HMAC is a Randomness Extractor and Applications to TLS

Pierre-Alain Fouque, David Pointcheval, Sébastien Zimmer

This extended abstract appeared in
Proceedings of the 2008 ACM Symposium on Information, computer and communications security (AsiaCCS '08)
(March 18–20, 2008, Tokyo, Japan), pages 21–32, ACM Press, New York, NY, USA.

HMAC is a Randomness Extractor and Applications to TLS

Pierre-Alain Fouque[1], David Pointcheval[2], and Sébastien Zimmer[3]

[1] ENS – CNRS – INRIA, Paris, France – fouque@di.ens.fr
[2] CNRS – ENS – INRIA, Paris, France – pointcheval@di.ens.fr
[3] ENS – CNRS – INRIA, Paris, France – zimmer@di.ens.fr

Abstract. In this paper, we study the security of a practical randomness extractor and its application in the TLS standard. Randomness extraction is the first stage of key derivation functions since the secret shared between the entities does not always come from a uniformly distributed source. More precisely, we wonder if the HMAC function, used in many standards, can be considered as a randomness extractor? We show that when the shared secret is put in the key space of the HMAC function, there are two cases to consider depending on whether the key is larger than the block-length of the hash function or not. In both cases, we provide a formal proof that the output is pseudo-random, but under different assumptions. Nevertheless, all the assumptions are related to the fact that the compression function of the underlying hash function behaves like a pseudo-random function. This analysis allows us to prove the TLS randomness extractor for Diffie-Hellman and RSA key exchange. Of independent interest, we study a computational analog to the leftover hash lemma for computational almost universal hash function families: any pseudo-random function family matches the latter definition.

1 Introduction

Randomness extraction is the first stage of key derivation mechanisms. After the key exchange protocol, entities share a secret element of a distribution, called *source* in the sequel, but the entropy of this source is not maximal in general. This means that it is not a uniformly distributed bit string. For example, the Decisional Diffie-Hellman assumption guarantees that a Diffie-Hellman element is a uniformly distributed *element in the group* but its binary representation is not a uniformly distributed *bit string in* $\{0,1\}^n$ (where n is the bit-size of the element). Consequently, the secret element cannot be just plugged as a secret key in a symmetric scheme. To transform this high entropy source into a bit string with maximal entropy, or at least *indistinguishable* from a maximal entropy bit string, randomness extractors come to play. This transformation *condenses* the entropy source by generating a bit string smaller than the input source. Even if they are not designed toward this security goal, many standards use hash functions or MACs (see for example [9, 10, 18, 19]) also for this task since they are already implemented in cryptographic products and so do not require to implement other functions. The reason why they have been considered for this is that MAC functions are usually thought as being good pseudo-random functions and that they condense their input. Here, we study the HMAC function as a randomness extractor. The main application we target is the proof of the randomness extractor of the *new* draft-version of TLS standard, namely TLS v.1.2 [11]. In this standard, HMAC is an intermediate function used in the randomness extraction function, and it is not difficult to see that the security of this function as a randomness extractor reduces to the security of HMAC as a randomness extractor. The key generation in the new TLS version 1.2 is not very different from the key generation in the previous TLS version 1.1 however we focus here in the emerging version. There is a small difference in the derivation function used, but the main difference relies on the specific hash functions

in used and some of our results could be adapted but one has to be careful on the output size of the hash functions.

1.1 Related works

There is some well-known extractor, the Leftover Hash Lemma [15], which can be applied on *any* source with high entropy. Such an extractor is particularly interesting since it can be built under standard assumption. The use of this lemma on the Diffie-Hellman source has been proposed by Gennaro *et al.* in [14]. But, for this particular source, there also exists simple and efficient extractor provided the group size is sufficiently large. For instance, in [8], Chevassut *et al.* show that for safe prime numbers, a simple extractor on the group of squares can be done whose output is perfect. Later, Fouque *et al.* in [13], extended this result to large subgroups by simply taking the high or low order bits of the group elements. This result is achieved using characters and exponential sums as proposed by Canetti *et al.* in [6, 7]. Such constructions are very simple but not so efficient as the Leftover Hash Lemma since the subgroup is always very large. Consequently, the key exchange is not very efficient in practice.

In [12], Dodis *et al.* were the first to consider randomness extraction as an important stage of key derivation mechanisms. They study how *classical* cryptographic primitives, such as MACs or hash functions, behave as randomness extractors. More precisely, they reduce the security of randomness extraction to the assumption that the compression function behaves like a good randomness extractor, namely like an almost universal hash function family.

One widely used MAC function is the HMAC function, hash-based Message Authentication Code, proposed by Bellare, Canetti and Krawczyk in [3]. In [2] Bellare shows that HMAC is a pseudo-random function under the whole assumption that the compression function is a pseudo-random function.

Finally, the TLS key exchange has been studied by Jonsson and Kaliski in [17], for the security of RSA Encryption in the *random oracle model*. They prove that the key exchange in TLS is an IND-CCA2 tagged key encapsulation mechanism, with the assistance of a "partial-RSA decision oracle", under the assumptions that both the key extraction and derivation functions are random functions and that RSA is hard to invert. Here we focus on a security proof of the key extraction function in the *standard model*.

1.2 Our Results

In this paper, we study the situation where the common secret is used as the secret key of HMAC. We show that in this case, for any input of HMAC, the output is indistinguishable from a random bit string, namely it is a pseudorandom string. This construction is used in TLS and therefore is of practical interest. More precisely, we give theoretical security results on this construction for HMAC and then reformulate these results for the particular case of TLS. We focus on the practical security of the TLS extraction function when SHA-384 is used and prove that in this case we can obtain a 124-bit security with RSA and Diffie-Hellman key exchange.

The construction we study is different from the one studied in [12]. In [12], there is a proof for HMAC as a randomness extractor but when the source is injected in the message space, with a random but known key. Whereas our construction is used in the TLS key extraction, the latter construction is used in the IPSec standard. In the IPSec construction, the shared key length can be larger than the block length. For

example, in HMAC-SHA-1 the block-size is 512 bits and a shared Diffie-Hellman element is at least 1024-bit long, therefore it is splitted over at least two blocks. Consequently, the hash function must be iterated and the results of [12] require high conditional min-entropy of at least one block. That means that, in our example, the entropy of the most significant bits of the Diffie-Hellman element is high, even when the least significant bits are given. This result can be achieved following result of [13] but it requires a large subgroup. With our technique, we avoid this drawback and, as in the Leftover Hash Lemma, we require only that some entropy is present in the group element. We are always able to extract the entropy diluted in the whole bit string. Therefore, we can use groups with rather small prime order subgroups, which allows much more efficient key exchange protocols.

In this work, we use some computational assumptions, notably the classical assumption in cryptography that the compression function is a pseudo-random function. This assumption has also been done by Bellare *et al.* in [3, 2].

In [2], Bellare introduced the notion of computationally almost universal hash function. We extend this notion and prove a computational analog of the famous Leftover Hash Lemma, which allows to extract entropy easily. Since any pseudo-random function (prf) is also computationally almost universal, therefore a strong key (*i.e.* computationally indistinguishable from a true random bit string) is derived from any good entropy source using a good prf. The only restriction is on the size of the output on the prf: the latter should be smaller than the prf key size, otherwise the advantage of the prf is not small enough to be used with the Leftover Hash Lemma. This means that this result has a practical impact for truncated iterated hash functions, as SHA-384 or HMAC-SHA-384. This justifies, with reasonable computational assumptions, the use of these hash functions in practice to derive keys.

Finally, the HMAC standard imposes that if the key is larger than the block-length, HMAC begins by hashing the secret to reduce it, and then the result is put as the key of HMAC. Therefore, to be complete, there are actually two cases to consider depending on whether the key is larger than the block-length of the hash function or not. If the common secret is larger than the block-length, we show that hashing the secret key allows us to extract entropy whose distribution is indistinguishable from a random bit-string. Then, we use the recent results of Bellare [2] at Crypto 2006, to show that the output of HMAC is pseudo-random. As far as we know, we are the first to study this particular case.

If the shared secret is smaller than the block size, two bit strings are generated and then used to key an intermediate pseudorandom function NMAC. As pointed out by Bellare [2], assuming that these keys are chosen independently is not true for HMAC since they are derived from a *single* bit string. Instead, we show that these strings are computationally indistinguishable from two random bit strings.

Note that the both cases may append in practice: the Diffie-Hellman key exchange over $\mathbb{Z}_p^\star$ (with p a prime) generates a key of at least 1024 bits, which is greater than the 512-bit HMAC-SHA-1 key size, whereas the elliptic curve Diffie-Hellman key exchange generates a key of generally exactly 512 bits.

1.3 Organization of the Paper

In section 2, we give useful notations and security definitions. Then, we give the main security results for HMAC, tearing apart the case when the key is smaller than the block-length and the case when it is longer. Finally, we apply the method presented in

section 3 to give theoretical and practical security results for the TLS key extraction function.

2 Notations and Definitions

NOTATIONS. If X is a random variable taking values in $\mathcal{X}$ and drawn according to the distribution $\mathcal{D}$ then $X \stackrel{\mathcal{D}}{\leftarrow} \mathcal{X}$ denotes the choice of X in $\mathcal{X}$ according to $\mathcal{D}$ and $X \stackrel{\$}{\leftarrow} \mathcal{X}$ denotes the choice of X when X is uniformly distributed in $\mathcal{X}$. The uniform distribution on $\{0,1\}^\kappa$ is denoted by $\mathcal{U}_\kappa$.

When an adversary A can interact with an oracle O and at the end of the interaction outputs b, it is denoted by $A^O \Rightarrow b$. If B and C are two events, the probability that the event B occurs, knowing the event C is denoted by $\Pr[B : C]$. When an adversary is involved in an event, the probability is considered upon the adversary random coins.

MIN-ENTROPY, UNIVERSAL HASH FAMILY AND RANDOMNESS EXTRACTOR.

