k = 80$ bits is required, we desire that the $\mathcal{EEDH}$ and $\mathcal{EER}$ distributions are indistinguishable, $q_1 = 1$ and $q_2 = 1$. Furthermore, suppose that m is prime. Then we need:

- a randomness expander with an 81 bit security level, e.g. CBC-MAC with a 128 bit key for its block cipher or MD5 with a 128 bit key;
- a $(t, 2^{-82})$ strong randomness extractor for some t that outputs enough bits to key the randomness expander, e.g. a universal hash function—in that case $t = 292$ (see Sect. 4.2);
- $\left(2^{84}, \frac{1}{2}\right)$ and $\left(3, \frac{1}{2^{83}}\right)$ t-DDH assumptions on G, e.g. G could be of order 292 bits on an elliptic curve (292 is the maximum of $t = 292$ and $2 \cdot 85$);
- exponents of the full 292 bits since the short exponent assumption needs the short exponent to be longer than 292 bits (probably around 600 bits).

Further details of the calculations are provided in Table 6 in Appendix B. This example contradicts the statement by Gennaro et al. [1, Sect. 6] that exponents of length $2k$ may be used to achieve a security level of k bits, since in our example, we need the exponent to be of length between $5k$ and $7.4k$. It seems that Gennaro et al. have not substituted actual values into their theorem stating that short exponents may be used, and have thus come to an incorrect conclusion about how long the short exponents really need to be.

4.2 Available Extractors

We now compare the available randomness extractors, focusing on output lengths of 128, 160, 256 and 512 bits, as these are the possible key lengths for the randomness expanders in Sect. 3.4. The reader may make his own comparisons for other output lengths with the information provided.

We first discuss the use of the Leftover Hash Lemma (LHL) to show that a universal (or almost universal) hash function may be used as a randomness extractor. Following this, we discuss the use of a PRFF as a randomness extractor, as analysed by Chevassut et al. [2], and then summarize the results of Fouque et al. [4] on deterministic extraction of lower order bits from subgroups of $\mathbb{Z}_p^*$. Then another work of Fouque et al. [5] is summarized with several results on using HMAC to extract randomness from the HMAC key, and a result on using the cascade construction as a randomness extractor. Appendix D provides an overview and detailed comments on the problems with the first work [3] to consider the suitability of CBC-MAC, the cascade construction, and HMAC for use as randomness extractors in the standard model.

We aim for the output of the extractor to be (S', ϵ') indistinguishable from uniform with $\frac{S'}{\epsilon'} \geq 2^{81}$ as a minimum requirement (this will achieve a security level no greater than $k = 80$ bits when the randomness extractor and expander are used together). Theorem 19 (summarized in Table 3) will provide the basis for our numerical analysis of the advantages of each extractor. We will use the notation of Sect. 4.1 and assume (as was done there) that $S_4 \approx q_1$, $S_6 \approx q_2$ and $S_8 \approx (q_1 - 1)q_2$. Furthermore, we let c be the key length of the expander, and hence the output length of the extractor; t be the min-entropy, b be the block size and L be the number of blocks of the pre-secret (ps, e.g. the DH value) which is input to the extractor. We will examine the parameters required of each extractor to achieve various security levels in the following cases (notation is as in Sect. 4.1). In our examples, we use the cascade construction as the expander, since it is the best (see Sect. 3.4). The parameters required to achieve other security levels or in other cases can be derived by the reader.

1. Each extractor key is used only once ($q_1 = 1$; this would be the case if the key is chosen afresh in each protocol run); the expander is used only once or twice with each key ($q_2 \leq 2$); it is desired that $\mathcal{EEDH}$ and $\mathcal{EER}$ are indistinguishable (the KA protocol's security will be lower than k bits, since the total number of sessions will appear as a factor in its security reduction).
2. The extractor key is a global parameter used up to 2^{30} times ($q_1 \leq 2^{30}$); other requirements are as for the previous case; e.g. many other applications use the extractor at a k-bit security level; the KA protocol proof focuses on one session; that session's two keys (output by the expander) have k bits of security (again, the protocol's overall security will be lower than k bits).
3. Each extractor key is used once ($q_1 = 1$); the expander is used many times with each key ($q_2 > 2$); other requirements are the same as for the first case.
4. The extractor key is the same in all KA protocol sessions (but not used in other applications), and there are up to 2^{30} sessions ($q_1 \leq 2^{30}$); the expander is used many times with each key ($q_2 > 2$); $\mathcal{EEDH}^*$ and $\mathcal{EER}^*$ must be indistinguishable (so the number of sessions will not be an extra factor in the protocol proof). We assume $S_8^2 \approx q_1^2 q_2^2 \leq 2^{k+1} q_1$ so that a cascade construction security level of $k + 1 + \log_2(q_1)$ bits (less conservative option) gives $\frac{S_5 - S_8}{\epsilon_5} \geq 2^{k+1} q_1$.[2]

Almost Universal Hash Functions The Leftover Hash Lemma (LHL) is well-known and allows the use of a universal (or almost universal) hash function as an extractor which is probabilistic and optimal in general [2]. There are several variations of the LHL in the literature; the one provided is mainly from Chevassut et al. [2], and similar to Dodis et al. [3, p.501].

[2] We want $(S_5 - S_8)/\epsilon_5 \geq 2^{k+1} q_1$. When using the cascade construction (less conservative option) we have $1/\epsilon_5 \geq 2^c / \left(2^{20} S_5^2\right)$ (see the comments following Theorem 22), so we need $\left((S_5 - S_8)\, 2^c\right) / \left(2^{20} S_5^2\right) \geq 2^{k+1} q_1$ where c is the key length of the randomness extractor. When $S_5 = S_8 + 1$, we have $\left((S_5 - S_8)\, 2^c\right) / \left(2^{20} S_5^2\right) \geq 2^{k+1} q_1$ implies $2^c \geq 2^{k+21} q_1 S_8^2$. However, for a security level of s bits for the randomness expander, we require $2^c \geq 2^{2s+20}$, and if $s = k + 1 + \log_2(q_1)$, this will imply the first requirement when $s \geq 2 \log_2(S_8)$. For values of S_5 much larger than S_8, $S_5 - S_8 \approx S_5$ and so a security level of $k + 1 + \log_2(q_1)$ bits will be sufficient.

Case	t	k	e	c	Case	t	k	e	c
1	$c+2e$	k	$k+2$	$\geq (k+2)+2$	3	$c+2e$	k	$k+2$	$\geq 2(k+2)+20$
1	292	80	82	128	3	420	80	82	256
1	380	124	126	128	3	492	116	118	256
1	476	156	158	160	3	900	192	194	512
1	764	252	254	256	3	1004	244	246	512
1	1532	508	510	512					
2	$c+2e$	k	$k+32$	$\geq (k+32)+2$	4	$c+2e$	k	$k+62$	$\geq 2(k+32)+20$
2	352	80	112	128	4	540	80	142	256
2	476	126	158	160	4	552	86	148	256
2	704	192	224	256	4	956	160	222	512
2	764	222	254	256	4	1020	192	254	512
2	1532	478	510	512	4	1064	214	276	512

Table 4. Universal hash function parameter examples

Definition 13 (δ-AU (almost universal)). *Let c and b be integers, and $\{h_\kappa\}_{\kappa\in\mathcal{K}}$ be a family of hash functions with domain $\{0,1\}^b$, range $\{0,1\}^c$ and key space $\mathcal{K}$. We say that the family $\{h_\kappa\}_{\kappa\in\mathcal{K}}$ is δ-almost universal (δ-AU)[3] if for every pair of different inputs x, y from $\{0,1\}^b$ it holds that $\Pr(h_\kappa(x) = h_\kappa(y)) \leq \delta$, where the probability is taken over $\kappa \in_R \mathcal{K}$. For a given probability distribution $\mathcal{X}$ on $\{0,1\}^b$, we say that $\{h_\kappa\}_{\kappa\in\mathcal{K}}$ is δ-AU w.r.t. $\mathcal{X}$ if $\Pr(h_\kappa(x) = h_\kappa(y)) \leq \delta$ where the probability is taken over $\kappa \in_R \mathcal{K}$ and $x, y \in_R \mathcal{X}$ conditioned on $x \neq y$.*

An example of a universal hash function is the function that multiplies a Toeplitz matrix (one with constant diagonals) by the input to create the output [17]. Appendix E gives more details and examples of universal hash functions.

Lemma 3 (LHL with δ-AU [2]). *Let $\mathcal{X}$ be a probabilistic distribution over $\{0,1\}^b$ with min-entropy at least t. Let e be an integer and $c \leq \alpha - 2e$ where $\alpha = \min(t, \log_2(1/\xi))$. Let $\mathcal{H} = \{h_\kappa\}_{\kappa\in\mathcal{K}}$, with h_κ having domain $\{0,1\}^b$ and range $\{0,1\}^c$ for any $\kappa \in \mathcal{K}$, be a δ-AU hash function family with $\delta = \frac{1}{2^c} + \xi$. Let H be a random variable uniformly distributed on $\mathcal{H}$, X denote a random variable taking values in $\{0,1\}^b$, and H and X be independent. Then, $(H, H(X))$ is 2^{-e}-uniform on $\mathcal{H} \times \{0,1\}^c$.*

This lemma states that a δ-almost universal hash function is a $(t, 2^{-e})$ strong randomness extractor. It was used to generate Table 4, where we must have $\xi \leq 2^{-t}$. It shows that even the most basic requirements mean a computational entropy of 292 bits in the input to the randomness extractor. More realistic requirements may mean a much higher level of computational entropy is required. Because of their significant key size requirements, and because other functions such as cryptographic hash functions are more readily available, universal hash functions are often not used for key derivation.

PRFFs as Randomness Extractors Chevassut et al. [2] have shown that a PRFF may be used for randomness extraction with a publicly known key.

Theorem 1 ([2]). *If a family of functions, $\mathcal{F}$, is a $(S, 2, \xi)$-PRFF with domain $\{0,1\}^b$ and range $\{0,1\}^c$, S is the size of a circuit that makes 2 oracle queries on an instance of $\mathcal{F}$, then it is a $(\frac{1}{2^c} + \xi)$-AU hash function family.*

By using Lemma 3, we can conclude that a PRFF can be a strong randomness extractor, although the output of the PRF will generally need to be truncated to a length compatible with Lemma 3. Reuse of the

[3] Being δ-AU in Dodis et al. [3] is the same as being ξ-AUH in Chevassut et al. [2] for $\delta = \frac{1}{2^c} + \xi$ where c is the number of bits of output of the function. We use the notation of Dodis et al. in this paper. When $\delta = \frac{1}{2^c}$, the function is universal.

Case	b	t	k	e	c	Case	b	t	k	e	c
1			k	$k+2$	$\geq k+4$	3			k	$k+2$	$\geq 2(k+2)+20$
1	1024	733	80	82	128	3	1024	861	80	82	256
1	2048	1178	80	82	128	3	1024	897	116	118	256
1	1024	777	124	126	128	3	2048	1742	192	194	512
1	1024	841	156	158	160	3	2048	1794	244	246	512
1	2048	1546	252	254	256						
1	2048	2058	508	510	512						
2			k	$k+32$	$\geq k+34$	4			k	$k+62$	$\geq 2(k+32)+20$
2	1024	763	80	112	128	4	1024	921	80	142	256
2	1024	841	126	158	160	4	1024	927	86	148	256
2	1024	1003	192	224	256	4	2048	1770	160	222	512
2	2048	1546	222	254	256	4	2048	1802	192	254	512
2	2112	2091	478	510	512	4	2048	1824	214	276	512

Table 5. Parameter examples for least significant bits extraction

extractor can then be covered by Theorem 13 or 18. For example, to achieve a security level of $k = 80$ bits in Case 1, as shown in Table 4, we will need $\xi < 2^{-292}$. This rules out the use of CBC-MAC, since the block size is only likely to be 128 bits, and so the security level will only be about 125 bits. The use of HMAC or the cascade construction seems appropriate, provided we do not need ξ smaller than 2^{-508} or 2^{-510} respectively. In our example, we could use SHA-384 or better, and would need to truncate the output to 128 bits.

Deterministic Extraction of Lower Order Bits The analysis of Fouque et al. [4] allows one to use the lower or higher-order bits from subgroups of $\mathbb{Z}_p^*$.