Let X be a random variable with values in a set $\mathcal{X}$. The *guessing probability* of X, denoted by $\gamma(X)$, is the probability $\max_{x\in\mathcal{X}}(\Pr[X = x])$. The *min entropy* of X denoted $H_\infty(X)$ is equal to $-\log_2(\gamma(X))$.

Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be two distributions on the same set $\mathcal{X}$. The *statistical distance* between $\mathcal{D}_1$ and $\mathcal{D}_2$ is:

$$\mathbf{SD}(\mathcal{D}_1, \mathcal{D}_2) = \frac{1}{2}\sum_{x\in\mathcal{X}} \left| \Pr_{X\stackrel{\mathcal{D}_1}{\leftarrow}\mathcal{X}}[X = x] - \Pr_{X\stackrel{\mathcal{D}_2}{\leftarrow}\mathcal{X}}[X = x] \right|.$$

Let Ext be a function family from $\{0,1\}^d \times \{0,1\}^n$ into $\{0,1\}^\ell$. Let i be a uniform random variable in $\{0,1\}^d$ and U_ℓ denote a random variable uniformly distributed in $\{0,1\}^\ell$. The function family Ext is an (ε, m)-*strong extractor* if for all random variables X over $\{0,1\}^n$ of min entropy at least m, with U_ℓ, i and X independent: $\mathbf{SD}(\langle i, \mathsf{Ext}_i(X)\rangle, \langle i, U_\ell\rangle) < \varepsilon$.

Presumably, the most famous way of extracting entropy is provided by the Leftover Hash Lemma presented in [15, 16]. A variant of this lemma introduced by Dodis *et al.* [12] is presented below.

Let $H : \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^\ell$ be a family of efficiently computable hash functions. The family H is called an ε-almost universal hash (ε-auh) function family if for every $x \neq y$ in $\{0,1\}^n$, $\Pr_{i\in\{0,1\}^d}[H_i(x) = H_i(y)] \leq 1/2^\ell + \varepsilon$.

Theorem 1 (Leftover Hash Lemma). *Let H be an ε-auh function family from $\{0,1\}^d \times \{0,1\}^n$ to $\{0,1\}^\ell$. Let i denote a random variable uniformly distributed in $\{0,1\}^d$, U_ℓ a random variable uniformly distributed in $\{0,1\}^\ell$, and A a random variable taking values in $\{0,1\}^n$, with i, A, U_ℓ mutually independent. Then:*

$$\mathbf{SD}(\langle i, H_i(A)\rangle, \langle i, U_\ell\rangle) \leq \sqrt{2^\ell(2^{-H_\infty(A)} + \varepsilon)}/2.$$

COMPUTATIONAL RANDOMNESS EXTRACTOR. A computational randomness extractor (cre) is an extension of randomness extractor where the output is *computationally* indistinguishable from the uniform variable. This notion has also been implicitly used in [15, 12]. It is a function family cExt from $\{0,1\}^d \times \{0,1\}^n \times \mathcal{D}om$ to $\{0,1\}^\ell$ that satisfies the following game. At the beginning, the challenger chooses uniformly at

random a bit b and a random d-bit string i. According to the value of b, he assigns ext to a random function taken in $\mathcal{F}_{(n,\ell)}$, the set of all functions from $\{0,1\}^n$ to $\{0,1\}^\ell$, or to the randomness extractor cExt_i. Then the adversary sends to the challenger an efficiently samplable probability distribution $\mathcal{D}$ over $\{0,1\}^n$ whose min-entropy is greater than m and possibly a label $label \in \mathcal{D}om$. The challenger chooses x according to the distribution $\mathcal{D}$ and sends to the adversary $(i, \mathsf{ext}(x, label))$. Finally, the adversary outputs a random bit b' for her guess of b. Her advantage, denoted $\mathsf{adv}^{\mathsf{cre}}_{\mathsf{cExt}}(A)$, is:

$$\left|\Pr[A^{\mathsf{ext}} \Rightarrow 1 \colon \mathsf{ext} \xleftarrow{\$} \mathsf{cExt}] - \Pr[A^{\mathsf{ext}} \Rightarrow 1 \colon \mathsf{ext} \xleftarrow{\$} \mathcal{F}_{(n,\ell)}]\right|$$

This notion directly implies the semantic security, against a passive adversary, of a key generated with a computational randomness extractor from a high-entropy random source. Indeed, if an adversary is able to attack the semantic security of the key, then it is able to distinguish this computational randomness extractor from a perfectly random function. Therefore, if a good computational randomness extractor is used to generate the key, the semantic security of the key is guaranteed. If authentication techniques are used, the key exchange security against an active adversary reduces to the security against a passive adversary and therefore the semantic security of the key is guaranteed even against an active adversary.

Computational Almost Universality. Let $F : \mathcal{K}eys \times \mathcal{D}om \to \mathcal{R}ng$ be a function family. We generalize here the definition of [2]. The goal of a m-au adversary A is to generate an efficiently samplable distribution $\mathcal{D}$ over $\mathcal{D}om^2$ with min-entropy at least m such that, for a random key K and a random couple (M_1, M_2) chosen according to $\mathcal{D}$ in $\mathcal{D}om^2$, $F_K(M_1)$ and $F_K(M_2)$ collision with high probability. Her m-au-advantage, denoted $\mathsf{adv}^{\mathtt{m-au}}_F(A)$, is:

$$\Pr\left[\begin{array}{c} F(K,M_1) = F(K,M_2) \\ M_1 \neq M_2 \end{array} : \begin{array}{c} A \Rightarrow \mathcal{D};\ K \xleftarrow{\$} \mathcal{K}eys \\ (M_1,M_2) \xleftarrow{\mathcal{D}} \mathcal{D}om^2 \end{array}\right].$$

Note that Bellare's definition is the particular case when m, the min-entropy of $\mathcal{D}$, equals 0. When $m \geq 1$, this is a weaker notion than the original one, because every m-au adversary can be turned into a 0-au adversary with the same running-time and the same advantage (the 0-au adversary runs the m-au adversary, chooses (M_1, M_2) according to $\mathcal{D}$ and sends it to the challenger).

Pseudo-Random Function. Let $F : \mathcal{K}eys \times \mathcal{D}om \to \mathcal{R}ng$ be a function family. We denote by $\mathcal{F} = \mathcal{F}_{(\mathcal{D}om,\mathcal{R}ng)}$ all the functions from $\mathcal{D}om$ to $\mathcal{R}ng$. The goal of a prf-adversary A, which runs in time T, against F is to guess the value of b in the following game. The challenger chooses a bit b at random; if $b = 1$ he assigns f to a random function from $\mathcal{F}$ otherwise he chooses a random key K in $\mathcal{K}eys$ and assigns f to $F(K, \cdot)$. The adversary can interact with f making up to q queries x_i and receives $f(x_i)$. The prf-advantage of A, denoted $\mathsf{adv}^{\mathtt{prf}}_F(A)$, is:

$$\left|\Pr\left[A^{F(K,\cdot)} \Rightarrow 1 \colon K \xleftarrow{\$} \mathcal{K}eys\right] - \Pr\left[A^{f} \Rightarrow 1 \colon f \leftarrow \mathcal{F}\right]\right|.$$

Prefix-freeness. Let $\mathcal{S}$ be a set of bit strings and let x and x' be a couple of bit string from $\mathcal{S}$, we denote by $x \subset x'$ the fact that x is a prefix of x'. A distribution $\mathcal{D}$ over $\mathcal{S}$ is prefix-free if for all couple $(x, x') \in \mathcal{S}^2$ such that $\Pr\left[(X, X') = (x, x') \colon X \xleftarrow{\mathcal{D}} \mathcal{S}, X' \xleftarrow{\mathcal{D}} \mathcal{S}\right] > 0$, $x \subset x'$ implies $x = x'$. The set $\mathcal{S}$ is prefix-free if for all couples $(x, x') \in \mathcal{S}^2$, $x \subset x'$ implies that $x = x'$. An adversary is said prefix-free if the set of its queries form a prefix-free set and if it outputs only prefix-free distributions.

3 Hmac Security as a Key Derivation Function

3.1 Description of Hmac

The cascade construction The cascade construction is the construction used for iterated hash functions. We denote by $H : \{0,1\}^\kappa \times \{0,1\}^* \to \{0,1\}^\kappa$ such a hash function and by $h : \{0,1\}^\kappa \times \{0,1\}^b \to \{0,1\}^\kappa$ the so-called compression function. The cascade construction of h is the function $h^* : \{0,1\}^\kappa \times \left(\{0,1\}^b\right)^* \to \{0,1\}^\kappa$, 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 $a \in \{0,1\}^\kappa$. To construct H, a way to pad messages to an exact multiple of b bits needs to be defined. In practice this padding is a function of the length of the input x, $|x|$. We denote by $\mathrm{pad}(|x|)$ the function induced by the padding and by $x_{pad} = x \| \mathrm{pad}(|x|)$. The function H is defined by $H(a,x) = h^*(a, x_{pad})$.

Let $1 \leq t \leq \kappa$ be an integer. In the following, for any function with range $\{0,1\}^\kappa$, we denote $\overline{F}$ the function F for which the $\kappa - t$ least significant bits of the output are truncated, that is if $\mathsf{msb}_t(\cdot)$ denote the t most significant bits of a bit string, for every input x, $\overline{F}(x) = \mathsf{msb}_t(F(x))$. We mostly use this notation for $\overline{h^*}$ and $\overline{H}$ (the reader may think about SHA-384 which is a truncated iterated hash function for which $t = 384$ and $\kappa = 512$).

Nmac Nmac is a hash function family from $\{0,1\}^\kappa \times \{0,1\}^\kappa \times \{0,1\}^*$ to $\{0,1\}^c$. It is constructed from a (possibly truncated) iterated hash function Hash from $\{0,1\}^\kappa \times \{0,1\}^*$ to $\{0,1\}^c$. If $(k_1, k_2) \in (\{0,1\}^\kappa)^2$ is a couple of keys and $x \in \{0,1\}^*$ the input, the definition of Nmac is:

$$\mathsf{Nmac}^{\mathsf{Hash}}(k_1, k_2, x) = \mathsf{Hash}\left(k_2, \mathsf{Hash}(k_1, x)\right). \tag{1}$$

The hash function family Hash can be either a classical or a truncated iterated hash function family, that is $\mathsf{Hash} = H$ and $c = \kappa$ or $\mathsf{Hash} = \overline{H}$ and $c = t$. In these cases equation (1) becomes:

$$\begin{aligned}
\mathsf{Nmac}^{H}(k_1, k_2, x) &= h^*\left(k_2, h^*\left(k_1, x_{pad}\right)_{pad}\right) \\
&= h\left(k_2, h^*(k_1, x_{pad})_{pad}\right), \\
\mathsf{Nmac}^{\overline{H}}(k_1, k_2, x) &= \overline{h^*}\left(k_2, \overline{h^*}\left(k_1, x_{pad}\right)_{pad}\right) \\
&= \overline{h}\left(k_2, \overline{h^*}(k_1, x_{pad})_{pad}\right).
\end{aligned}$$

From now on, we assume that the padded message obtained from any κ-bit query is never larger then b bits (it is the case in practice). This explains the last equality of the equations above.

Hmac Hmac is a hash function from $\{0,1\}^* \times \{0,1\}^*$ to $\{0,1\}^\kappa$. Let *ipad* and *opad* be two b-bit strings and IV be a κ-bit string. If the key k is a bit string from $\{0,1\}^b$, then $\mathsf{Hmac}^{\mathsf{Hash}}_{IV}(ipad, opad; k, x)$ is equal to:

$$\mathsf{Hash}\Big(IV, \big[k \oplus opad\big] \big\| \mathsf{Hash}\big(IV, [k \oplus ipad] \| x\big)\Big). \tag{2}$$

The bit strings *opad*, *ipad* and IV are constants defined in the HMAC standard [5], but in the following we assume that *ipad* and *opad* could vary and are chosen uniformly at random. In practice, these variables were chosen at random once for all when the standard was defined. The consequences of this assumption in practice are discussed in details in subsection 4.2. In the following we denote as index of HMAC the fixed value IV and we put between the brackets variables *ipad* and *opad* which are chosen randomly.

If k is a b-bit string, in the cases when $\mathsf{Hash} = H$ and $\mathsf{Hash} = \overline{H}$, definition (2) can be restated using NMAC, and then $\mathsf{Hmac}^{H}_{IV}(ipad, opad; k, x)$ and $\mathsf{Hmac}^{\overline{H}}_{IV}(ipad, opad; k, x)$ are respectively equal to:

$$\mathsf{Nmac}^{H}\left(h(IV, k \oplus ipad), h(IV, k \oplus opad), x\right),$$
$$\mathsf{Nmac}^{\overline{H}}\left(h(IV, k \oplus ipad), h(IV, k \oplus opad), x\right).$$

Note that these equations are not exactly true because the padding is not exactly the same in HMAC and in NMAC: in HMAC one block key is concatenated to the message and this changes the length of the hash function input and then changes the associated padding. However, to simplify the notations, we can omit this particularity since it does not alterate the results.

If k is not a b-bit string, then it is first transformed into a b-bit string. If k is smaller than b bits, then it is first padded with as many '0' as needed to obtain a b-bit string ; the resulting bit string is used as a key, as defined in (2). If k is longer than b bits, as we explain in the introduction, the HMAC standard [5] imposes to first hash k using Hash to obtain a c key digest ; since $c \leq b$ in practice, the key digest is padded with $b - c$ '0' and the resulting b-bit string is used as a key, as defined in (2).

The key extraction construction In this paper we study the following construction used for key derivation: let *pmk* denote a high entropy s-bit string called the premaster-secret, *label* some bit string possibly adversarily generated, *opad* the fixed bit string as described in the HMAC standard [5] and mk the master-key generated by this construction. The variables *ipad* and *opad* are chosen uniformly at random and mk is computed as follows:

$$mk = \mathsf{Hmac}^{\mathsf{Hash}}_{IV}(ipad, opad; pmk, label).$$

We show that this construction is a good computational randomness extractor that is that the triplet $(ipad, opad; mk)$ is indistinguishable from a random bit string. As the definition of HMAC depends on the size of the *pmk*, our proof is also *pmk* dependent: the proof method is not exactly the same if *pmk* is smaller that the block size or if it is longer.

3.2 When the Shared Key is Smaller Than the Block Length

The study of this case is motivated by the use in practice of elliptic curve Diffie-Hellman key exchange. The premaster-secret *pmk* generated is then presumably 512-bit long, and is smaller than the block-length. We directly show that HMAC is a randomness extractor when it is used with H and $\overline{H}$.

Hmac with H Firstly we show that, for a key smaller than the block size, HMAC is a good randomness extractor when it is used with H. For the proof see the appendix A. We underline that in this theorem we assume that *ipad* and *opad* are chosen uniformly at random, that $h(k,\cdot)$ is a prf when k is the key, and that $h(IV,\cdot\oplus k)$ is a prf when k is the key.

Theorem 2. *Let IV be a fixed κ-bit string and let h be a function family from $\{0,1\}^\kappa \times \{0,1\}^b$ to $\{0,1\}^\kappa$, where the key is the first input on κ bits. Let h' be the hash function defined by $h'_{IV}(pad,\cdot) = h(IV,\cdot\oplus pad)$ where the key is pad. Let A be a cre-adversary against the construction that has time-complexity at most T, generates labels of at most ℓ blocks and a key of at most 1 block and min-entropy m. Then there exist one prf-adversary A_1 against h' and two prf-adversaries A_2 and A_3 against h such that $\mathsf{adv}^{\mathtt{cre}}_{Hmac^H}(A)$ is upper bounded by:*

$$\frac{\sqrt{2^{2\kappa}\left(2^{-m}+2\cdot\mathsf{adv}^{\mathtt{prf}}_{h'}(A_1)\right)}}{2}+\frac{1}{2^\kappa} + \mathsf{adv}^{\mathtt{prf}}_{h}(A_2)+2\ell\cdot\mathsf{adv}^{\mathtt{prf}}_{h}(A_3)$$

where A_1 makes two queries with time-complexity $T+2T_h$, A_2 makes one query with time-complexity T and A_3 makes at most 2 queries with time-complexity $\mathcal{O}(\ell\cdot T_h)$, where T_h is the time for one computation of h.

The main ideas of the proof is to show that the two bitstrings $k_1 = h(IV, ipad\oplus k)$ and $k_2 = h(IV, opad\oplus k)$ are pseudorandom and independent and then to use them to key NMAC as a prf. Firstly, contrary to [4] where it is assumed that k_1 and k_2 are computationally independent, we prove it using the following hash function family:

$$F = (h(IV,\cdot\oplus ipad)\|h(IV,\cdot\oplus opad))_{(ipad,opad)}$$

which is a prf and therefore it is cau. More precisely, there exists a prf-adversary A_1 against h' such that the advantage of the cau-adversary against F is upper bounded by $2\mathsf{adv}^{\mathtt{prf}}_{h'}(A_1)+1/2^{2\kappa}$. Then we can apply a computational variant of the Leftover Hash Lemma to F to extract the entropy of the key and thus show that the output is indistinguishable from a random bit string.

The computational Leftover Hash Lemma is the following:

Lemma 3 (computational LHL). *Let H be a family of functions from $\{0,1\}^k \times \mathcal{D}om$ to $\{0,1\}^t$ such that for every au-adversary B, running in time T and producing a distribution over $\mathcal{D}om\times\mathcal{D}om$ of min-entropy at least $2m$, $\mathsf{adv}^{\mathtt{cau}}_H(B)\le 1/2^t+\varepsilon$. Then for every adversary A running in time $\mathcal{O}(T)$ producing a distribution of min-entropy at least m:*

$$\mathsf{adv}^{\mathtt{cre}}_H(A)\le\sqrt{2^t\cdot(2^{-m}+\varepsilon)}.$$

The proof of this lemma is in appendix C. Note that if ε were greater than $2^{-2\kappa}$, we would have $2^{2\kappa}\cdot\varepsilon\ge 1$ and the upper bound would be meaningless. We need that $\varepsilon \ll 2^{-2\kappa}$, that is why we make *ipad* and *opad* vary and not only IV as we would have preferred to have one assumption on h. Indeed, making the IV vary is equivalent to consider h as a prf when the key is IV. Yet, the exhaustive search prf-adversary against h has a prf-advantage which is equal to $\mathcal{O}(2^{-\kappa})$. It means the better prf-adversary against h has an advantage better than $\mathcal{O}(2^{-\kappa})$, where κ is the key size.

Therefore, assuming that h is a prf is not enough, whereas, since *ipad* and *opad* are large, h' security level may be sufficient.

In the previous step of the proof, we have generated with F, two (concatenated) *computationally pseudorandom and independent* κ-bit strings which can be used to key NMAC. Thus, we can use the fact that NMAC is a prf. When NMAC is used with a classical iterated hash function, this fact was proved by Bellare [2]:

Lemma 4. *Let $h\colon \{0,1\}^\kappa \times \{0,1\}^b \to \{0,1\}^\kappa$ be a family of functions. Let A_{Nmac^H} be a prf-adversary against Nmac^H that makes at most q oracle queries, each of at most ℓ blocks, and has time-complexity T. Then there exist prf-adversaries A_1 and A_2 against h such that $\mathsf{adv}^{\mathtt{prf}}_{\mathsf{Nmac}^H}\left(A_{\mathsf{Nmac}^H}\right)$ is upper bounded by:*

$$\mathsf{adv}^{\mathtt{prf}}_h\left(A_1\right) + \binom{q}{2}\left[2\ell \cdot \mathsf{adv}^{\mathtt{prf}}_h\left(A_2\right) + \frac{1}{2^\kappa}\right].$$

Furthermore, A_1 has time-complexity at most T and makes at most q oracle queries while A_2 has time-complexity at most $\mathcal{O}(\ell \cdot T_h)$ and makes at most 2 oracle queries, where T_h is the time for one computation of h.