Theorem 2. *Let p be a b-bit prime, that is $2^{b-1} < p < 2^b$, G a subgroup of $\mathbb{Z}_p^*$ of order q with $q \gg \sqrt{p}$, l the integer such that $2^{l-1} \leq q \leq 2^l$ and X a random variable uniformly distributed in G. Let $\mathrm{lsb}_c(X)$ denote the c least significant bits of X. Let e be a positive integer and let $l > t = b/2 + c + e + \log_2(b) + 1$. Then the function $\mathrm{lsb}_c(\cdot)$ is a $(t, 2^{-e})$-deterministic extractor for the G-group distribution. If $p^{1/2} \leq q \leq p^{2/3}$ then the requirement on l may be refined to $l > t = b/4 + 3l/8 + c + e + \log_2(b) + 3$, and if $256 \leq q \leq p^{1/2}$, it may be refined to $l > t = b/8 + 5l/8 + c + e + \log_2(b) + 3$. Let $\mathrm{msb}_c(X)$ denote the c most significant bits of X and let $\delta = (2^n - p)/2^n$. If $3\delta < 2^{-e-1}$ and $l > t = n/2 + k + e + \log_2(n) + 1$, then $\mathrm{msb}_c(\cdot)$ is a $(t, 2^{-e})$-deterministic extractor.*

Table 5 shows some parameter examples using Theorem 2 with the four cases under consideration. Comparing it with Table 4, we can see that more computational entropy is generally required than when using a universal hash function. Fouque et al. recommended the use of the DLSE assumption to shorten the exponents required and thus improve efficiency. However, Sect. 4.1 indicates that much more than $2e$ bits will be required, contrary the indication of Fouque et al. (summarizing Gennaro et al.'s work [1]). However, one advantage of this method is that it is deterministic, and so does not require a key for the extractor.

HMAC Fouque et al. [5] have analysed the security of HMAC as a randomness extractor when the data from which the randomness is to be extracted (pre-secret, *ps*) is used as the key of HMAC. Because the pre-secret is used as the HMAC key, some other data (denoted *label*, of at most l blocks), which is possibly adversarily generated, is used as the input to HMAC. There are two separate results, depending on whether the pre-secret is longer than one block or not.

Theorem 3 ([5]). *Using the notation of this section and Definition 9, let $L = 1$, let ipad and opad be chosen uniformly at random and let IV be a fixed string. Let h' be the hash function defined by $h'_{IV}(pad, \cdot) = h(IV, \cdot \oplus pad)$ where the key is pad. Let S_h be the circuit size for one computation of h. Let h' be a $(S' + 2S_h, q = 2, \epsilon_1)$ PRFF, and h be both a $(S', q = 1, \epsilon_2)$ and $(O(l \cdot S_h), q = 2, \epsilon_3)$ PRFF. Then $\mathrm{Hmac}_{IV}^{\mathrm{Hash}}(ipad, opad; ps, label)$ is a $(t, \infty, 0, S', \epsilon')$ computational randomness extractor with $\epsilon' \leq \frac{\sqrt{2^{2c}(2^{-t} + 2\epsilon_1)}}{2} + \frac{1}{2^{c'}} + \epsilon_2 + 2l\epsilon_3$.*

This is only useful if $b \gg 2c$, since $L = 1$ implies $t \leq b$ and when $t = 2c$ the term under the square root is at least one. In the case of SHA-1, we have $b = 512$ and $c = c' = 160$. To achieve a security level of e bits for the output of HMAC, we want $S'/\epsilon' \geq 2^e$. If we assume $\epsilon_1 \leq (S' + 2S_h)/(S_h 2^b)$, $\epsilon_2 \leq S'/(S_h 2^c)$, $\epsilon_3 \leq lS_h/(S_h 2^c)$, and $l \ll 2^c$, and consider the case where $S' = S_h = 1$, we require $e \leq \min\left(\frac{t-2c+1}{2}, \frac{b-2c+1.6}{2}, c', c - 2\log_2(l) - 1\right)$. These conditions will also ensure that the conditions placed on e when $S' = 2^{e-1}$, $S_h = 1$ and we want $\epsilon' \leq \frac{1}{2}$, are met. Hence, when $t = 512$, we achieve the maximum security level of $e = 96$ bits; for $e = 82$ bits, we need $t = 483$ bits min-entropy.

To overcome the problem of the above theorem only being useful when $b \gg 2c$, the assumptions on the compression function can be modified. That is, it is assumed that h is a PRFF resistant to related key attacks (RKA) when it is keyed with a bit string of min-entropy at least t (denoted t-RKA; $t = c$ for classical RKA). This assumption cannot be reduced to the h PRFF-security against RKA, since it is possible to have a good PRFF for a uniformly distributed key that is not a good PRFF for a high-entropy key. We omit the details of a RKA adversary used in the following theorems, but note that if the exhaustive search adversary with circuit size S' is the best known t-RKA adversary, its advantage is smaller than $(S'/S_h)/2^t$. Fouque et al. state their revised theorem in terms of HPRF, which is constructed from several concatenations and iterations of HMAC (they do not describe HPRF in detail but refer the reader to TLS v1.2 [18]).

Theorem 4 ([5]). *Let $L = 1$, let ipad and opad be two fixed strings and let IV be chosen uniformly at random. Let h be a function family resistant to a t-RKA adversary with circuit size S' that makes at most 2 queries with advantage ϵ_0. Let S_h be the circuit size for one computation of h. Let HPRF be a concatenation of v HMAC, and* Hash *be truncated. Let h be both a $(S', q = 2v, \epsilon_1)$ and $(O(l \cdot S_h), q = 2, \epsilon_2)$ PRFF. Then* $\mathrm{Hprf}^{\mathrm{Hash}}_{ipad,opad}(IV; ps, label)$ *is a $(t, \infty, 0, S', \epsilon')$ computational randomness extractor with $\epsilon' \leq \epsilon_0 + \epsilon_1 + 4v^2 l\epsilon_2 + \frac{2v^2}{2^{c'}} + \frac{v^2}{2^c}$.*

Assuming $l = v = 1$, $\epsilon_0 \leq \frac{S'/S_h}{2^t}$, $\epsilon_1 \leq \frac{(S'/S_h)}{2^c}$, and $\epsilon_2 \leq \frac{(lS_h/S_h)}{2^c}$, we have $S'/\epsilon' \geq 2^e$ when $e \leq t-3$, $e \leq c-5$ and $e \leq c' - 4$. Hence, we can extract almost all of the pre-secret's entropy when it has less entropy than the number of bits output by HPRF, and the pre-secret is only one block long.

When the pre-secret is longer than one block, it is first hashed and padded with '0' bits to obtain a b-bit string. The following theorem covers this case for HMAC. We omit the similar theorem for HPRF (when it is constructed from several concatenations and iterations of HMAC) due to lack of space.

Theorem 5 ([5]). *Let $L \geq 2$, ipad and opad be fixed strings, and IV be a variable chosen uniformly at random. Define $\hat{h} : \{0,1\}^{c'} \times \{0,1\}^c \to \{0,1\}^c$ as $\hat{h}(x, y) = h(y, x \parallel 0^{b-c'})$. Let S_h be the circuit size for one computation of h. Let* Hash *be truncated. Let ϵ_2 be the RKA advantage of an adversary against $\hat{h}$ making at most 2 related key queries with circuit size S'. Let h be a $(S', q = 2, \epsilon_1)$, $(S', q = 1, \epsilon_3)$ and $(O(l \cdot S_h), q = 2, \epsilon_4)$ PRFF. Then* $\mathrm{Hmac}^{\mathrm{Hash}}_{IV}(ipad, opad; ps, label)$ *is a $(t, \infty, 0, S', \epsilon')$ computational randomness extractor with $\epsilon' \leq \frac{1}{2^{c'}} + \epsilon_2 + \epsilon_3 + 2l\epsilon_4 + \sqrt{2^{c'}\left(3 \cdot 2^{-t} + 2L\epsilon_1\right)}$.*

Assuming $\epsilon_1 \leq \frac{S'/S_h}{2^c}$, $\epsilon_2 \leq \frac{(S'/S_h)}{2^{c'}}$, $\epsilon_3 \leq \frac{(S'/S_h)}{2^c}$, and $\epsilon_4 \leq \frac{O(lS_h)/S_h}{2^c}$, we have $S'/\epsilon' \geq 2^e$ when $e \leq \frac{t-3.6-c'}{2}$, $e \leq \frac{c-c'-\log_2(L)-3}{2}$, $e \leq c' - 2$ and $e \leq c - 2\log_2(l) - 3$. Hence, when $L = 2$, $l = 1$, and SHA-384 is used, only $e = 62$ bits of security can be achieved for the output of HMAC, and this requires $t \geq 512$. To achieve a value of e close to a value of c', we need $c' = e + 2 = \frac{c}{3}$. To achieve this we could further truncate the output of SHA-384 to only $c' = 170$ bits, and use this new hash function in the HMAC implementation. Then, provided $t \geq 510$, we would have $e = 168$. For $e > 168$, a new compression function h with output larger than 512 bits is needed. Alternatively, to achieve our minimum requirement for Case 1 described above, of $e = 82$ and $c' = 128$, we could use SHA-384 but further truncate the output to only $c' = 128$ bits. In that case we would only need $t \geq 296$ bits. This is similar to using a universal hash function, which is not surprising, since the analysis of Fouque et al. made use of the LHL.

Cascade Construction Fouque et al. [5] also analysed the use of the cascade construction as a randomness extractor when the output is truncated to contain only c' bits, instead of c bits. Assume the compression

function h of hash function H (with key IV) is an $(S, q = 2, \epsilon)$ PRFF. Then H is a $(t, \infty, 0, S', \epsilon')$ computational randomness extractor for prefix free distributions of at most L blocks with $S = O(S')$ and $\epsilon' \leq \sqrt{2^{c'} \cdot (3 \cdot 2^{-t} + 2L\epsilon)}$. As before, assume $\epsilon \leq S/2^c$. Hence, $\epsilon' \leq \sqrt{2^{c'} \cdot (3 \cdot 2^{-t} + 2^{1-c}L \cdot O(S'))}$. To achieve a security level of e bits for the output of H, we want $S'/\epsilon' \geq 2^e$. When $O(S') = 1$, this equates to requiring $\min\left(\frac{t-c'-3.6}{2}, \frac{c-c'-3-\log_2(L)}{2}\right) \geq e$. When the requirements for $O(S') = 1$ are met, those for when $O(S') = 2^{e-1}$ will be met also. These restrictions on e are almost the same as for HMAC when the pre-secret is more than one block long, and so similar comments to those made for HMAC apply here.

5 Conclusion

This paper examined the use of randomness extraction and expansion in key agreement protocols to generate uniformly distributed keys. Although other works exist that provide the basic theorems necessary, they lack details or examples of what cryptographic primitives are appropriate and/or how large the parameters of those primitives must be. We have therefore summarized existing work in the area and examined the security levels achieved with the use of various extractors and expanders for particular sizes of parameters.

As noted in some existing works ([1, p.4], [2, p.2]), the large amount of min-entropy needed in the pre-secret is often overlooked in efficiency comparisons of KA protocols. In fact, using the tables presented in this paper, one may conclude that this shared secret will need a min-entropy of at least 292 bits to achieve an overall security level of 80 bits. More realistic assumptions on the number of times the randomness extractor and expander are used may require a much higher min-entropy for this security level. The tables may be used to find the min-entropy required for various security levels and assumptions on how the extractor and expander will be used. We also found that when numbers are substituted into the short exponent theorems of Gennaro et al., the exponents may need to be much longer than they suggested.