Hmac with $\overline{H}$ Secondly we show that, for a key smaller than the block size, HMAC used with $\overline{H}$ is a good randomness extractor.

Theorem 5. *Let IV be a fixed κ-bit string and let h be a function family from $\{0,1\}^\kappa \times \{0,1\}^b$ to $\{0,1\}^\kappa$, where the key is the first input on κ bits. Let h' be the hash function defined by $h'_{IV}(pad,\cdot) = h(IV, \cdot \oplus pad)$ where the key is pad. Let A be a cre-adversary against the construction that has time-complexity at most T, generates labels of at most ℓ blocks and a key of at most 1 block and min-entropy m. Then there exist one prf-adversary A_1 against h' and two prf-adversaries A_2 and A_3 against h such that $\mathsf{adv}^{\mathtt{cre}}_{\mathsf{Hmac}^H}\left(A\right)$ is upper bounded by:*

$$\frac{\sqrt{2^{2\kappa}\left(2^{-m} + 2 \cdot \mathsf{adv}^{\mathtt{prf}}_{h'}\left(A_1\right)\right)}}{2} + \frac{1}{2^t} + \mathsf{adv}^{\mathtt{prf}}_h\left(A_2\right) + 2\ell \cdot \mathsf{adv}^{\mathtt{prf}}_h\left(A_3\right)$$

where A_1 makes two queries with time-complexity $T + 2T_h$, A_2 makes one query with time-complexity T and A_3 makes at most 2 queries with time-complexity $\mathcal{O}(\ell \cdot T_h)$, where T_h is the time for one computation of h.

The proof of this theorem, which can be found in appendix A, is similar to the proof of theorem 2, but Bellare's result cannot be applied directly to NMAC used with $\overline{H}$: the output of $\overline{H}$ is much smaller than the output of H and due to it, his proof has to be adapted. We obtain the following result:

Lemma 6. *Let $h\colon \{0,1\}^\kappa \times \{0,1\}^b \to \{0,1\}^\kappa$ be a family of functions. Let $A_{\mathsf{Nmac}^{\overline{H}}}$ be a pf-prf-adversary against $\mathsf{Nmac}^{\overline{H}}$ that makes at most q oracle queries, each of at most ℓ blocks, and has time-complexity T. Then there exist prf-adversaries A_1 and A_2 against h such that the advantage $\mathsf{adv}^{\mathtt{pf-prf}}_{\mathsf{Nmac}^{\overline{H}}}\left(A_{\mathsf{Nmac}^{\overline{H}}}\right)$ is upper bounded by:*

$$\mathsf{adv}^{\mathtt{prf}}_h\left(A_1\right) + \binom{q}{2}\left[2\ell \cdot \mathsf{adv}^{\mathtt{prf}}_h\left(A_2\right) + \frac{1}{2^t}\right].$$

Furthermore, A_1 has time-complexity at most T and makes at most q oracle queries while A_2 has time-complexity at most $\mathcal{O}(\ell T_h)$ and makes at most 2 oracle queries, where T_h is the time for one computation of h.

This lemma can be established with a proof similar to the one for HMAC with H, that can be found in [2], except that the tests are made upon the t most significant bits of the output of H and that the adversary is constrained to output prefix-free messages.

3.3 When the Shared Key is larger than the block length

As explain in section 3.1, if the key is larger than the block size, then it is first hashed and padded with '0' bits to obtain a b-bit string. This case is rarely studied in HMAC security analysis and requires that we study what is the impact of the key hashing. However it is of practical interest since the Diffie-Hellman key exchange over $\mathbb{Z}_p^\star$ (where p is a prime) generates a premaster-secret of at least 1024 bits, which is greater than the 512-bit HMAC-SHA-1 key size. In this section, we focus on HMAC used with a truncated hash function $\overline{H}$. We first give the security results and then give the intermediate lemmas used in the proof, in particular we study the cascade mode as a randomness extractor.

Results for Hmac The hashing of the premaster-secret has two main consequences on our proof. The output is a t-bit string and as a first consequence we have to show that a lot of the entropy of the input is preserved: if the output had low entropy, an exhaustive search could allow to guess the few possible values of the key. We are more precise and show that the output of the hashing is indistinguishable from the uniform when HMAC is used with $\overline{H}$.

The other consequence of the hashing is that HMAC is keyed with a key with the $b-t$ least significant bits equal to '0'. We have to show that even in these circumstances it is still a good prf, which guarantees that the output of HMAC is indistinguishable from the uniform. To this end, we consider the related key attacks against h when the input and the output are reversed.

From the function family $h\colon \{0,1\}^\kappa \times \{0,1\}^b \to \{0,1\}^\kappa$ we define the family of functions $\widehat{h}\colon \{0,1\}^t \times \{0,1\}^\kappa \to \{0,1\}^\kappa$ defined by $\widehat{h}(x,y) = h(y, x\|0^{b-t})$.

A related-key attack on a family of functions $\widehat{h}\colon \{0,1\}^t \times \{0,1\}^\kappa \to \{0,1\}^\kappa$ is parametrized by a set $\Phi \subset \mathcal{F}_{(t,t)}$ of key-derivation functions (where $\mathcal{F}_{(t,t)}$ is the set of all functions from $\{0,1\}^t$ to $\{0,1\}^t$). In the rka game, a challenger chooses a random bit b and a random key K. If $b = 1$ it chooses a random function G from the set of all the functions from $\{0,1\}^t \times \{0,1\}^\kappa$ to $\{0,1\}^\kappa$ and uses $G(K,\cdot)$. If $b = 0$, it uses $\widehat{h}(K,\cdot)$. The goal of the rka-adversary is to guess the value of b. She may make an oracle query of the form ϕ, x where $\phi \in \Phi$ and $x \in \{0,1\}^\kappa$ and the oracle returns $G(\phi(K), x)$ if $b = 1$ and $\widehat{h}(\phi(K), x)$ otherwise. Her rka-advantage is defined by:

$$\mathsf{adv}^{\mathtt{rka}}_{\widehat{h},\Phi}(A) = \Pr\left[A^{\widehat{h}} \Rightarrow 1\right] - \Pr\left[A^{G} \Rightarrow 1\right].$$

For any string $str \in \{0,1\}^t$ let $\Delta_{str}\colon \{0,1\}^t \to \{0,1\}^t$ be defined by $\Delta_{str}(K) = K \oplus str$.

Theorem 7. *Let h be a function family from $\{0,1\}^\kappa \times \{0,1\}^b$ to $\{0,1\}^\kappa$. Let ipad and opad be two b-bit strings and let $\Phi = \{\Delta_{ipad}, \Delta_{opad}\}$. Let A be a pf-cre-adversary against the construction that has time-complexity at most t, generate labels of at most*

ℓ blocks and a key of $s \geq 2$ blocks and min-entropy m. Then there exist a rka-adversary A_2 against $\widehat{h}$ and three prf-adversaries A_1, A_3, A_4 such that $\mathsf{adv}^{\mathtt{pf-cre}}_{\mathit{Hmac}^{\overline{H}}}(A)$ is upper bounded by:

$$\sqrt{2^t\left(3\cdot 2^{-m}+2s\cdot \mathsf{adv}^{\mathtt{prf}}_h(A_1)\right)}+\mathsf{adv}^{\mathtt{rka}}_{\widehat{h}}(A_2)$$
$$+\mathsf{adv}^{\mathtt{prf}}_h(A_3)+2\ell\cdot\mathsf{adv}^{\mathtt{prf}}_h(A_4)+\tfrac{1}{2^t}$$

where A_1 and A_2 make at most 2 queries and have time-complexity t, A_3 makes one query with time-complexity t and A_4 makes at most 2 queries with time-complexity $\mathcal{O}(\ell\cdot T_h)$.

To show this theorem, we first apply a prefix-free computational variant of the Leftover Hash Lemma to the cascade construction. This result is stated in lemma 8 below. This way we show that the output of the hashing is a random looking t-bit string. Since h, where input and key are reversed and where the key is restricted to the t first bits, is a prf resistant to rka, and since the output k of the hashing is random looking, the output of $h(IV, ipad\oplus k\|0^{b-t})\|h(IV, opad\oplus k\|0^{b-t})$ is indistinguishable from the uniform. Therefore we key with two random looking bit strings and since NMAC is a prf, its output seems to be uniformly distributed. All the precise proofs of the results of this section are in appendix B.

Note that in this proof we assume that IV is chosen at random at the beginning of the game. On the other hand, we do not use the fact that *ipad* and *opad* are chosen at random at the beginning of the game. This assumption is indeed not useful in this particular context.

The cascade mode is a good pf-cre In this section we show that the cascade mode is a good extractor of entropy, if the compression function is a prf. The main result of this part is the following lemma, used in the proof of theorem 7:

Lemma 8. *Let A be a pf-cre-adversary against $\overline{h^*}$ which has a time-complexity at most T and produces a distribution of min-entropy at least m, with messages of at most ℓ blocks. Then there is a prf-adversary A' with running-time at most $\mathcal{O}(T)$ and messages at most ℓ-block long such that:*

$$\mathsf{adv}^{\mathtt{pf\text{-}cre}}_{\overline{h^*}}(A)\leq\sqrt{2^t\cdot(3\cdot 2^{-m}+2\ell\cdot\mathsf{adv}^{\mathtt{prf}}_h(A'))}.$$

This lemma is a direct consequence of the two lemmas 9 and 10 below.

Lemma 9. *Let A^* be a pf-au-adversary against h^* which generates messages of at most ℓ blocks. Then there is a prf-adversary A against h such that:*

$$\mathsf{adv}^{\mathtt{pf\text{-}au}}_{\overline{h^*}}(A^*)\leq 2\ell\cdot\mathsf{adv}^{\mathtt{prf}}_h(A)+\frac{1}{2^t}$$

and A makes at most 2 queries and has about the same time-complexity as A^*.

Lemma 10 (pf computational LHL). *Let H be a family of functions from $\{0,1\}^k\times\mathcal{D}om$ to $\{0,1\}^t$ such that for every au-adversary B, running in time T and producing a distribution over $\mathcal{D}om\times\mathcal{D}om$ of min-entropy at least $2m-2$, $\mathsf{adv}^{\mathtt{pf-cau}}_H(B)\leq 1/2^t+\varepsilon$. Then for every adversary A running in time $\mathcal{O}(T)$ producing a distribution of min-entropy at least m:*

$$\mathsf{adv}^{\mathtt{pf\text{-}cre}}_H(A)\leq\sqrt{2^t\cdot(3\cdot 2^{-m}+\varepsilon)}$$

Remark that $\varepsilon \geq 2^{-\kappa}$ that is why the output of the hash function has to be smaller than the key, that is $t < \kappa$. Indeed, consider the following prf-adversary with running time T and which makes two queries: she chooses at random $(x_1, x_2) \in \{0,1\}^b$, sends it to the challenger which returns (y_1, y_2). Then she chooses $\overline{T} = T/T_f$ keys K and tests for all key if $h(K, x_1)$ is equal to y_1. If it is the case, she checks if $h(K, x_2)$ equals y_2 and if it is also the case, then she returns 1, else she returns 0 at the end. Her prf-advantage is greater than $2^{-\kappa}$, therefore $\varepsilon \geq 2^{-\kappa}$. This adversary is called the exhaustive search adversary.

4 Applications to the Key Derivation Function of TLS

In this section we apply the methods and the results of previous section to the new draft-version of TLS v.1.2 [11]. We give security proofs for the key-extraction function described in the standard, function which is very similar to the one used in previous versions of TLS.

Besides, the new TLS standard promotes the use of at least SHA-256 or a stronger standard hash function. In this paper, we focus on SHA-384 and give security results addressing the specific case of a truncated iterated hash function.

4.1 Brief Description of TLS Key Extraction Function

In TLS the key extraction is performed the following way. Firstly the client and the server exchange two random 256-bit strings $rand_s$ and $rand_c$ with 224 random bits in each. Then the client and the server exchange a premaster key.

In the RSA key exchange, the client generates a 368-bit random string, concatenates it to the latest version of the protocol supported, encoded on 16 bits, and encrypts them under the server's RSA public key. The latter 384-bit string is the premaster-secret. It is a 384-bit value, but there are only 368 random bits of entropy (the 16 most significant bits are fixed).

In the Diffie-Hellman key exchange, the client and the server use a group G, in which the DDH assumption holds, and then perform a DH protocol to obtain a common random element of G. The binary representation of this element is the premaster-secret. Note that this binary representation is not a uniformly distributed bit string.

In both cases, we denote by pmk the premaster-secret. Then, the so-called master-secret, denoted by mk is created. During the first computation, the parties extract the entropy of pmk using a function called HPRF.

The function HPRF can be any function specified by the cipher-suite in used, but in this paper we focus on the function proposed in the standard, function which is very similar to the one used in the previous version of the protocol, TLS 1.1. This function is constructed from several concatenations and iterations of HMAC. For sake of simplicity, we do not describe precisely this function here, for more details, see [11]. The same way HMAC is derived from NMAC, function HPRF can be seen as derived from a function that we call NPRF, that is $\mathsf{Hprf}^{\mathsf{Hash}}_{ipad,opad}(IV; k, x)$ is equal to:

$$\mathsf{Nprf}^{\mathsf{Hash}}\left(h(IV, k \oplus ipad), h(IV, k \oplus opad), x\right).$$

The same way we have shown that HMAC is a good computational randomness extractor since it is the composite of NMAC, which is a prf, with a computational randomness extractor, we show here that HPRF is a good computational randomness extractor since is the composite of NPRF, which is a prf, with a computational randomness

extractor. Note that contrarily to HMAC, we only have to choose randomly IV, *ipad* is fixed.

As NPRF is a concatenation and a composite of several NMAC, the prf-resistance of NPRF can be reduced to the prf-resistance of NMAC. The number v of concatenations depends on the output length required by the cipher suite and the prf-security of this number.

Theorem 11. *Let $u \geq 1$, $t \geq 1$ and let $h : \{0,1\}^\kappa \times \{0,1\}^b \to \{0,1\}^\kappa$ be a family of functions. Let A be a prf-adversary against $\mathsf{Nprf}^{\overline{H}}$ constructed with v concatenations of $\mathsf{Nmac}^{\overline{H}}$. The algorithm A can make at most q queries, each of at most u blocks, and has time-complexity at most T. Then there exist a prf-adversary A' against* NMAC *such that:*

$$\mathsf{adv}^{\mathtt{prf}}_{\mathsf{Nprf}^{\overline{H}}}(A) \leq \mathsf{adv}^{\mathtt{prf}}_{\mathsf{Nmac}^{\overline{H}}}\left(A'\right) + qv^2/2^\kappa.$$

Besides, A' has time-complexity at most $T + O(qv)$ and makes at most $2vq$ queries of at most u blocks.

4.2 Security Results

In this subsection, we adapt theorems of the previous section to the case of TLS.

Theoretical Results First we give the security result for a long key, that is a s-block key with $s \geq 2$. Note the similarity with theorem 7. It is proved exactly the same way, except that at the end of the proof the NPRF prf-security is introduced and is reduced to the prf-security of NMAC.

Theorem 12. *Let h be a function family from $\{0,1\}^\kappa \times \{0,1\}^b$ to $\{0,1\}^\kappa$. Let ipad and opad be two b-bit strings and let $\Phi = \{\Delta_{ipad}, \Delta_{opad}\}$. Let A be a pf-cre-adversary against* HPRF *that has time-complexity at most T, generate labels of at most ℓ blocks and a key of $s \geq 2$ blocks and min-entropy m. Assume that* HPRF *is a concatenation of v* HMAC*. Then there exist a rka-adversary A_2 against $\widehat{h}$ and three prf-adversaries A_1, A_3, A_4 such that $\mathsf{adv}^{\mathtt{pf-cre}}_{\mathsf{Hmac}^{\overline{H}}}(A)$ is upper bounded by:*

$$\sqrt{2^t\left(3 \cdot 2^{-m} + 2s \cdot \mathsf{adv}^{\mathtt{prf}}_h\left(A_1\right)\right)} + \mathsf{adv}^{\mathtt{rka}}_{\widehat{h}}\left(A_2\right)$$
$$+\mathsf{adv}^{\mathtt{prf}}_h\left(A_3\right) + 4v^2\ell \cdot \mathsf{adv}^{\mathtt{prf}}_h\left(A_4\right) + \frac{2v^2}{2^t} + \frac{v^2}{2^\kappa}$$

where A_1 and A_2 make at most 2 queries and have time-complexity T, A_3 makes $2v$ queries with time-complexity T and A_4 makes at most 2 queries with time-complexity $\mathcal{O}(\ell \cdot T_h)$.

When the key is not longer than a block, similarly to theorem 5, we could establish a security result where there would be the term $\sqrt{2^{2\kappa}(2^{-m} + \varepsilon)}$ for some ε. This term is small for SHA-1, but it is greater than 1 in the case of SHA-384, since $b = 2\kappa$ and $m \leq b$. Therefore, we adapt the security hypothesis and assume that $\widehat{h}$ is a prf resistant against related key attacks when it is keyed with a bit string of min-entropy at least m ($m = \kappa$ for the classical rka). That is, the aim and the power of the m-rka are the same as the ones of a classical rka-adversary, excepted that at the beginning of the game, the m-rka adversary generates an efficiently samplable distribution of min-entropy at least m, gives it to the challenger and the latter chooses the key according

to this distribution. We say that $\widehat{h}$ is resistant against m-rka and note $\mathsf{adv}^{\mathtt{m-rka}}_{\widehat{h},\Phi}(A)$ the advantage of a m-rka adversary against $\widehat{h}$.

This assumption cannot be reduced to the h prf-security against rka. Indeed, the prf assumption requires that the key is uniformly distributed and a good prf for a uniformly distributed key is not necessary a good prf for a high-entropy key. Consider the following example. Let f from $\{0,1\}^n \times \{0,1\}^n$ to $\{0,1\}^n$ be the function family defined by $f_K(x) = K \oplus x$. If K is chosen uniformly at random in $\{0,1\}^n$, the function family f is a perfect random function family against adversaries which are limited to ask one query. But if $K = K' \| 0$ where K' is chosen uniformly in $\{0,1\}^{n-1}$, K is a n-bit string with min-entropy $n-1$ and f is not secure any more against adversaries which are limited to ask one query, since the f output least significant bit can be guessed.

Theorem 13. *Let ipad and opad be two fixed b-bit string, let $\Phi = \{\Delta_{ipad}, \Delta_{opad}\}$ and let h be a function family from $\{0,1\}^\kappa \times \{0,1\}^b$ to $\{0,1\}^\kappa$ which is resistant against m-rka. Let A be a cre-adversary against* Hprf *that has time-complexity at most T, generate labels of at most ℓ blocks and a key of at most 1 block and min-entropy m. Assume that* Hprf *is a concatenation of v* Hmac*. Then there exist a m-rka adversary A_0 and two prf-adversaries A_1, A_2 such that the advantage $\mathsf{adv}^{\mathtt{pf-cre}}_{\mathit{Hprf}^{\overline{H}}}(A)$ is upper bounded by:*

$$\mathsf{adv}^{\mathtt{m-rka}}_{\widehat{h},\Phi}(A_0) + \mathsf{adv}^{\mathtt{prf}}_{h}(A_1) + 4v^2\ell \cdot \mathsf{adv}^{\mathtt{prf}}_{h}(A_2) + \frac{2v^2}{2^t} + \frac{v^2}{2^\kappa}$$

where A_0 makes at most two queries with time complexity T, A_1 makes $2v$ query with time-complexity T and A_2 makes at most 2 queries with time-complexity $\mathcal{O}(\ell \cdot T_h)$, where T_h is the time for one computation of h.