References

1. Gennaro, R., Krawczyk, H., Rabin, T.: Secure hashed Diffie-Hellman over non-DDH groups. In: Advances in Cryptology — EUROCRYPT 2004 Proceedings. Volume 3027 of Lecture Notes in Computer Science., Springer (2004) 361–381 Full paper: `http://eprint.iacr.org/2004/099`.
2. Chevassut, O., Fouque, P.A., Gaudry, P., Pointcheval, D.: The Twist-AUgmented technique for key exchange. In: Proceedings of the 2006 International Workshop on Practice and Theory in Public Key Cryptography (PKC 2006). Volume 3958 of Lecture Notes in Computer Science., Springer (2006) 410–426 Full paper: `http://eprint.iacr.org/2005/061`.
3. Dodis, Y., Gennaro, R., Håstad, J., Krawczyk, H., Rabin, T.: Randomness extraction and key derivation using the CBC, cascade and HMAC modes. In: Advances in Cryptology — CRYPTO 2004 Proceedings. Volume 3152 of Lecture Notes in Computer Science., Springer (2004) 494–510
4. Fouque, P.A., Pointcheval, D., Stern, J., Zimmer, S.: Hardness of distinguishing the MSB or LSB of secret keys in Diffie-Hellman schemes. In: ICALP '06: 33rd International Colloquium on Automata, Languages and Programming, Part II. Volume 4052 of Lecture Notes in Computer Science., Springer (2006) 240–251
5. Fouque, P.A., Pointcheval, D., Zimmer, S.: HMAC is a randomness extractor and applications to TLS. In: ASIACCS '08: Proceedings of the 2008 ACM symposium on Information, computer and communications security, ACM (2008) 21–32
6. Bellare, M., Kilian, J., Rogaway, P.: The security of the cipher block chaining message authentication code. Journal of Computer and System Sciences **61**(3) (2000) 362–399 Full paper: `http://www-cse.ucsd.edu/~mihir/papers/cbc.html`.
7. Goldreich, O.: The Foundations of Cryptography. Volume 1. Cambridge University Press (2001) Drafts at `http://wisdom.weizmann.ac.il/~oded/frag.html`.
8. Shoup, V.: A Computational Introduction to Number Theory and Algebra. Cambridge University Press (2005) Available at `http://shoup.net/ntb/`.
9. Bellare, M., Canetti, R., Krawczyk, H.: Pseudorandom functions revisited: The cascade construction and its concrete security. In: Proceedings of the 37$^{\text{th}}$ Annual Symposium on the Foundations of Computer Science, IEEE (1996) 514–523 Full paper at `http://www-cse.ucsd.edu/users/mihir/papers/cascade.html`.

10. NIST (National Institute for Standards and Technology): Advanced encryption standard (AES). FIPS PUB 197 (2001)
11. Bellare, M.: New proofs for NMAC and HMAC: Security without collision-resistance. In: Advances in Cryptology—CRYPTO 2006. Volume 4117 of Lecture Notes in Computer Science., Springer (2006) 602–619
12. Rivest, R.: The MD5 message-digest algorithm. Internet RFC 1321, Internet Engineering Task Force (1992)
13. Dobbertin, H., Bosselaers, A., Preneel, B.: RIPEMD-160: A strengthened version of RIPEMD. In: Fast Software Encryption—FSE '96. Volume 1039 of Lecture Notes in Computer Science., Springer (2096) 71–82
14. NIST (National Institute for Standards and Technology): Secure hash standard. FIPS PUB 180-2 (2000)
15. Preneel, B., van Oorschot, P.: On the security of iterated message authentication codes. IEEE Transactions on Information Theory **45**(1) (1999) 188–199
16. Dodis, Y.: Exposure-Resilient Cryptography. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (2000) `http://theory.lcs.mit.edu/~yevgen/academic.html`.
17. Mansour, Y., Nisan, N., Tiwari, P.: The computational complexity of universal hashing. In: Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing—STOC '90, ACM Press (1990) 235–243
18. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) protocol version 1.2. Internet RFC 5246, Internet Engineering Task Force (2007)
19. van Oorschot, P.C., Wiener, M.J.: On Diffie-Hellman key agreement with short exponents. In: Advances in Cryptology — EUROCRYPT 1996 Proceedings. Volume 1070 of Lecture Notes in Computer Science., Springer (1996) 190–199
20. Patel, S., Sundaram, G.S.: An efficient discrete log pseudo random generator. In: Advances in Cryptology — CRYPTO 1998 Proceedings. Volume 1462 of Lecture Notes in Computer Science., Springer (1998) 304–317
21. Brincat, K., Mitchell, C.: New CBC-MAC forgery attacks. In: 6th Australasian Conference on Information Security and Privacy—ACISP 2001. Volume 2119 of Lecture Notes in Computer Science., Springer (2001) 3–14
22. Gauravaram, P., Millan, W., Nieto, J.G., Dawson, E.: 3c—a provably secure pseudorandom function and message authentication code. a new mode of operation for cryptographic hash function. Cryptology ePrint Archive, Report 2005/390 (2005) `http://eprint.iacr.org/2005/390`.
23. Blum, L., Blum, M., Shub, M.: A simple unpredictable pseudorandom number generator. SIAM Journal on Computing **15** (1986) 364–383
24. Vazirani, U., Vazirani, V.: Efficient and secure pseudo-random number generation. In: Proceedings of the IEEE 25th Annual Symposium on Foundations of Computer Science. (1984) 458–463
25. Sidorenko, A., Schoenmakers, B.: Concrete security of the blum-blum-shub pseudorandom generator. In: Cryptography and Coding: 10th IMA International Conference Proceedings. Volume 3796 of Lecture Notes in Computer Science., Springer (2005) 355–375
26. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press (1997)
27. Nevelsteen, W., Preneel, B.: Software performance of universal hash functions. In: Advances in Cryptology—EUROCRYPT 1999. Volume 1592 of Lecture Notes in Computer Science., Springer (1999) 24–41

A Diffie-Hellman and Short Exponent Assumptions

A.1 Diffie-Hellman Assumptions and Theorems

Assumption 1 (Decisional Diffie-Hellman (DDH) [1, p.5]) *Let G be a cyclic group of order m generated by an element g. Consider the set $G^3 = G \times G \times G$ and the following two probability distributions over it:*

$$\mathcal{R}_G = \{(g^a, g^b, g^c) \text{ for } a, b, c \in_R \mathbb{Z}_m\} \tag{6}$$

and

$$\mathcal{DH}_G = \{(g^a, g^b, g^{ab}) \text{ for } a, b \in_R \mathbb{Z}_m\} \tag{7}$$

We say the (S, ϵ) Decisional Diffie-Hellman (DDH) Assumption holds over $G = \langle g \rangle$ (alternatively, that G is a (S, ϵ) DDH group) if the two distributions $\mathcal{R}_G$ and $\mathcal{DH}_G$ are (S, ϵ)-indistinguishable.

The t-DDH assumption makes clear how much computational entropy a DH value is assumed to hold. It is equivalent to the DDH assumption for a subgroup of prime order t. However, for a group G of non-prime order, the t-DDH assumption allows one to make the DDH assumption on the large prime order subgroup(s), and deduce the t-DDH assumption on the group G if certain conditions are met. The assumption and theorems follow, but further details may be found in Gennaro et al. [1]. Making the t-DDH assumption is likely to make the use of a randomness extractor necessary also.

Assumption 2 (t-Decisional Diffie Hellman (t-DDH) [1, p.11]) *We say that the (S, ϵ) t-DDH Assumption holds over a group G if there exists a family of probability distributions $\mathcal{X}(g^a, g^b)$ over G (one distribution for each pair g^a, g^b) such that:*

- $\text{min-ent}\left(\mathcal{X}\left(g^a, g^b\right)\right) \geq t$
- *The probability distribution $\mathcal{DH}_G$ (where G is a cyclic group of order m generated by an element g; see Assumption 1) is (S, ϵ) indistinguishable from the ensemble*

$$\mathcal{R}^* = \{(g^a, g^b, C) \text{ for } a, b \in_R \mathbb{Z}_m \text{ and } C \in_{\mathcal{X}(g^a, g^b)} G\} \tag{8}$$

Theorem 6 ([1, p.12]). *Let G be a cyclic group of order $m = m_1 m_2$ where $\gcd(m_1, m_2) = 1$, and G_1 be a subgroup of order m_1 in G. If the (S, ϵ) DDH Assumption holds over G_1 then the (S', ϵ) $\log(m_1)$-DDH Assumption holds in G, where $S' = S - 5\exp_G$ and $\exp_G$ is the circuit size required for one exponentiation in G.*

Theorem 7 ([1, p.14]). *Let G be a cyclic group of order $m = \prod_{i=1}^{l} p_i^{e_i}$ where $p_1 < \ldots < p_l$ is the prime decomposition of m. Thus G is the direct product of the subgroups G_i where each G_i has order $p_i^{e_i}$. Fix an (S, ϵ) security parameter and consider the subgroups $\{G_{j_1}, \ldots, g_{j_{l'}}\}$ which are (S, ϵ)-DDH. Then the orders of the subgroups are relatively prime with each other and G is (S', ϵ') m'-DDH where: $m' = \sum_{i=1}^{l'} e_{j_i} \log(p_{j_i})$, $S' = S - 14\exp_G$, $\epsilon' = l'\epsilon$ and $\exp_G$ is the circuit size required for one exponentiation in G.*

A.2 Short Exponent Assumptions and Theorems

The short-exponent discrete-log (DLSE) assumption allows the use of exponents much shorter than the size of the group order, but with a similar level of security to the use of full-length exponents. This may greatly increase the efficiency of the computations required. The assumption was first analysed by van Oorschot and Wiener [19], and formalized by Patel and Sundaram [20]. We give the version provided by Gennaro et al.

Assumption 3 (s-DLSE[1, p.14]) *Let G be a cyclic group generated by g and of order m. We say that the (S, ϵ) s-DLSE Assumption holds in G if for every circuit I of size $\leq S$, we have that $\Pr_{x \in_R [1 \ldots 2^s]}[I(g, m, s, g^x) = x] \leq \epsilon$.*

Gennaro et al. provide the following summary of the validity of this assumption:

> Current knowledge points to the plausibility of the above assumption even for exponents s significantly shorter than $\log(\text{ord}(g))$. The exact values of s for which the assumption seems to hold depend on the group generated by the element g. An obvious lower bound on s, if one wants to achieve security against 2^k-complexity attacks, is $s \geq 2k$ which is necessary to thwart the usual square-root attacks such as Shanks and Pollard methods. However, as pointed out in [19], there are cases where s needs to be chosen larger than $2k$. Specifically, they show how to use a Pohlig-Hellman decomposition to obtain some of the bits of the exponent. The power of the attack depends on the (relatively) small prime factors of the group order. For example, when working over $\mathbb{Z}_p^*$ with a random prime p, the [19] results indicate the use of $s \approx 4k$ (e.g., with a security parameter of 80 one should use $s = 320$ which is much shorter than the 1024 or 2048 bits of p, yet twice as much as the bare minimum of $s = 160$). If one wants to use $s = 2k$ (i.e., assume the $2k$-DLSE), it is necessary to work in special groups such as those of prime order or $\mathbb{Z}_p^*$ with p a safe prime (i.e., $p = 2q + 1$, and q prime).

Given the s-DLSE assumption, we may conclude that g raised to a short exponent is indistinguishable from a randomly chosen member of G, and hence that the t-DDH assumption may be used with short exponents[4]:

[4] Proposition 15 and Theorem 16 in the January 10, 2006 eprint version of Gennaro et al. (corresponding to Theorems 8 and 9 here) are incorrect, and they are corrected here according to details supplied by Gennaro in a personal communication. A new version of Gennaro et al.'s paper is yet to be provided.