Practical Security The tls standard imposes that the master-secret is 384-bit long. Therefore if one uses sha-384 as the underlying hash function, $v = 1$, $\kappa = 512$ and $t = 384$. The label and the two random nonces, when concatenated, are 616-bit long and then smaller than the 1024-bit block of sha-384, that is why in practice $\ell = 1$.

Let h denote the compression function of sha-384. Assume that the best-known prf-adversary against h in time T, is the exhaustive search adversary whose advantage is smaller than $(T/T_h)/2^\kappa$. Similarly, assume that the best known m-rka-adversary against h with time-complexity T and with $\Phi = \{\Delta_{ipad}, \Delta_{opad}\}$ is the exhaustive search adversary whose advantage is smaller than $(T/T_h)/2^m$.

We examine, in this context, the practical security of the key derivation, when the master-secret is smaller than the block size and when it is longer than the block size. For a long key of $s = 2$ blocks, the pf-cre-advantage of an adversary in time T is smaller than $\sqrt{(T/T_h) \cdot 2^{-124}}$ if $m \geq 512$. This implies a 62-bit security if $m \geq 512$.

For a small key, the cre-advantage of an adversary in time T is smaller than $(T/T_h)\left(2^{-383} + 2^{-m+3}\right)$. This implies at least a 124-bit security if $m \geq 128$.

In the case of RSA, the premaster-secret length is 384 bits which is smaller than the 1024-bit block. As its min-entropy is 368 bits, therefore, this case has a 124-bit security at least.

In the case of Diffie-Hellman, if the DDH assumption is true then the result of the key exchange is indistinguishable for the adversary from a random element in the group. Therefore, with the DDH assumption, if the key exchange is performed in a subgroup G of $\mathbb{Z}_p^\star$, where p is a prime of exactly 1024 bits, a 256-bit subgroup is enough

to guarantee a 124-bit security. If p is strictly larger than 1024-bit block size, then G has to be at least a 512-bit subgroup to guarantee a 62-bit security.

When the IV is not Random Our security proofs rely on the fact that IV (and for HMAC, *ipad* also) is chosen randomly every time a new master-secret is extracted. However, IV (and *ipad*) are fixed once for all in the HMAC standard [5] and cannot vary. Consequently, it may seem that our proofs are not of practical interest. Fortunately, it is not the case.

Indeed, our definition of computational randomness extractor allows the adversary to make only one query to guess the bit b. However, one could allow the adversary to make at most q queries with the same IV. In this case, using an hybrid argument, it can be proven that the advantage of the adversary is upper bounded by q times its advantage in the one-query game.

It implies that if IV was generated randomly when the HMAC standard was written, then the advantage of any cre-adversary against the TLS extraction function or HMAC increases linearly with the number of master-secret extractions the adversary witnesses. Such an assumption has already been made by Barak *et al.* [1] with the same consequences upon the security bound. One can found a proof of it in the particular case of the Leftover Hash Lemma in Shoup's book [20] (see theorem 6.22.).

5 Conclusion

We have shown that HMAC is a good randomness extractor, whatever the size of the key is, even when it is greater than the block size. These results can be applied to the security of the TLS key extraction function. Our results promote the use of SHA-384 as the hash function in the key extraction function. We can guarantee a security of 124 bits in the case of RSA key exchange and in the case of Diffie-Hellman key exchange with a 1024-bit prime for a 256-bit group size, which is very reasonable. We can also guarantee a 62-bit security in the case of Diffie-Hellman key exchange with a prime longer than 1024 bits for a 512-bit group size.

References

1. B. Barak, R. Shaltiel, and E. Tromer. True random number generators secure in a changing environment. In C. D. Walter, Çetin Kaya Koç, and C. Paar, editors, *CHES 2003*, volume 2779 of *LNCS*, pages 166–180. Springer, Sept. 2003.
2. M. Bellare. New proofs for NMAC and HMAC: security without collision-resistance. In *Crypto '06*, LNCS 4117. Springer-Verlag, Berlin, 2006.
3. M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *Crypto '96*, LNCS 1109. Springer-Verlag, Berlin, 1996.
4. M. Bellare, R. Canetti, and H. Krawczyk. Message authentication using hash functions: the HMAC construction. *RSA Laboratories' Cryptobytes*, 2(1), Spring 1996.
5. M. Bellare, R. Cannetti, and H. Krawczyk. HMAC: keyed-hashing for message authentication, february 1997. RFC 2104 Available from `http://www.ietf.org/rfc.html`.
6. R. Canetti, J. Friedlander, S. Konyagin, M. Larsen, D. Lieman, and I. Shparlinski. On the statistical properties of Diffie-Hellman distributions. *Israel Journal of Mathemathics*, 120:23–46, 2000.
7. R. Canetti, J. Friedlander, and I. Shparlinski. On certain exponential sums and the distribution of Diffie-Hellman triples. *Journal of the London Mathematical Society*, 59(2):799–812, 1999.
8. O. Chevassut, P. A. Fouque, P. Gaudry, and D. Pointcheval. The twist-augmented technique for key exchange. In *PKC '06*, LNCS. Springer-Verlag, Berlin, 2006.
9. Q. Dang and T. Polk. Hash-based key derivation function (hkd). draft-dang-nistkdf-01.txt, June 2006.

10. T. Dierks and C. Allen. *RFC 2246 - The TLS protocol version 1.0.* Internet Activities Board, Jan. 1999.
11. T. Dierks and E. Rescorla. The Transport Layer Security (TLS) protocol version 1.2, July 2007. *Internet Request for Comment RFC 4346 bis*, Internet Engineering Task Force.
12. Y. Dodis, R. Gennaro, J. Håstad, H. Krawczyk, and T. Rabin. Randomness extraction and key derivation using the CBC, cascade and HMAC modes. In *Crypto '04*, LNCS, pages 494–510. Springer-Verlag, Berlin, 2004.
13. P.-A. Fouque, D. Pointcheval, J. Stern, and S. Zimmer. Hardness of distinguishing the MSB or LSB of secret keys in Diffie-Hellman schemes. In M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, editors, *ICALP 2006, Part II*, volume 4052 of *LNCS*, pages 240–251. Springer, July 2006.
14. R. Gennaro, H. Krawczyk, and T. Rabin. Secure hashed Diffie-Hellman over non-DDH groups. In *Eurocrypt '04*, LNCS 3027, pages 361–381. Springer-Verlag, Berlin, 2004.
15. J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal of Computing*, 28(4):1364–1396, 1999.
16. R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *Proc. of the 30th FOCS*, pages 248–253. IEEE, New York, 1989.
17. J. Jonsson and B. S. Kaliski Jr. On the security of RSA encryption in TLS. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 127–142. Springer, Aug. 2002.
18. C. Kaufman. *RFC 4306: Internet Key Exchange (IKEv2) protocol*, Dec. 2005.
19. Recommendations for pair-wise key establishment schemes using discrete logarithm cryptography (revised). NIST Special Publications 800-56A, Mar. 2007.
20. V. Shoup. *A computational introduction to number theory and algebra.* Cambridge University Press, 2005.

A Security Proof for Small Keys

In this appendix we give the two proofs of the theorem of subsection 3.2, when the key is smaller than the block size. First we remind the theorem when HMAC is used with a classical iterated hash function and give its proof, and then remind the theorem in the case of a truncated iterated hash function with its proof.

Theorem 2. *Let IV be a fixed κ-bit string and let h be a function family from $\{0,1\}^\kappa \times \{0,1\}^b$ to $\{0,1\}^\kappa$, where the key is the first input on κ bits. Let h' be the hash function defined by $h'_{IV}(pad, \cdot) = h(IV, \cdot \oplus pad)$ where the key is pad. Let A be a cre-adversary against the construction that has time-complexity at most T, generates labels of at most ℓ blocks and a key of at most 1 block and min-entropy m. Then there exist one prf-adversary A_1 against h' and two prf-adversaries A_2 and A_3 against h such that $\mathsf{adv}^{\mathtt{cre}}_{Hmac^H}(A)$ is upper bounded by:*

$$\frac{\sqrt{2^{2\kappa}\left(2^{-m} + 2 \cdot \mathsf{adv}^{\mathtt{prf}}_{h'}(A_1)\right)}}{2} + \frac{1}{2^\kappa} + \mathsf{adv}^{\mathtt{prf}}_{h}(A_2) + 2\ell \cdot \mathsf{adv}^{\mathtt{prf}}_{h}(A_3)$$

where A_1 makes two queries with time-complexity $T + 2T_h$, A_2 makes one query with time-complexity T and A_3 makes at most 2 queries with time-complexity $\mathcal{O}(\ell \cdot T_h)$, where T_h is the time for one computation of h.

Proof. Before considering the proof itself, we prove that the hash function family

$$F = (h(IV, \cdot \oplus ipad) \| h(IV, \cdot \oplus opad))_{(ipad, opad)}$$

is cau. Indeed since any prf-adversary A' against h' with 2 queries and a time-complexity T has a prf-advantage denoted $\mathsf{adv}^{\mathtt{prf}}_{h'}(A')$, any prf-adversary A_F against F

with time-complexity $T-2T_h$ and with 2 queries, has a prf-advantage which is smaller than $2\mathsf{adv}^{\mathtt{prf}}_{h'}(A')$. Then, it can be easily seen that from any cau-adversary against F one can construct a prf-adversary against F. This implies that any cau-adversary against F which has a time-complexity at most $T+2T_h$ and generates probability distributions of at least min-entropy at least m (for any m !), has a cau-advantage which is upper bounded by $\mathsf{adv}^{\mathtt{prf}}_{F}(A)+2^{-t}$, for a particular prf-adversary A against F with two queries and same time complexity.

Let consider now the following sequence of games.

Game 0: this game corresponds to the attack when the real extraction is performed.

1. A sends $(\mathcal{D}, label)$
2. $pmk \xleftarrow{\mathcal{D}} \{0,1\}^s$, $opad \xleftarrow{\$} \{0,1\}^b$, $ipad \xleftarrow{\$} \{0,1\}^b$
3. $(k_1,k_2) = (h(IV, pmk \oplus ipad), h(IV, pmk \oplus opad))$
4. $k = \mathsf{Nmac}^H(k_1,k_2,label)$, send $(IV, ipad, k)$ to A
5. A sends its guess b'

Game 1: in this game, we choose k_1 and k_2 uniformly at random in $\{0,1\}^\kappa$.

Game 2: in this game, we choose k uniformly at random in $\{0,1\}^k$.

Firstly, the distance between Game 0 and Game 1 can be upper bounded using the computational Leftover Hash Lemma with F: it is upper bounded by

$$\sqrt{2^{2\kappa}\cdot(2^{-m}+2\cdot\mathsf{adv}^{\mathtt{prf}}_{h'}(A_1))}.$$

Secondly there exists a prf-adversary A' against NMAC which makes at most one query and has time-complexity T such that the distance between Game 1 and Game 2 is upper bounded by $\mathsf{adv}^{\mathtt{prf}}_{\mathsf{Nmac}^H}(A')$. Bellare's result implies that the latter is smaller than $\mathsf{adv}^{\mathtt{prf}}_{h}(A_3)+2\ell\cdot\mathsf{adv}^{\mathtt{prf}}_{h}(A_4)+1/2^\kappa$. □

We consider now the case when HMAC is used with an truncated iterated hash function.

Theorem 5. *Let IV be a fixed κ-bit string and let h be a function family from $\{0,1\}^\kappa\times\{0,1\}^b$ to $\{0,1\}^\kappa$, where the key is the first input on κ bits. Let h' be the hash function defined by $h'_{IV}(pad,\cdot)=h(IV,\cdot\oplus pad)$ where the key is pad. Let A be a cre-adversary against the construction that has time-complexity at most T, generates labels of at most ℓ blocks and a key of at most 1 block and min-entropy m. Then there exist one prf-adversary A_1 against h' and two prf-adversaries A_2 and A_3 against h such that $\mathsf{adv}^{\mathtt{cre}}_{Hmac^H}(A)$ is upper bounded by:*

$$\frac{\sqrt{2^{2\kappa}\left(2^{-m}+2\cdot\mathsf{adv}^{\mathtt{prf}}_{h'}(A_1)\right)}}{2}+\frac{1}{2^t}$$
$$+\,\mathsf{adv}^{\mathtt{prf}}_{h}(A_2)+2\ell\cdot\mathsf{adv}^{\mathtt{prf}}_{h}(A_3)$$

where A_1 makes two queries with time-complexity $T+2T_h$, A_2 makes one query with time-complexity T and A_3 makes at most 2 queries with time-complexity $\mathcal{O}(\ell\cdot T_h)$, where T_h is the time for one computation of h.

Proof. Let consider the following sequence of games.

Game 0: this game corresponds to the attack when the real extraction is performed.

1. A sends $(\mathcal{D}, label)$

2. $pmk \xleftarrow{\mathcal{D}} \{0,1\}^s$, $IV \xleftarrow{\$} \{0,1\}^\kappa$, $ipad \xleftarrow{\$} \{0,1\}^b$
3. $(k_1, k_2) = (h(pmk \oplus ipad, IV), h(pmk \oplus opad, IV))$
4. $k = \mathsf{Nmac}^{\overline{H}}(k_1, k_2, label)$, send $(IV, ipad, k)$ to A
5. A sends its guess b'

Game 1: in this game, we choose k_1 and k_2 uniformly at random in $\{0,1\}^\kappa$.
Game 2: in this game, we choose k uniformly at random in $\{0,1\}^k$.

Firstly, the distance between Game 0 and Game 1 can be upper bounded using lemma 1:

$$\sqrt{2^{2\kappa} \cdot (2^{-m} + 2 \cdot \mathsf{adv}^{\mathtt{prf}}_{h'}(A_1))}.$$

Secondly there exists a prf-adversary A' against NMAC which makes at most one query and has time-complexity T such that the distance between Game 1 and Game 2 is upper bounded by $\mathsf{adv}^{\mathtt{prf}}_{\mathsf{Nmac}^{\overline{H}}}(A')$. Since A' makes only one query, she is obviously prefix-free and therefore her advantage is in particular smaller than $\mathsf{adv}^{\mathtt{pf-prf}}_{\mathsf{Nmac}^{\overline{H}}}(A')$. The latter is smaller than $\mathsf{adv}^{\mathtt{prf}}_{h}(A_3) + 2\ell \cdot \mathsf{adv}^{\mathtt{prf}}_{h}(A_4) + \frac{1}{2^t}$, as we prove in lemma 6.

B Security Proof for Long Keys

In this section we give a proof of theorem 7. First we give the proofs for the cascade mode.

B.1 The Cascade Mode

Lemma 8. *Let A^* be a pf-au-adversary against h^* which generates messages of at most ℓ blocks. Then there is a prf-adversary A against h such that:*

$$\mathsf{adv}^{\mathtt{pf-au}}_{\overline{h^*}}(A^*) \leq 2\ell \cdot \mathsf{adv}^{\mathtt{prf}}_{h}(A) + \frac{1}{2^t}$$

and A makes at most 2 queries and has about the same time-complexity as A^*.

To show this lemma, we need the following result from [4].

Lemma 14 (BKC). *If D is a prefix-free prf-adversary against h^* that makes at most q queries, each of at most ℓ blocks, then there is a prf-adversary A against h such that*

$$\mathsf{adv}^{\mathtt{pf-prf}}_{h^*}(D) \leq q\ell \cdot \mathsf{adv}^{\mathtt{prf}}_{h}(A)$$

and A makes at most q queries and has about the same time complexity as D.

We can now prove the lemma.

Proof. Let D be the following prf-adversary:
Adversary D^{h^*}
$(M_1, M_2) \leftarrow A^*$
Send (M_1, M_2) to the challenger C, the latter answers $(h^*(M_1), h^*(M_2))$.
If $\mathsf{msb}_t(h^*(M_1)) == \mathsf{msb}_t(h^*(M_2))$ then return 1, else return 0.
Note that $\mathsf{adv}^{\mathtt{pf-au}}_{\overline{h^*}}(A^*) - 2^{-t} \leq \mathsf{adv}^{\mathtt{pf-prf}}_{h^*}(D)$ and that D makes at most 2 queries. If A^* is prefix-free and the messages are at most ℓ-block-long, lemma 14 gives us a prf-adversary A against h such that:

$$\mathsf{adv}^{\mathtt{pf-prf}}_{h^*}(D) \leq 2\ell \cdot \mathsf{adv}^{\mathtt{prf}}_{h}(A)$$

The lemma follows.

Besides we need this lemma.

Lemma 10 (pf computational LHL). *Let H be a family of functions from $\{0,1\}^k \times \mathcal{D}om$ to $\{0,1\}^t$ such that for every au-adversary B, running in time T and producing a distribution over $\mathcal{D}om \times \mathcal{D}om$ of min-entropy at least $2m-2$, $\mathsf{adv}_H^{\mathtt{pf-cau}}(B) \leq 1/2^t + \varepsilon$. Then for every adversary A running in time $\mathcal{O}(T)$ producing a distribution of min-entropy at least m:*

$$\mathsf{adv}_H^{\mathtt{pf\text{-}cre}}(A) \leq \sqrt{2^t \cdot (2^{-m} + \varepsilon)}.$$

This lemma can be proven similarly as the proof of the original Leftover Hash Lemma that can be found in [20]. However, due to the prefix-freeness assumption, one have to adapt the proof. All the details are in the last appendix.

As a direct consequence of the two above lemmas, we have the following result.

Lemma 8. *Let A be a pf-cre-adversary against $\overline{h^*}$ which has a time-complexity at most T and produces a distribution of min-entropy at least m, with messages of at most ℓ blocks. Then there is a prf-adversary A' with running-time at most $\mathcal{O}(T)$ and messages at most ℓ-block long such that:*

$$\mathsf{adv}_{\overline{h^*}}^{\mathtt{pf\text{-}cre}}(A) \leq \sqrt{2^t \cdot (3 \cdot 2^{-m} + 2\ell \cdot \mathsf{adv}_h^{\mathtt{prf}}(A'))}.$$

B.2 The case of Hmac

Theorem 7. *Let h be a function family from $\{0,1\}^\kappa \times \{0,1\}^b$ to $\{0,1\}^\kappa$. Let ipad and opad be two b-bit strings and let $\Phi = \{\Delta_{ipad}, \Delta_{opad}\}$. Let A be a pf-cre-adversary against the construction that has time-complexity at most t, generate labels of at most ℓ blocks and a key of $s \geq 2$ blocks and min-entropy m. Then there exist a rka-adversary A_2 against $\widehat{h}$ and three prf-adversaries A_1, A_3, A_4 such that $\mathsf{adv}_{\mathsf{Hmac}^{\overline{H}}}^{\mathtt{pf-cre}}(A)$ is upper bounded by:*

$$\sqrt{2^t \left(3 \cdot 2^{-m} + 2s \cdot \mathsf{adv}_h^{\mathtt{prf}}(A_1)\right)} + \mathsf{adv}_{\widehat{h}}^{\mathtt{rka}}(A_2)$$
$$+\mathsf{adv}_h^{\mathtt{prf}}(A_3) + 2\ell \cdot \mathsf{adv}_h^{\mathtt{prf}}(A_4) + \tfrac{1}{2^t}$$

where A_1 and A_2 make at most 2 queries and have time-complexity t, A_3 makes one query with time-complexity t and A_4 makes at most 2 queries with time-complexity $\mathcal{O}(\ell \cdot T_h)$.