Theorem 8 ([1, p.15]). *Let G be a cyclic group of order m generated by g, such that m is odd or $m/2$ is odd. If the (S, ϵ) s-DLSE Assumption holds in G, then the following two distributions $\mathcal{S}_G = \{g^x : x \in_{\mathrm{R}} [1 \ldots 2^s]\}$ and $\mathcal{U}_G = \{g^x : x \in_{\mathrm{R}} \mathbb{Z}_m\}$ are (S', ϵ') indistinguishable, where $\epsilon' = \epsilon\,(\log_2(m) - s)$ and $S' \approx \frac{\epsilon^3}{s \ln\left(\frac{s}{1-\epsilon}\right)} S$ where $\ln(\cdot)$ is the natural logarithm. If $\log_2(m) - s + \log_2(\epsilon) > s$ then the expression for S' may be refined to $S' \approx \frac{\epsilon^2}{s \ln\left(\frac{s}{1-\epsilon}\right)} S \geq \frac{\epsilon^2}{(\log_2(m)-s) \ln\left(\frac{s}{1-\epsilon}\right)} S$.*

Theorem 9 ([1, p.15]). *Let G be a cyclic group of order m generated by g, such that m is odd, or $m/2$ is odd. Let s, t be such that the (S_1, ϵ_1) s-DLSE and the (S_2, ϵ_2) t-DDH Assumptions hold in G. Denote with $\mathcal{X}(g^a, g^b)$ the family of distributions induced by the t-DDH assumption over G (see Assumption 2). Assume that $\mathcal{X}(g^a, g^b)$ is S_3-semi-samplable (i.e. there exists a circuit of size S_3 which is run on input either a or b and whose output distribution is $\mathcal{X}(g^a, g^b)$). Then the following two distributions*

$$\mathcal{SDH} = \{(g^a, g^b, g^{ab}) \text{ for } a, b \in_{\mathrm{R}} [1 \ldots 2^s]\} \quad \text{and} \tag{9}$$

$$\mathcal{SR}^* = \{(g^a, g^b, C) \text{ for } a, b \in_{\mathrm{R}} [1 \ldots 2^s] \text{ and } C \in_{\mathcal{X}(g^a, g^b)} G\} \tag{10}$$

are (S, ϵ) indistinguishable where $\epsilon = \epsilon_2 + 4\,(\log_2(m) - s)\,\epsilon_1$, $S = \min(S_2, S' - S_3)$ and $S' \approx \frac{\epsilon_1^3}{s \ln\left(\frac{s}{1-\epsilon_1}\right)} S_1$ (or if $\log_2(m) - s + \log_2(\epsilon_1) > s$ then the expression for S' may be refined to $S' \approx \frac{\epsilon_1^2}{s \ln\left(\frac{s}{1-\epsilon_1}\right)} S_1 \geq \frac{\epsilon_1^2}{(\log_2(m)-s) \ln\left(\frac{s}{1-\epsilon_1}\right)} S_1$).

Note that $\mathcal{X}(g^a, g^b)$ is semi-samplable given the factorization of m, and S_3 will be the time required for a multi-exponentiation in G. Therefore, the use of t-DDH with short exponents remains secure even if an adversary knows the factorization of m (it is not assumed that honest parties know the factorization of m) [1, p.13,18].

B Theorems and Proofs

This appendix provides theorems and proofs omitted from the main body due to lack of space. For completeness, we give the definition of Renyi entropy:

Definition 14 (Renyi (or collision) entropy [3, p.500]). *Let $\mathcal{X}$ be a probability distribution over A. The Renyi entropy of $\mathcal{X}$ is* $\text{Renyi-ent}(\mathcal{X}) = -\log_2(\text{Col}(\mathcal{X}))$ *where* $\text{Col}(\mathcal{X}) = \Pr_{x,x' \in_{\mathrm{R}} \mathcal{X}}(x = x') = \sum_x \left(\Pr_{\mathcal{X}}[x]\right)^2$. *Note:* $\text{min-ent}(\mathcal{X}) \leq \text{Renyi-ent}(\mathcal{X}) \leq 2 \cdot \text{min-ent}(\mathcal{X})$ *[3, p.500].*

Now, we state Theorems 10 and 11 and provide their proofs. These theorems deal with the key to a PRFF being only computationally indistinguishable from random, and the use of one PRFF with various different keys.

Theorem 10. *Suppose $f = \{f_\kappa\}_{\kappa \in K}$ is a (S_2, q, ϵ_2) PRFF, κ is chosen from a distribution, $\mathcal{X}$, that is (S_1, ϵ_1) indistinguishable from the uniform distribution, $\mathcal{U}$, over K, and S_3 is the circuit size required to evaluate f_κ on q different inputs. Then a circuit, $\mathcal{A}$, of size $\min(S_1 - S_3, S_2)$, given oracle access to either f_κ or a TRF with the same domain and range, and making at most q oracle queries, has advantage at most $\epsilon_1 + \epsilon_2$ in distinguishing whether its oracle is a random member of f or a TRF.*

Proof. The (S_1, ϵ_1) indistinguishability of $\mathcal{X}$ and $\mathcal{U}$ means that if we try to construct a distinguisher, D, of size $\leq S_1$ to tell apart $\mathcal{X}$ and $\mathcal{U}$, it cannot succeed with probability better than ϵ_1. We may construct such a distinguisher (that uses $\mathcal{A}$) by using the input, x, of D to key f, and then using this function as the oracle used to reply to oracle queries by $\mathcal{A}$. D outputs with whatever $\mathcal{A}$ outputs. Let $\delta_{\mathcal{A},\mathcal{X}} = \Pr_{x \in \mathcal{X}}[\mathcal{A}^{f_x(\cdot)} = 1]$ and let $\delta_{\mathcal{A},\mathcal{U}} = \Pr_{x \in \mathcal{U}}[\mathcal{A}^{f_x(\cdot)} = 1]$. Then we have $|\delta_{\mathcal{A},\mathcal{X}} - \delta_{\mathcal{A},\mathcal{U}}| \leq \epsilon_1$. Now let $\delta_{\mathcal{A},\mathcal{R}} = \Pr_{\mathcal{O}(\cdot) \in_{\mathrm{R}} \mathrm{Rand}}[\mathcal{A}^{\mathcal{O}(\cdot)} = 1]$. Then, by security of the pseudorandom function family, we have $|\delta_{\mathcal{A},\mathcal{U}} - \delta_{\mathcal{A},\mathcal{R}}| \leq \epsilon_2$, since the size of $\mathcal{A}$ is no greater than S_2. Hence, $|\delta_{\mathcal{A},\mathcal{X}} - \delta_{\mathcal{A},\mathcal{R}}| \leq \epsilon_1 + \epsilon_2$, and the theorem follows. □

Theorem 11. *Suppose that $\kappa_1, \kappa_2, \ldots, \kappa_q$ are each chosen independently according to a distribution, $\mathcal{X}$, that is (S_1, ϵ_1) indistinguishable from the uniform distribution, $\mathcal{U}$, over the set of keys, K, for a (S_2, q_2, ϵ_2) PRFF $f = \{f_\kappa\}_{\kappa \in K}$. Let S_3 be the size of a circuit required to evaluate a member of f on a total of q_2 different inputs. For $0 \leq j \leq q-1$, let S_4 be the maximum size of a circuit required to generate j values independently according to $\mathcal{X}$, evaluate j members of f each on a total of q_2 different inputs, and generate $(q-1-j)q_2$ independent uniformly distributed values from the range of f. Then a circuit, $\mathcal{A}$, of size $\min(S_1 - S_3, S_2) - S_4$ which is given oracle access to either $f_{\kappa_1}, \ldots, f_{\kappa_q}$ or q truly random functions with the same domain and range as the functions in f, and makes at most q_2 queries to each oracle, has advantage at most $q(\epsilon_1 + \epsilon_2)$ in distinguishing whether it has access to a random member of f or a truly random function.*

Proof. We attempt to construct a distinguisher D_j for for some j such that $0 \leq j \leq q-1$ that breaks Theorem 10. D_j runs a circuit $\mathcal{A}$ that is against Theorem 11. D_j uses its oracle to provide the $j+1^{\text{th}}$ oracle to $\mathcal{A}$. The other $q-1$ oracles that $\mathcal{A}$ needs are provided by D_j as follows. First, D_j chooses j keys, $\kappa_1, \ldots, \kappa_j$ independently according to distribution $\mathcal{X}$. The first j oracles are provided as $f_{\kappa_1}, \ldots, f_{\kappa_j}$. The remainder of the oracles are provided as truly random functions. Now when $j = 0$ and D_0 has a random function as its oracle, the view of $\mathcal{A}$ is the same as in Theorem 10 when it is provided with all random functions for its oracles. When $j = q-1$ and D_{q-1} is provided with a member of f as its oracle, the view of $\mathcal{A}$ is the same as in Theorem 10 when it is provided with q members of f for its oracles. Let $p_{j,i}$ be the probability that D_j answers 1 given that the correct answer is i, where $i = 0$ means a member of f and $i = 1$ means a truly random function. Note that $p_{k,0} = p_{k+1,1}$. Then we have:

$$\begin{aligned}
\mathbf{Adv}(A) &= |p_{0,1} - p_{q-1,0}| && (11)\\
&\leq |p_{0,1} - p_{0,0}| + |p_{1,1} - p_{1,0}| + \ldots + |p_{q-1,1} - p_{q-1,0}| && (12)\\
&= \sum_{j=0}^{q-1} \mathbf{Adv}(D_j) && (13)
\end{aligned}$$

Hence, for at least one of the D_j, the advantage of that D_j is at least $\mathbf{Adv}(A)/q$, and Theorem 11 follows. □

Next, we justify the statement that a strong randomness extractor is also a strong computational randomness extractor.

Theorem 12. *A (t, ϵ_2) strong randomness extractor $H = \{h_\kappa : A \to \{0,1\}^c | \kappa \in \{0,1\}^d\}$ is a $(t, S, \epsilon_1, S - S_h, \epsilon_1 + \epsilon_2)$ strong computational randomness extractor for any S and ϵ_1, where S_h is the maximum circuit size required to generate a random key, κ, for the randomness extractor and compute $h_\kappa(\cdot)$.*

Proof. Let the input distribution for the extractor be $\mathcal{X}$ over A, and let this distribution be (S, ϵ_1) indistinguishable from distribution $\mathcal{Y}$ over B which has min-entropy t. Then the distributions $\{(\kappa, h_\kappa(x))$ for $\kappa \in_{\mathrm{R}} \{0,1\}^d$ and $x \in_{\mathcal{X}} A\}$ and $\{(\kappa, h_\kappa(y))$ for $\kappa \in_{\mathrm{R}} \{0,1\}^d$ and $y \in_{\mathcal{Y}} B\}$ are $(S - S_h, \epsilon_1)$ indistinguishable. Otherwise, a distinguisher for these distributions can be used to distinguish whether a value z is from $\mathcal{X}$ or $\mathcal{Y}$ by giving it $(\kappa, h_\kappa(z))$ for $\kappa \in_{\mathrm{R}} \{0,1\}^d$.

By the security of H and Lemma 1, the distributions $\{(\kappa, h_\kappa(y))$ for $\kappa \in_{\mathrm{R}} \{0,1\}^d$ and $y \in_{\mathcal{Y}} B\}$ and $\{(\kappa, R)$ for $\kappa \in_{\mathrm{R}} \{0,1\}^d$ and $R \in_{\mathrm{R}} \{0,1\}^c\}$ are (∞, ϵ_2) indistinguishable, and therefore, by Lemma 2 the theorem holds. □

The following theorems justify using a single key with an extractor.

Theorem 13 (Strong randomness extractor reuse). *Let $H = \{h_\kappa : \{0,1\}^n \to \{0,1\}^c | \kappa \in \{0,1\}^d\}$ be a (t, ϵ) strong randomness extractor, let $X_1, X_2, \ldots, X_q$ denote random variables taking values in $\{0,1\}^n$, each having min-entropy at least t, let κ denote a random variable with uniform distribution in $\{0,1\}^d$, and let $R_1, R_2, \ldots, R_q$ be chosen uniformly at random from $\{0,1\}^c$. Let the random variables $h_\kappa, X_1, X_2, \ldots, X_q, R_1, R_2, \ldots, R_q$ be mutually independent. Then the following distributions are within statistical distance ϵ' from each other:*

$$(\kappa, h_\kappa(X_1), h_\kappa(X_2), \ldots, h_\kappa(X_q),) \cong_{\epsilon'} (\kappa, R_1, R_2, \ldots, R_q) \text{ where } \epsilon' = q\epsilon.$$

The following theorems are used in the proof of Theorem 13.

Theorem 14 ([8, Theorem 6.14]). *For random variables X, Y, Z, we have $\Delta[X;Z] \leq \Delta[X;Y] + \Delta[Y;Z]$*

Theorem 15 ([8, Theorem 6.16]). *Let X, Y be random variables taking values on a set $\mathcal{V}$ and let f be a function from $\mathcal{V}$ into a set $\mathcal{W}$. Then $\Delta[f(X); f(Y)] \leq \Delta[X;Y]$.*

Theorem 16 ([8, Theorem 6.17]). *Let X, Y be random variables taking values on a set $\mathcal{V}$ and let W be a random variable taking values on a set $\mathcal{W}$. Further, suppose that X and W are independent and that Y and W are independent. Then the statistical distance between (X, W) and (Y, W) is equal to the statistical distance between X and Y. That is, $\Delta[(X, W); (Y, W)] = \Delta[X;Y]$.*

We can now give the proof of Theorem 13:

Proof. The theorem may be proven in exactly the same way as Shoup's Theorem 6.22 [8], which is a special case of the above theorem for when the strong randomness extractor is a universal hash function. Shoup's proof is given below, slightly modified for the general case.

We make a hybrid argument. Define random variables $W_0, W_1, \ldots, W_q$ as follows:

$$\begin{aligned} W_0 &= (h_\kappa, h_\kappa(X_1), h_\kappa(X_2), \ldots, h_\kappa(X_q)) \\ W_i &= (h_\kappa, R_1, \ldots, R_i, h_\kappa(X_{i+1}), \ldots, h_\kappa(X_q)) \\ W_q &= (h_\kappa, R_1, \ldots, R_q) \end{aligned}$$

We have:

$$\begin{aligned} \epsilon' &= \Delta[W_0; W_q] \\ &\leq \sum_{i=1}^{l} \Delta[W_{i-1}; W_i] \quad \text{(by Theorem 14)} \\ &\leq \sum_{i=1}^{q} \Delta[\, \begin{matrix} h_\kappa, R_1, \ldots, R_{i-1}, h_\kappa(X_i), X_{i+1}, \ldots, X_q); \\ h_\kappa, R_1, \ldots, R_{i-1}, \quad R_i, \quad X_{i+1}, \ldots, X_q)] \end{matrix} \quad \text{(by Theorem 15)} \\ &= \sum_{i=1}^{q} \Delta[h_\kappa, h_\kappa(X_i); h_\kappa, R_i] \quad \text{(by Theorem 16)} \\ &\leq q\epsilon \quad \text{(by Definition 11)} \end{aligned}$$

□

Theorem 17 (Strong computational randomness extractor reuse). *Let $H = \{h_\kappa : \{0,1\}^n \to \{0,1\}^c | \kappa \in \{0,1\}^d\}$ be a $(t, S, \epsilon, S', \epsilon')$ strong computational randomness extractor, let $X_1, X_2, \ldots, X_q$ denote random variables taking values in $\{0,1\}^n$, each having (S, ϵ) computational entropy at least t, let κ denote a random variable with uniform distribution in $\{0,1\}^d$, and let $R_1, R_2, \ldots, R_q$ be chosen uniformly at random from $\{0,1\}^c$. Let the random variables $h_\kappa, X_1, X_2, \ldots, X_q, R_1, R_2, \ldots, R_q$ be mutually independent. Let S_1 be the size of a circuit that generates up to $q-1$ values from $\{0,1\}^n$ that are mutually independent and have (S, ϵ) computational entropy t, generates up to $q-1$ values uniformly at random from $\{0,1\}^c$, and evaluates h_κ on up to $q-1$ inputs for a given κ. Then the following distributions are $(S' - S_1, q\epsilon')$ indistinguishable from each other:*

$$(\kappa, h_\kappa(X_1), h_\kappa(X_2), \ldots, h_\kappa(X_q)) \text{ and } (\kappa, R_1, R_2, \ldots, R_q)$$

Proof. We use a hybrid argument. Let $\mathcal{W}_0, \mathcal{W}_1, \ldots, \mathcal{W}_q$ be defined as follows:

$$\begin{aligned} \mathcal{W}_0 &= (\kappa, h_\kappa(X_1), h_\kappa(X_2), \ldots, h_\kappa(X_q)) \\ \mathcal{W}_i &= (\kappa, R_1, \ldots, R_i, h_\kappa(X_{i+1}), \ldots, h_\kappa(X_q)) \\ \mathcal{W}_q &= (\kappa, R_1, \ldots, R_q) \end{aligned}$$

Then $\mathcal{W}_i$ and $\mathcal{W}_{i-1}$ for $1 \leq i \leq q$ are $(S' - S_1, \epsilon')$ indistinguishable, otherwise a distinguisher for these distributions may be used to distinguish $(\kappa, h_\kappa(X))$ and (κ, R), where $X \in \{0,1\}^n$ has (S, ϵ) computational entropy t and $R \in_{\mathrm{R}} \{0,1\}^c$, contrary to the definition of the strong computational randomness extractor. By applying Lemma 2, the theorem follows. □

Theorem 18 (Strong randomness extractor reuse with computationally indistinguishable distributions). *Let $H = \{h_\kappa : \{0,1\}^n \to \{0,1\}^c | \kappa \in \{0,1\}^d\}$ be a (t, ϵ_2) strong randomness extractor, let $X_1, X_2, \ldots, X_q$ denote random variables taking values in $\{0,1\}^n$, each having (S, ϵ) computational entropy at least t, let κ denote a random variable with uniform distribution in $\{0,1\}^d$, and let $R_1, R_2, \ldots, R_q$ be chosen uniformly at random from $\{0,1\}^c$. Let the random variables $h_\kappa, X_1, X_2, \ldots, X_q, R_1, R_2, \ldots, R_q$ be mutually independent. Let S_1 be the size of the circuit that generates up to $q-1$ values from $\{0,1\}^n$ that are mutually independent and have (S, ϵ) computational entropy t, and generates up to $q-1$ values from $\{0,1\}^n$ that have min-entropy t. Then the following distributions are $(S - S_1, q(\epsilon + \epsilon_2))$ indistinguishable from each other:*

$$(\kappa, h_\kappa(X_1), h_\kappa(X_2), \ldots, h_\kappa(X_q)) \quad \textit{and} \quad (\kappa, R_1, R_2, \ldots, R_q)$$

Proof. Let Y_i for $1 \leq i \leq q$ be random variables taking values in $\{0,1\}^n$, each having min-entropy t. We may use a hybrid argument to show that the distributions $(X_1, X_2, \ldots, X_q)$ and $(Y_1, Y_2, \ldots, Y_q)$ are $(S - S_1, q\epsilon)$ indistinguishable from one another, since each of the X_i have (S, ϵ) computational entropy t and are mutually independent. Then we may apply Theorem 13 and Lemmas 1 and 2 to derive the theorem. □

Here we combine Theorems 10, 11, 9 and 17 as a useful example.

Theorem 19.

- *Let G be a cyclic group of order m generated by g, such that m is odd, or $m/2$ is odd.*
- *Let s, t be such that the (S_1, ϵ_1) s-DLSE and the (S_2, ϵ_2) t-DDH Assumptions hold in G. Denote with $\mathcal{X}(g^a, g^b)$ the family of distributions induced by the t-DDH assumption over G and assume that $\mathcal{X}(g^a, g^b)$ is S_3-semi-samplable*
- *Let:*

$$\epsilon = \epsilon_2 + 4\left(\log_2(m) - s\right)\epsilon_1 \tag{14}$$

$$S = \min\left(S_2, S^* - S_3\right) \quad \textit{where} \tag{15}$$

$$S^* \approx \frac{\epsilon_1^3}{s \ln\left(\frac{s}{1-\epsilon_1}\right)} S_1 \quad \textit{unless } \log_2(m) - s + \log_2(\epsilon_1) > s\textit{; then} \tag{16}$$

$$S^* \approx \frac{\epsilon_1^2}{s \ln\left(\frac{s}{1-\epsilon_1}\right)} S_1 \geq \frac{\epsilon_1^2}{(\log_2(m) - s) \ln\left(\frac{s}{1-\epsilon_1}\right)} S_1 \tag{17}$$

- *Let $H = \{h_\kappa : \{0,1\}^n \to \{0,1\}^c | \kappa \in \{0,1\}^d\}$ be a $(t, S, \epsilon, S', \epsilon')$ strong computational randomness extractor.*
- *Let κ denote a random variable with uniform distribution in $\{0,1\}^d$.*
- *Let S_4 be the size of a circuit that generates up to $q_1 - 1$ values from $\{0,1\}^n$ that are mutually independent and have (S, ϵ) computational entropy t, generates up to $q_1 - 1$ values uniformly at random from $\{0,1\}^c$, and evaluates h_κ on up to $q_1 - 1$ inputs for a given κ.*
- *Let $f = \{f_\kappa\}_{\kappa \in K}$ be a (S_5, q_2, ϵ_5) PRFF.*
- *Let S_6 be the size of a circuit required to evaluate a function from f on a total of q_2 different inputs.*
- *Let $a_1, a_2, \ldots, a_{q_1}, b_1, b_2, \ldots, b_{q_1} \in_{\mathrm{R}} [1 \ldots 2^s]$ and mutually independent.*
- *Let $S_7 = \min(S' - S_4 - S_6, S_5)$ and let $\epsilon_7 = q_1\epsilon' + \epsilon_5$.*
- *Let $Z_{1,1}, \ldots, Z_{q_1,q_2}$ be any publicly known values in the domain of f.*
- *Let $R_1, R_2, \ldots, R_{q_1}$ be randomly chosen from the range of h.*
- *Let $T_{1,1}, \ldots, T_{q_1,q_2}$ be randomly chosen from the range of f.*

– *For $0 \leq j \leq q_1 - 1$, let S_8 be the maximum size of a circuit required to generate j mutually independent values of the form g^{ab} where $a, b \in_{\mathrm{R}} [1 \ldots 2^s]$, evaluate j members of f each on a total of q_2 different inputs, and generate $(q_1 - 1 - j)q_2$ independent uniformly distributed values from the range of f.*
– *Let $S_9 = S_7 - S_8 = \min(S' - S_4 - S_6, S_5) - S_8$ and let $\epsilon_9 = q_1\epsilon_7 = q_1\left(q_1\epsilon' + \epsilon_5\right)$.*
– *Let the notation $\langle\eta(i)\rangle_{i=\alpha}^{\beta}$ denote the sequence $\eta(\alpha), \eta(\alpha+1), \ldots, \eta(\beta)$ where η is some function of i.*

Then for any j such that $1 \leq j \leq q_1$, the following two distributions are (S_7, ϵ_7) indistinguishable:

$$\mathcal{EEDH} = \left(\kappa,\ \langle g^{a_i}, g^{b_i}\rangle_{i=1}^{q_1},\ \langle h_\kappa(g^{a_i b_i})\rangle_{i=1}^{j-1},\ \langle h_\kappa(g^{a_i b_i})\rangle_{i=j+1}^{q_1},\ \langle Z_{i,j},\, f_{h_\kappa(g^{a_j b_j})}(Z_{i,j})\rangle_{i=1}^{q_2}\right)$$

$$\mathcal{EER} = \left(\kappa,\ \langle g^{a_i}, g^{b_i}\rangle_{i=1}^{q_1},\ \langle R_i\rangle_{i=1}^{j-1},\ \langle R_i\rangle_{i=j+1}^{q_1},\ \langle Z_{i,j},\, T_{i,j}\rangle_{i=1}^{q_2}\right)$$

Furthermore, the following two distributions are (S_9, ϵ_9) indistinguishable:

$$\mathcal{EEDH}^* = \left(\kappa,\ \langle g^{a_i}, g^{b_i}\rangle_{i=1}^{q_1},\ \ \langle\langle Z_{i,j},\, f_{h_\kappa(g^{a_j b_j})}(Z_{i,j})\rangle_{i=1}^{q_2}\rangle_{j=1}^{q_1}\right)$$

$$\mathcal{EER}^* = \left(\kappa,\ \langle g^{a_i}, g^{b_i}\rangle_{i=1}^{q_1},\ \ \langle\langle Z_{i,j},\, T_{i,j}\rangle_{i=1}^{q_2}\rangle_{j=1}^{q_1}\right)$$

From the above theorem we may see what it is that influences the level of security provided by the outputs of the randomness expander. The main ones are how many times the randomness extractor is used with the one key (q_1), how good the output of the randomness extractor is (measured by (S', ϵ')), and how good the randomness expander is (measured by (S_5, ϵ_5)).

If we require a security level of k bits, and wish $\mathcal{EEDH}$ and $\mathcal{EER}$ to be indistinguishable, then we require $\frac{S_7}{\epsilon_7} \geq 2^k$. We may achieve this by requiring $\frac{S' - S_4 - S_6}{\epsilon'} \geq 2^{k+1}q_1$ and $\frac{S_5}{\epsilon_5} \geq 2^{k+1}$ since these conditions imply that $\frac{S_7}{\epsilon_7} \geq \frac{\min\left(2^{k+1}q_1\epsilon',\ 2^{k+1}\epsilon_5\right)}{q_1\epsilon' + \epsilon_5}$, for any values of ϵ' and ϵ_5 and by setting $q_1\epsilon' = \epsilon_5$ we achieve the desired result.

Similarly, if we require a security level of k bits, and wish $\mathcal{EEDH}^*$ and $\mathcal{EER}^*$ to be indistinguishable, then we require $\frac{S_9}{\epsilon_9} \geq 2^k$. This may be achieved by requiring $\frac{S' - S_4 - S_6 - S_8}{\epsilon'} \geq 2^{k+1}q_1^2$ and $\frac{S_5 - S_8}{\epsilon_5} \geq 2^{k+1}q_1$.

Let us assume that $S_4 \approx q_1$, $S_6 \approx q_2$ and $S_8 \approx q_1 q_2$. If h is a $(t, 2^{-e})$ strong randomness extractor instead of a $(t, S, \epsilon, S', \epsilon')$ strong computational randomness extractor, then by Theorem 12 (and ignoring the time to generate a key for h) we have $\frac{S' - S_4 - S_6}{\epsilon'} \approx \frac{S - q_1 - q_2}{\epsilon + 2^{-e}}$ which we require $\geq 2^{k+1}q_1$ or $\frac{S' - S_4 - S_6 - S_8}{\epsilon'} \approx \frac{S - q_1 - q_2 - q_1 q_2}{\epsilon + 2^{-e}}$ which we require $\geq 2^{k+1}q_1^2$. These requirements are satisfied with either $e \geq k + 2 + \log_2(q_1)$ and $\frac{S - q_1 - q_2}{\epsilon} \geq 2^{k+2}q_1$ or $e \geq k + 2 + 2\log_2(q_1)$ and $\frac{S - q_1 q_2}{\epsilon} \geq 2^{k+2}q_1^2$. The requirement on e must be met by choosing an appropriate randomness extractor, and the other requirement may be met by requiring either $\frac{S_2 - q_1 - q_2}{\epsilon_2} \geq 2^{k+3}q_1$ and $\frac{S^* - S_3 - q_1 - q_2}{\epsilon_1} \geq (\log_2(m) - s)\,2^{k+5}q_1$ or $\frac{S_2 - q_1 q_2}{\epsilon_2} \geq 2^{k+3}q_1^2$ and $\frac{S^* - S_3 - q_1 q_2}{\epsilon_1} \geq (\log_2(m) - s)\,2^{k+5}q_1^2$. The requirement on S_2 and ϵ_2 may be met by choosing G such that the $\left(2^{k+4}q_1 + q_1 + q_2, \frac{1}{2}\right)$ and $\left(q_1 + q_2 + 1, \frac{1}{2^{k+3}q_1}\right)$ t-DDH assumptions hold or the $\left(2^{k+4}q_1^2 + q_1 q_2, \frac{1}{2}\right)$ and $\left(q_1 q_2 + 1, \frac{1}{2^{k+3}q_1^2}\right)$ t-DDH assumptions hold. Let $Y \stackrel{\text{def}}{=} (\log_2(m) - s)\,2^{k+5}q_1$ or $(\log_2(m) - s)\,2^{k+5}q_1^2$, whichever is appropriate, let $Z \stackrel{\text{def}}{=} S_3 + q_1 + q_2$ or $S_3 + q_1 q_2$, whichever is appropriate, and let $S^* = \frac{\epsilon_1^i}{s\ln\left(\frac{s}{1-\epsilon_1}\right)}S_1$ for $i = 2$ or 3, whichever is appropriate. Then for a large value of ϵ_1, e.g. $\frac{1}{2}$, we require:

$$\frac{S^* - Z}{\epsilon_1} \geq Y \tag{18}$$

$$\frac{S_1}{2^{i-1}s\ln(2s)} \geq Y + 2Z \quad \text{for } \epsilon_1 = \frac{1}{2} \tag{19}$$

$$S_1 \geq 2^{i-1}s\ln(2s)\,(Y + 2Z) \tag{20}$$

However, for a small value of ϵ_1, the requirement is much stronger. We consider $\epsilon_1 = \frac{1}{Y}$ to be the smallest sensible value of ϵ_1, since we are in effect requiring a security level of $\log_2(Y)$ bits for this part. Then we

have:

$$\frac{S^* - Z}{\epsilon_1} \geq Y \tag{21}$$

$$\frac{YS_1}{Y^i s \ln(s)} \geq Y + YZ \quad \text{for } \epsilon_1 = \frac{1}{Y} \tag{22}$$

$$S_1 \geq Y^i s \ln(s)(Z+1) \tag{23}$$

Hence G must be chosen such that the $\left(2^{i-1} s \ln(2s)(Y+2Z), \frac{1}{2}\right)$ and $\left(Y^i s \ln(s)(Z+1), \frac{1}{Y}\right)$ s-DLSE assumptions hold.

These requirements are summarized in Sect. 4.1. The theorem may be used to prove security in a variety of situations; for example, to create two or more keys from a shared secret Diffie-Hellman value, the holders of the value may use a randomness extractor to extract some random bits from the Diffie-Hellman value, and then use the extracted bits to key a pseudorandom function family. Finally, the value of the function at two or more points may be evaluated to generate the values of the required secret keys. Alternatively, the randomness extracted using the randomness extractor may be used directly for keying material (i.e. the use of a randomness expander may be omitted). However, this approach is unlikely to be attractive unless only one key is required, since a large amount of min-entropy is required of the DH value to be able to extract a sufficient number of bits to be used as the keying material. Table 6 provides further details of the example in Sect. 4.1 applying Theorem 19.

C Pseudorandom Function Family Choices and Security Levels

This appendix discusses the merits of the available pseudorandom function family options. We observe that because a pseudorandom function family is a secure MAC, the security of a MAC scheme (such as CBC-MAC or HMAC) may be proven by showing that it is a secure pseudorandom function family. This means that any attacks against the MAC are also attacks against the pseudorandomness of the function family, and thus the complexity of such attacks is of interest when evaluating the security level of the MAC when used as a pseudorandom function family; see Bellare et al. [6, Prop. 2.7] for details.

The security of some of the options below is based on the assumption that the compression function, $h : \{0,1\}^c \times \{0,1\}^b \to \{0,1\}^c$, with block length b and key length c, of the underlying hash function used with the option is itself a pseudorandom function family. That is, we are assuming that $\mathbf{Adv}^{\text{prf}}_h(q, S)$ is small. However, if we are to find the security level in bits provided by the options below, we must make an assumption on how small $\mathbf{Adv}^{\text{prf}}_h(q, S)$ is.

Let h_{R} be a family of 2^c functions where each function is chosen at random from the set of all functions with the same domain and range. Then Bellare et al. [9, Sect. 6] claim that when an adversary is given black box access to the function family h_{R} (i.e. the adversary does not have a description of h_{R}, but can request the function output for a particular key and input pair), $\mathbf{Adv}^{\text{prf}}_{h_{\text{R}}}(q, S) \leq \frac{c'S}{2^c}$ where $q \leq S$ for some $c' > 0$. In the analysis that follows, we therefore use the assumption that $\mathbf{Adv}^{\text{prf}}_h(q, S) \leq \frac{S}{2^c}$ as a minimum assumption. However, when q is not small (e.g. > 2), it seems prudent to assume that it might be possible to attack the compression function, h, of an actual cryptographic hash function more efficiently than the function family h_{R}. Therefore, we examine how the results are altered when the assumption that $\mathbf{Adv}^{\text{prf}}_h(q, S) \leq \frac{S^2}{2^c}$ is made instead, and regard this as a more conservative option.

C.1 CBC-MAC

Bellare et al. [6] have provided a proof that CBC-MAC is a secure pseudorandom function family if the underlying block cipher is a secure pseudorandom permutation family and the input length is constant. One of a number of slight modifications can be made to cater for the case where the input length is not constant. Their main theorem states:

Security level	$k = 80$
(S_5, q_2, ϵ_5) PRFF	$f = \{f_\lambda\}_{\lambda \in K}$
with q_2 queries per (secret) λ	$q_2 = 1$
and $S_5/\epsilon_5 \geq 2^{k+1}$	$\Rightarrow f$ has 81 bit security level, e.g. CBC-MAC with 128 bit key
$(t, 2^{-e})$ strong rand. ext.	$H = \{h_\kappa\}_{\kappa \in \{0,1\}^d}$
with q_1 queries per (public) κ	$q_1 = 1$
and $e \geq k + 2 + \log_2(q_1)$	$\Rightarrow e \geq 82$; e.g. H a universal hash function with 128 bits output; in that case,
Min-entropy of pre-secret is at least	$t = 292$
Value of Y	$Y = (\log_2(m) - s)\, 2^{k+5} q_1$
	$\leq 2^{93.2}$ (obtained by ignoring s and approximating $\log_2(m)$ with 292)
Value of Z	$Z = S_3 + q_1 + q_2 \approx 1$
t-DDH assumption one	$\left(2^{k+4} q_1 + q_1 + q_2, \frac{1}{2}\right) = (2^{84}, \frac{1}{2})$ t-DDH assumption
t-DDH assumption two	$\left(q_1 + q_2 + 1, \frac{1}{2^{k+3} q_1}\right) = \left(3, \frac{1}{2^{83}}\right)$ t-DDH assumption
Order of cyclic group G	$m = 2^{\max(t, 2 \cdot 85)}$ (where 85 is from t-DDH assumptions)
	$= 2^{\max(292, 2 \cdot 85)} = 2^{292}$

To use short exponents with $i = 3$ (optional):

- $\left(2^{i-1} s \ln(2s)\,(Y + 2Z), \frac{1}{2}\right) \approx (2^{95.2} s \ln(2s), \frac{1}{2})$ s-DLSE assumption $\Rightarrow s \geq 2 \log_2\left(2^{96.2} s \ln(2s)\right) \Rightarrow s \geq 213$
- $\left(Y^i s \ln(s)\,(Z+1), \frac{1}{Y}\right) \approx \left(2^{3 \cdot 93.2 + 1} s \ln(s), \frac{1}{2^{93.2}}\right) = \left(2^{280.6} s \ln(s), \frac{1}{2^{93.2}}\right)$ s-DLSE assumption
  $\Rightarrow s > 2 \log_2\left(2^{280.6} s \ln(2s)\right) \Rightarrow s > 585$, but s would probably need to be even larger, as we have not taken into account the $\frac{1}{2^{93.2}}$ value in this calculation; since $585 > \log_2(m) = 292$, it is better not to use short exponents.

To use short exponents with $i = 2$ (optional):

- We need $\log_2(m) > 2s - \log_2(\epsilon_1) \Rightarrow m \geq 2^{679}$ (because $s \geq 292$ and $\epsilon_1 \geq 2^{-95}$)
- $Y \approx 2^{95}$ since $Y = (\log_2(m) - s)\, 2^{k+5} q_1$ (ignore s; assume $\log_2(m) \geq 679$)
- $\left(2^{i-1} s \ln(2s)\,(Y + 2Z), \frac{1}{2}\right) \approx \left(2^{96} s \ln(2s), \frac{1}{2}\right)$ s-DLSE assumption $\Rightarrow s \geq 2 \log_2\left(2^{97} s \ln(2s)\right)$
- $\left(Y^i s \ln(s)\,(Z+1), \frac{1}{Y}\right) \approx \left((\log_2(m) - s)\, 2^{85} s \ln(s), \frac{1}{(\log_2(m) - s) 2^{85}}\right)$ s-DLSE assumption; To satisfy the weaker $\left((\log_2(m) - s)\, 2^{85} s \ln(s), \frac{1}{2}\right)$ s-DLSE assumption, we need $s \geq 399$ which requires $m \geq 892$. If we consider the $\left(Y^i s \ln(s), \frac{1}{Y}\right) \approx \left(Y^{i+1} s \ln(s), \frac{1}{2}\right)$ s-DLSE assumption, we need $m \geq 1276$ and $s \geq 2 \log_2\left(2^{255} (1276 - s)^3 s \ln(s)\right)$, which needs $s \geq 591$. Hence, G needs an order at least 896 bits, but probably 1276 bits, and short exponents must be at least 399 bits, but probably 591 bits, depending on whether a better analysis of how to satisfy the $\left(Y^i s \ln(s), \frac{1}{Y}\right)$ s-DLSE assumption is available for the group under consideration. In this example it is better to use a group with order 292 bits and full length exponents, since $292 < s$.

Table 6. Example combining randomness extraction and expansion

Theorem 20. *Let f be a block cipher with block length b and CBC^l-f be the family of CBC-MAC functions constructed using that block cipher on inputs of l blocks. Then*

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{CBC}^l\text{-}f}(q,S) \leq \mathbf{Adv}^{\mathrm{prp}}_{f}(q',S') + \frac{q^2l^2}{2^{b-1}} \tag{24}$$

where $q' = lq$ and $S' = S + O(lqb)$ and the output length of the CBC-MAC is b bits.

If the block cipher to be used with CBC-MAC is AES-128, AES-192, or AES-256, then the block length, b, will be 128 bits for each of these ciphers [10]. For CBC-MAC to provide a security level of k bits as a pseudorandom function family, we will require that $S/\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{CBC}^l\text{-}f}(q,S) \geq 2^k$ for all $S, q, l \geq 1$ and assume $S \geq ql$. We then require $\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{CBC}^l\text{-}f}(q,S)/S \leq 1/2^k$ which will be satisfied if $\mathbf{Adv}^{\mathrm{prp}}_{f}(q',S')/S \leq 1/2^{k+1}$ and $\frac{q^2l^2}{2^{b-1}}/S \leq 1/2^{k+1}$. We then have

$$\frac{q^2l^2}{2^{b-1}}/S \leq \frac{q^2l^2}{2^{b-1}ql} = \frac{ql}{2^{b-1}} \tag{25}$$

$$\text{and so want } \frac{ql}{2^{b-1}} \leq \frac{1}{2^{k+1}} \tag{26}$$

$$\Rightarrow \quad ql \leq 2^{b-k-2} \tag{27}$$

We consider it likely that when (27) is satisfied, then we will also have $\mathbf{Adv}^{\mathrm{prp}}_{f}(q',S')/S \leq 1/2^{k+1}$. Therefore, when ql is small (e.g. 2), we have a security level of about $k = b - 3$ bits. Otherwise, if we have $ql \leq 2^k$ (which we are assuming when we consider a security level of 2^k sufficient), then we will require $k \leq (b-2)/2$.

The level of security provided by CBC-MAC when used in conjunction with any of these ciphers will therefore be no greater than 125 bits, and will be less for values of q and l larger than 1. Hence, CBC-MAC is likely to be an acceptable choice of randomness expander for security levels of 80 bits if the number of queries to randomness expander with a single key is small and the length of each query is also small, but inadequate for security levels of 128 bits and higher. If an unlimited number of queries or queries with a very large length are able to be made by the adversary to the randomness expander with a single key, the security level will only be $(b-2)/2 = 63$ bits when $b = 128$. A security level of only $(b-2)/2$ bits is consistent with the complexities of known forgery attacks against CBC-MAC provided by Brincat and Mitchell [21], since their attacks require $q \geq 2^{b/2}$.

C.2 HMAC

The security of HMAC has been analysed by Bellare [11], who has proven that HMAC is a secure pseudorandom function if the compression function of the underlying hash function is a pseudorandom function. Here we provide a summary of the combination of Theorem 3.3 and Lemmas 5.1 and 5.2 from that paper which shows the level of security provided by HMAC.

Theorem 21. *Let $h : \{0,1\}^c \times B \rightarrow \{0,1\}^c$ be the compression function of the hash function used in conjunction with HMAC. Let $\bar{h} : B \times \{0,1\}^c \rightarrow \{0,1\}^c$ be defined as $\bar{h}(m,\kappa) = h(\kappa, m)$. Assume $b \geq c$ and let $B = \{0,1\}^b$. Let $\mathcal{A}$ be a PRF adversary against* $\mathrm{HMAC} : \{0,1\}^b \times B^+ \rightarrow \{0,1\}^c$ *that has circuit size at most S and makes $q \geq 2$ oracle queries, each of at most l blocks. Let* IV *be the fixed initialization vector specified by the hash function used in conjunction with HMAC for use with its compression function. Then there exists a related-key-advantage (RKA) adversary $\mathcal{A}_{\bar{h}}$ against $\bar{h}$, a PRF adversary $\mathcal{A}'_{\bar{h}}$ against $\bar{h}$, and PRF adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ against h such that:*

$$\begin{aligned}\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{HMAC}}(\mathcal{A}) \leq\ & \mathbf{Adv}^{\mathrm{rka}}_{\bar{h}}(\mathcal{A}_{\bar{h}}) + 2 \cdot \mathbf{Adv}^{\mathrm{prf}}_{\bar{h}}(\mathcal{A}'_{\bar{h}}) \\ & + \mathbf{Adv}^{\mathrm{prf}}_{h}(\mathcal{A}_1) + \binom{q}{2}\left[2l \cdot \mathbf{Adv}^{\mathrm{prf}}_{h}(\mathcal{A}_2) + \frac{1}{2^c}\right]\end{aligned} \tag{28}$$

Adversary $\mathcal{A}_{\bar{h}}$ has circuit size at most S and makes only 2 oracle queries which return $\mathcal{O}_{\bar{h}}(\kappa \oplus \text{opad}, \text{IV})$ and $\mathcal{O}_{\bar{h}}(\kappa \oplus \text{ipad}, \text{IV})$, where κ is unknown to $\mathcal{A}_{\bar{h}}$, opad *and* ipad *are bit strings specified in the HMAC description, and the objective of $\mathcal{A}_{\bar{h}}$ is to determine whether $\mathcal{O}_{\bar{h}}$ is $\bar{h}$ or a truly random function (i.e. $\mathcal{A}_{\bar{h}}$ is essentially a PRF adversary that is able to make a related key query).*

Adversary $\mathcal{A}'_{\bar{h}}$ has circuit size at most S and makes only 1 oracle query, this being IV*. Adversary $\mathcal{A}_1$ has circuit size at most S and makes at most q oracle queries and $\mathcal{A}_2$ has circuit size at most $O(mS_h)$ and makes at most 2 oracle queries, where S_h is the circuit size for one computation of h.*

For HMAC to provide a security level of k bits as a pseudorandom function family, we will require that $\mathbf{Adv}^{\text{prf}}_{\text{HMAC}}(\mathcal{A})/S \leq 1/2^k$ for all $S, q, l \geq 1$ and assume $S \geq qm$.

We will assume $\mathbf{Adv}^{\text{rka}}_{\bar{h}}(\mathcal{A}_{\bar{h}})$ and $\mathbf{Adv}^{\text{prf}}_{\bar{h}}(\mathcal{A}'_{\bar{h}})$ are no more than $\frac{S}{2^c}$, and $\mathbf{Adv}^{\text{prf}}_{h}(\mathcal{A}_2)$ is no more than $\frac{mS_h}{2^c}$, since only one or two oracle queries can be made by these adversaries (the assumption on h was discussed previously; we assume that it is reasonable to make a similar assumption on $\bar{h}$). We will assume $\mathbf{Adv}^{\text{prf}}_{h}(\mathcal{A}_1)$ is no more than $\frac{S^2}{2^c}$ since $\mathcal{A}_1$ may make q oracle queries, with no restriction on q other than the requirement that $qm \leq S$. We also assume $S_h = 1$ unit. Combining these assumptions with (28) we find that:

$$\frac{\mathbf{Adv}^{\text{prf}}_{\text{HMAC}}(\mathcal{A})}{S} \leq \frac{S}{S2^c} + 2 \cdot \frac{S}{S2^c} + \frac{S^2}{S2^c} + \binom{q}{2}\left[2l \cdot \frac{mS_h}{S2^c} + \frac{1}{S2^c}\right] \tag{29}$$

$$\leq \frac{3}{2^c} + \frac{S}{2^c} + \frac{q^2l^2}{S2^c} + \frac{q^2}{S2^{c+1}} \tag{30}$$

By requiring that each term in (30) be no more than $\frac{1}{2^{k+2}}$, we can achieve a security level of k bits. This translates to the following requirements:

$$\begin{array}{llll} \frac{3}{2^c} \leq \frac{1}{2^{k+2}} & \frac{S}{2^c} \leq \frac{1}{2^{k+2}} & \frac{q^2l^2}{S2^c} \leq \frac{1}{2^{k+2}} & \frac{q^2}{S2^{c+1}} \leq \frac{1}{2^{k+2}} \\ 1 \leq 2^{c-k-4} & S \leq 2^{c-k-2} & \frac{q^2l^2}{qm2^c} \leq \frac{1}{2^{k+2}} & \frac{q^2}{q2^{c+1}} \leq \frac{1}{2^{k+2}} \\ & & qm \leq 2^{c-k-2} & q \leq 2^{c-k-1} \end{array}$$

Hence, when $k \leq c-4$ and $S \leq 2^{c-k-2}$, we assume k bits of security for HMAC. In assuming a k-bit security level is adequate, we assume that $S > 2^k$ is infeasible. Hence, we may refine the requirement $S \leq 2^{c-k-2}$ to be $k \leq \frac{c-2}{2}$. However, if q and l are small (e.g. 2), and if we estimate $\mathbf{Adv}^{\text{prf}}_{h}(\mathcal{A}_1)$ with $\frac{S}{2^c}$, we may achieve a security level of around $c-4$ bits instead of $\frac{c-2}{2}$ bits.

We observe that if the usual implementation of HMAC is used, the above analysis assumes that the key provided to HMAC is the same length as a block for the underlying hash function (i.e. b bits). To achieve a shorter key, the function NMAC may be used, which is similar to HMAC but differs in how the keying material is used. In that case only $2c$ bits of key are required, where c is the length of the output of the hash function. However, NMAC is generally used for analysis of HMAC only, and availability of an existing implementation is therefore unlikely. If it is decided to implement NMAC, the implementation will require access to the compression function underlying the hash function to be used, and such access may be difficult to acquire. The security of NMAC as a PRF is given by the second line of the right hand side of (28).

Hash functions likely to be used with HMAC include MD5 [12], RIPEMD-160 [13], SHA-1, SHA-256, SHA-384 and SHA-512 [14]. Table 1 shows the block size, output length and HMAC security level for each of these algorithms. The traditional security level is $c/2$ bits, due to the existence of the birthday based forgery attacks against iterated MACs [15] that require $2^{c/2}$ oracle queries.

C.3 The Cascade Construction

Bellare, Canetti and Krawczyk [9] have provided a proof of pseudorandom function family security for cryptographic hash functions such as SHA-1 or SHA-256 based on the cascade construction, but with the fixed IV (initialization vector) replaced with a random key, provided the input is prefix-free and the underlying compression function used by the hash function is a pseudorandom function family. A proof of the pseudorandomness of another construction (F^{acsc}) to remove the prefix-free requirement is also provided in the

same paper, but this construction requires extra keying material and is not considered further here, since inputs to the randomness expander are likely to be of a fixed length in our setting. Another construction was provided to improve the security of the cascade construction using randomization (F^{rcsc}), but if the extra randomness required is counted as part of the key, then this construction requires more keying material than HMAC for a similar security level[5].

The theorem stating the security of the cascade construction may be summarized as follows:

Theorem 22. *Let $h : \{0,1\}^c \times \{0,1\}^b \to \{0,1\}^c$ be a compression function with key length c bits and block length b bits. Let H be the hash function built from h using the cascade construction, and let the number of blocks input to H be l, and let the queries to H are prefix-free. Then:*

$$\mathbf{Adv}_H^{\text{prf}}(q, S) \leq ql\mathbf{Adv}_h^{\text{prf}}(q, S') \tag{31}$$

where $S' = S + c'q(l + c + b)(S_h + \log(q))$, S_h is the size of a circuit evaluating h and c' is a small constant.

We will assume $S \geq lq$ and $S' \leq S + 2^{20}q$, since if $c' = 1$, $c = 512$, $b = 1024$, $l = 10$, $\log(q) \leq 512$, $S_h = 1$ then we have $c'q(l+c+b)(S_h+\log(q)) < 2^{20}q)$. As discussed above, we then assume $\mathbf{Adv}_h^{\text{prf}}(q, S') \leq \frac{S+2^{20}q}{2^c}$, and so have $\mathbf{Adv}_H^{\text{prf}}(q, S)/S \leq ql\frac{S+2^{20}q}{S2^c} \leq \frac{2^{20}S}{2^c}$, which we want to be no more than $\frac{1}{2^k}$ for a security level of k bits. This translates to requiring $k \leq \frac{c-20}{2}$. However, when $q, l \leq 2$, it would be possible to achieve a security level of $k = c - 2$ bits with the same assumption on $\mathbf{Adv}_h^{\text{prf}}(q, S')$, provided c is larger than 20.

Alternatively, if we make the more conservative assumption that $\mathbf{Adv}_h^{\text{prf}}(q, S') \leq \frac{\left(S+2^{20}q\right)^2}{2^c}$, then we have $\mathbf{Adv}_H^{\text{prf}}(q, S)/S \leq ql\frac{\left(S+2^{20}q\right)^2}{S2^c}$, giving a security level of $k \leq \frac{c-40}{3}$ bits. Table 1 shows the security levels achieved for various hash functions under these assumptions.

C.4 Other Alternatives

Another pseudorandom function family that has recently been proposed is called 3C. Its proposal includes a proof of security [22], and it allows the input to be of variable length. It is similar to the cascade construction, but processes an extra block of data at the end which is dependent on all of the previous outputs of the compression function. It is a useful way to remove the prefix-free requirement from the cascade construction, but the level of security proven for 3C is worse than that of the cascade construction when all hash queries are prefix-free. Therefore, we do not discuss 3C any further.

An alternative to using functions like CBC-MAC and SHA256 to provide PRFFs is to use a cryptographically secure pseudorandom bit generator instead, such as the Blum-Blum-Shub pseudorandom bit generator [23] with its efficiency improvement [24]. Sidorenko and Schoenmakers [25] have analysed the concrete security of this generator, showing how to select the size of the modulus and number of bits extracted to reach a desired security level with minimum computational effort. (Other cryptographically secure pseudorandom bit generators are also available; see Menezes et al. [26].) However, such a strategy requires a key or seed of size around 1024 bits, if one desires to achieve security similar to 1024-bit RSA. Furthermore, since there are two parties who need to use the generator, they must share two secret 512-bit primes in addition to the seed. Hence this option is not considered further here due to the large amount of keying material that would need to be generated by the randomness extractor, even though its security assumptions (e.g. integer factorization being intractable) may have withstood more scrutiny than the assumptions of the other schemes above (e.g. that the compression function of a particular hash function is pseudorandom).

D Extractor Results of Dodis et al. and their Problems

Dodis et al. [3] have analysed the security of CBC-MAC, the cascade construction and HMAC, and were the first to do so in the standard model. Their results for CBC-MAC are of little practical use, since a

[5] Assuming $\mathbf{Adv}_h^{\text{prf}}(q, S) \leq \frac{S}{2^c}$ we would have $\mathbf{Adv}_{F^{\text{rcsc}}}^{\text{prf}}(q, S)/S \leq \frac{S}{2^c}$ which we would want no greater than $\frac{1}{2^k}$ for a security level of k bits. This translates to requiring $k \leq c/2$. However, F^{rcsc} would require $b + c$ bits of keying material, whereas HMAC requires only b bits.

security level of only $e = 62$ bits can be achieved using existing AES block ciphers, which all have a block size of 128 bits. To achieve a security level of $e = 82$ bits for the extractor, a block size of at least 169 bits would be required. Their other results either use the random oracle model (which we wish to avoid) or have unreasonable assumptions or lemmas/propositions. The unreasonable assumptions or lemmas/propositions are (using the same notation as Sect. 4.2):

- in an analysis of the cascade construction, that the cascade construction is δ-almost universal (with $\delta \leq \frac{1}{2^c} + \frac{L}{2^{c+\omega}} + O\left(\frac{L^2}{2^{2c}}\right)$ where ω is the min-entropy of the last block of the input to the hash function) when defined over a family of compression functions that are random;
- in an alternative analysis of cascade construction, that the family of compression functions are δ-almost universal (with $\delta \leq \frac{1}{2^c} + \frac{1}{2^{2c}}$ which, because the key length and output length of the compression functions are equal, equates to assuming $\delta \leq \frac{1}{2^c}$, i.e. the family of compression functions is a universal hash function family);
- in an analysis of HMAC, that the cascade construction is δ-almost universal with $\delta \leq \frac{L}{2^c} + O\left(\frac{L^3}{2^{2c}}\right)$, and that the *truncated* compression function family is universal.

We note that other authors have briefly mentioned that the work of Dodis et al. is problematic: Fouque et al. [4] mentioned that Dodis et al. assume that the compression functions of the hash-based constructions under review are a family of almost universal hash functions, which is not realistic; Chevassut et al. [2] note that the analysis of Dodis et al. is incomplete because of the problem of assuming the compression functions are universal.

E Universal Hash Function Examples

Nevelsteen and Preneel [27] have provided a survey of universal hash functions (UHFs). Examples of UHFs include the following:

- [8, p.127, Ex.6.30] Let $\mathcal{A}$ be $\mathbb{Z}_n^{\times(t+1)}$ and $\mathcal{Z}$ be $\mathbb{Z}_n$ where n is prime. Let $\mathcal{H} = \{h_{x_1,\ldots,x_t} : x_1, \ldots, x_t \in \mathbb{Z}_n\}$ be a family of hash functions from $\mathcal{A}$ to $\mathcal{Z}$, where for $h_{x_1,\ldots,x_t} \in \mathcal{H}$, and for $(a_0, a_1, \ldots, a_t) \in \mathcal{A}$ we define
$$h_{x_1,\ldots,x_t}(a_0, a_1, \ldots, a_t) = a_0 + a_1 x_1 + \cdots + a_t x_t. \tag{32}$$
Show that $\mathcal{H}$ is universal, but not pairwise independent.
- [8, p.127, Ex.6.31]Let k be a prime and let n be any positive integer. Let $\mathcal{A} = \{0, \ldots, k-1\}$ and $\mathcal{Z} = \{0, \ldots, n-1\}$. Let $\mathcal{H} = \{h_{x,y} : x = 1, \ldots, k-1,\ y = 0, \ldots, k-1\}$ be a family of hash functions from $\mathcal{A}$ to $\mathcal{Z}$, where for $h_{x,y} \in \mathcal{H}$ and for $a \in \mathcal{A}$ we define
$$h_{x,y}(a) = ((ax + y) \pmod k) \pmod n. \tag{33}$$
Show that $\mathcal{H}$ is universal. (A hint is included in the original.)
- Multiplying a Toeplitz matrix (one with constant diagonals) by the input to create the output [17] is a UHF. Let the key be $\kappa = (r_{c\,1}, r_{c-1\,1}, \ldots, r_{2\,1}, r_{1\,1}, r_{1\,2}, \ldots, r_{1\,b-1}, r_{1\,b})$ (having $b + c - 1$ bits), defining matrix
$$R = \begin{bmatrix} r_{1\,1} & r_{1\,2} & \cdots & r_{1\,b} \\ r_{2\,1} & r_{1\,1} & \cdots & r_{1\,b-1} \\ \vdots & \vdots & \ddots & \vdots \\ r_{c\,1} & r_{c-1\,1} & \cdots & r_{1\,b-c+1} \end{bmatrix} \tag{34}$$
Then multiplying this matrix by the input as a column vector of b bits to produce a column vector of c bits is a universal hash function.

Examples of these UHFs are shown in Tables 7 and 8.

Shoup Ex. 6.30; $n = 3$, $t = 1$

$h_{x_1}(a_0, a_1) = a_0 + a_1 x_1 \pmod 3$

	a_1	0	0	0	1	1	1	2	2	2
	a_0	0	1	2	0	1	2	0	1	2
x_1										
0		0	1	2	0	1	2	0	1	2
1		0	1	2	1	2	0	2	0	1
2		0	1	2	2	0	1	1	2	0

Toeplitz example

$$\begin{bmatrix} r_{1\,1} & r_{1\,2} \\ r_{2\,1} & r_{1\,1} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Output is $2b_1 + b_2$

		a_1	0	0	1	1
		a_2	0	1	0	1
$r_{2\,1}$	$r_{1\,1}$	$r_{1\,2}$				
0	0	0	0	0	0	0
0	0	1	0	2	0	2
0	1	0	0	1	2	3
0	1	1	0	3	2	1
1	0	0	0	0	1	1
1	0	1	0	2	1	3
1	1	0	0	1	3	2
1	1	1	0	3	3	0

Shoup Ex. 6.31; $k = 5$, $n = 3$

$h_{x,y}(a) = ((ax + y) \pmod 5) \pmod 3$

		a	0	1	2	3	4
x	y						
1	0		0	1	2	0	1
1	1		1	2	0	1	0
1	2		2	0	1	0	1
1	3		0	1	0	1	2
1	4		1	0	1	2	0
2	0		0	2	1	1	0
2	1		1	0	0	2	1
2	2		2	1	1	0	0
2	3		0	0	2	1	1
2	4		1	1	0	0	2
3	0		0	0	1	1	2
3	1		1	1	2	0	0
3	2		2	0	0	1	1
3	3		0	1	1	2	0
3	4		1	2	0	0	1
4	0		0	1	0	2	1
4	1		1	0	1	0	2
4	2		2	1	0	1	0
4	3		0	2	1	0	1
4	4		1	0	2	1	0

Table 7. Universal Hash function examples

Toeplitz example

$$\begin{bmatrix} r_{1\,1} & r_{1\,2} & r_{1\,3} & r_{1\,4} \\ r_{2\,1} & r_{1\,1} & r_{1\,2} & r_{1\,3} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Output is $2b_1 + b_2$

					a_1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
					a_2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
					a_3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
					a_4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$r_{2\,1}$	$r_{1\,1}$	$r_{1\,2}$	$r_{1\,3}$	$r_{1\,4}$																	
0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1		0	2	0	2	0	2	0	2	0	2	0	2	0	2	0	2
0	0	0	1	0		0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
0	0	0	1	1		0	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1
0	0	1	0	0		0	0	1	1	2	2	3	3	0	0	1	1	2	2	3	3
0	0	1	0	1		0	2	1	3	2	0	3	1	0	2	1	3	2	0	3	1
0	0	1	1	0		0	1	3	2	2	3	1	0	0	1	3	2	2	3	1	0
0	0	1	1	1		0	3	3	0	2	1	1	2	0	3	3	0	2	1	1	2
0	1	0	0	0		0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3
0	1	0	0	1		0	2	0	2	1	3	1	3	2	0	2	0	3	1	3	1
0	1	0	1	0		0	1	2	3	1	0	3	2	2	3	0	1	3	2	1	0
0	1	0	1	1		0	3	2	1	1	2	3	0	2	1	0	3	3	0	1	2
0	1	1	0	0		0	0	1	1	3	3	2	2	2	2	3	3	1	1	0	0
0	1	1	0	1		0	2	1	3	3	1	2	0	2	0	3	1	1	3	0	2
0	1	1	1	0		0	1	3	2	3	2	0	1	2	3	1	0	1	0	2	3
0	1	1	1	1		0	3	3	0	3	0	0	3	2	1	1	2	1	2	2	1
1	0	0	0	0		0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
1	0	0	0	1		0	2	0	2	0	2	0	2	1	3	1	3	1	3	1	3
1	0	0	1	0		0	1	2	3	0	1	2	3	1	0	3	2	1	0	3	2
1	0	0	1	1		0	3	2	1	0	3	2	1	1	2	3	0	1	2	3	0
1	0	1	0	0		0	0	1	1	2	2	3	3	1	1	0	0	3	3	2	2
1	0	1	0	1		0	2	1	3	2	0	3	1	1	3	0	2	3	1	2	0
1	0	1	1	0		0	1	3	2	2	3	1	0	1	0	2	3	3	2	0	1
1	0	1	1	1		0	3	3	0	2	1	1	2	1	2	2	1	3	0	0	3
1	1	0	0	0		0	0	0	0	1	1	1	1	3	3	3	3	2	2	2	2
1	1	0	0	1		0	2	0	2	1	3	1	3	3	1	3	1	2	0	2	0
1	1	0	1	0		0	1	2	3	1	0	3	2	3	2	1	0	2	3	0	1
1	1	0	1	1		0	3	2	1	1	2	3	0	3	0	1	2	2	1	0	3
1	1	1	0	0		0	0	1	1	3	3	2	2	3	3	2	2	0	0	1	1
1	1	1	0	1		0	2	1	3	3	1	2	0	3	1	2	0	0	2	1	3
1	1	1	1	0		0	1	3	2	3	2	0	1	3	2	0	1	0	1	3	2
1	1	1	1	1		0	3	3	0	3	0	0	3	3	0	0	3	0	3	3	0

Table 8. Universal Hash function example constructed using a Toeplitz matrix