Proof. Let consider the following sequence of games.
Game 0: this game corresponds to the attack when the real extraction is performed.

1. A sends $(\mathcal{D}, label)$
2. $pmk \xleftarrow{\mathcal{D}} \{0,1\}^s$, $IV \xleftarrow{\$} \{0,1\}^\kappa$, $K = \overline{H}(IV, pmk)$
3. $(k_1, k_2) = \left(\widehat{h}(K \oplus ipad, IV), \widehat{h}(K \oplus opad, IV)\right)$
4. $k = \mathsf{Nmac}^{\overline{H}}(k_1, k_2, labels)$, send (IV, k) to A
5. A sends its guess b'

Game 1: in this game, we choose K uniformly at random in $\{0,1\}^t$.
Game 2: in this game, we choose k_1 and k_2 uniformly at random in $\{0,1\}^\kappa$.
Game 3: in this game, we choose k uniformly at random in $\{0,1\}^k$. It corresponds to the attack when the extraction is performed thanks to a random function.

As A is prefix-free, the distance between Game 0 and Game 1 can be upper bounded using lemma 8: it is upper bounded by $\sqrt{2^t \cdot (2^{-m} + 2s \cdot \mathsf{adv}_h^{\mathtt{prf}}(A_1))}$. The distance between Game 1 and Game 2 is upper bounded by $\mathsf{adv}_{\widehat{h},\Phi}^{\mathtt{rka}}(A_2)$. As A is prefix-free, there exists an adversary A' making at most 1 query of length at most ℓ blocks with time-complexity approximately t and distance between Game 1 and Game 2 is upper bounded by $\mathsf{adv}_{\mathsf{Nmac}^{\overline{H}}}^{\mathtt{pf-prf}}(A')$. The latter is smaller than $\mathsf{adv}_h^{\mathtt{prf}}(A_3) + 2\ell \cdot \mathsf{adv}_h^{\mathtt{prf}}(A_4) + \frac{1}{2^t}$ (where A_3 and A_4 are as described in the theorem) as we proved in lemma 6. □

C Proof of the cLHL

C.1 The cLHL

Lemma 3 (computational LHL). *Let H be a family of functions from $\{0,1\}^k \times \mathcal{D}om$ to $\{0,1\}^t$ such that for every au-adversary B, running in time T and producing a distribution over $\mathcal{D}om \times \mathcal{D}om$ of min-entropy at least $2m$, $\mathsf{adv}_H^{\mathtt{cau}}(B) \leq 1/2^t + \varepsilon$. Then for every adversary A running in time $\mathcal{O}(T)$ producing a distribution of min-entropy at least m:*

$$\mathsf{adv}_H^{\mathtt{cre}}(A) \leq \sqrt{2^t \cdot (2^{-m} + \varepsilon)}.$$

Proof. Let A be a pf-cre-adversary against H which outputs distributions of min-entropy at least m, let E_1 and E_2 denote respectively the events

$$\left\{ \begin{array}{c} K \leftarrow U_{\mathcal{K}eys}, A \Rightarrow \mathcal{D}_A, \\ X \leftarrow \mathcal{D}_A, Y = H(K,X) \end{array} \right\}, \left\{ \begin{array}{c} K \leftarrow U_{\mathcal{K}eys}, A \Rightarrow \mathcal{D}_A, \\ Y \leftarrow U_{\mathcal{R}ng} \end{array} \right\}.$$

Let $\delta = \mathsf{adv}_H^{\mathtt{pf-cre}}(A)$, that is:

$$\delta = \Pr\left[A(K,Y) \Rightarrow 1 \colon E_1\right] - \Pr\left[A(K,Y) \Rightarrow 1 \colon E_2\right].$$

To show lemma 10 we successively show:

$$\frac{1+\delta^2}{2^k \cdot 2^t} \leq \Pr\left[\begin{array}{c} K = K' \\ Y = Y' \end{array} : E\right] \tag{3}$$

$$\Pr\left[\begin{array}{c} K = K' \\ Y = Y' \end{array} : E\right] \leq \frac{1}{2^k}\left(2^{-m} + \frac{1}{2^t} + \varepsilon\right) \tag{4}$$

where

$$E_1' = \left\{ \begin{array}{c} K' \leftarrow U_{\mathcal{K}eys}, A \Rightarrow \mathcal{D}'_A, \\ X' \leftarrow \mathcal{D}'_A, Y' = H(K',X') \end{array} \right\}, E = E_1 \wedge E_1'.$$

With these two results it is easy to conclude.

To show the first result, we remark that δ is equal to:

$$\sum_{y,k} \Pr\left[\begin{array}{c} A(k,y) \Rightarrow 1 \\ K = k \\ Y = y \end{array} : E_1\right] - \Pr\left[\begin{array}{c} A(k,y) \Rightarrow 1 \\ K = k \\ Y = y \end{array} : E_2\right].$$

Since the way (k,y) is chosen is independent of the event $A(k,y) \Rightarrow 1$ knowing that $A \Rightarrow \mathcal{D}_A$, this can be restated as:

$$\sum_{y,k} \Pr[A(k,y) \Rightarrow 1 \colon A \Rightarrow \mathcal{D}_A] \cdot \left(\Pr\left[\begin{array}{c} K = k \\ Y = y \end{array} : E_1\right] - \frac{1}{2^k \cdot 2^t}\right).$$

Let denote by $q_{k,y}$:

$$\frac{1}{\delta} \cdot \Pr[A(k,y) \Rightarrow 1 \colon A \Rightarrow \mathcal{D}_A] \cdot \left(\Pr\left[\begin{matrix} K = k \\ Y = y \end{matrix} \colon E_1 \right] - \frac{1}{2^k \cdot 2^t} \right).$$

As $\sum_{k,y} q_{k,y} = 1$, we have $\frac{1}{2^k \cdot 2^t} \leq \sum_{k,y} q_{k,y}^2$ and then:

$$\begin{aligned} \frac{1}{2^k \cdot 2^t} &\leq \frac{1}{\delta^2} \sum_{k,y} \left(\Pr\left[K = k \wedge Y = y \colon E_1\right] - \frac{1}{2^k \cdot 2^t} \right)^2 \\ &\leq \frac{1}{\delta^2} \left(\sum_{k,y} \Pr\left[K = k \wedge Y = y \colon E_1\right]^2 - \frac{1}{2^k \cdot 2^t} \right) \end{aligned}$$

Therefore, it follows immediately the first result:

$$\begin{aligned} \frac{1+\delta^2}{2^k \cdot 2^t} &\leq \sum_{k,y} \Pr\left[K = k \wedge Y = y \colon E_1\right]^2 \\ &\leq \Pr\left[K = K' \wedge Y = Y' \colon E\right] \end{aligned}$$

We now upper bound $\Pr\left[K = K' \wedge Y = Y' \colon E\right]$. This collision probability is equal to:

$$\Pr\left[K = K' \colon \begin{matrix} K \leftarrow U_{\mathcal{K}eys} \\ K' \leftarrow U_{\mathcal{K}eys} \end{matrix}\right] \cdot \Pr\left[H(K,X) = H(K,X') \colon \begin{matrix} A \Rightarrow \mathcal{D}_A, X \leftarrow \mathcal{D}_A, \\ A \Rightarrow \mathcal{D}'_A, X' \leftarrow \mathcal{D}'_A, \\ K \leftarrow U_{\mathcal{K}eys} \end{matrix}\right]$$

This can be restated as:

$$\frac{1}{2^k} \left(\Pr\left[X = X' \colon \begin{matrix} A \Rightarrow \mathcal{D}_A, X \leftarrow \mathcal{D}_A, \\ A \Rightarrow \mathcal{D}'_A, X' \leftarrow \mathcal{D}'_A, \end{matrix}\right] + \Pr\left[\begin{matrix} X \neq X' \\ H(K,X) = H(K,X') \end{matrix} \colon \begin{matrix} A \Rightarrow \mathcal{D}_A, X \leftarrow \mathcal{D}_A, \\ A \Rightarrow \mathcal{D}'_A, X' \leftarrow \mathcal{D}'_A, \\ K \leftarrow U_{\mathcal{K}eys} \end{matrix}\right] \right)$$

Let denote by $\mathcal{D}_A$ and $\mathcal{D}'_A$ two distributions given by two independent runs of A and $\mathcal{D}_B$ the product distribution $\mathcal{D}_A \times \mathcal{D}'_A$. Let B be the following cau-adversary: B runs A twice independently, simulating A challenger. Let denote $\mathcal{D}_A$ and $\mathcal{D}'_A$ the two distributions given by A in each run and $\mathcal{D}_B$ the probability distribution which is sent by B to the challenger which is constructed as describe above. Since $\mathcal{D}_A$ and $\mathcal{D}'_A$ have min-entropy at least m, $\mathcal{D}_B$ has min-entropy at least $2m$.

The time complexity of B is $2T_A + \mathcal{O}(1)$ and its cau-advantage is exactly:

$$\Pr\left[\begin{matrix} X \neq X' \\ H(K,X) = H(K,X') \end{matrix} \colon \begin{matrix} A \Rightarrow \mathcal{D}_A, X \leftarrow \mathcal{D}_A, \\ A \Rightarrow \mathcal{D}'_A, X' \leftarrow \mathcal{D}'_A, \\ K \leftarrow U_{\mathcal{K}eys} \end{matrix}\right]$$

This cau-advantage is upper bounded by $1/2^t + \varepsilon$, thus $\Pr[K = K' \wedge Y = Y' \colon E]$ is upper bounded by:

$$\frac{1}{2^k}\left(\Pr\left[X = X' \colon \begin{array}{l} A \Rightarrow \mathcal{D}_A, X \leftarrow \mathcal{D}_A, \\ A \Rightarrow \mathcal{D}'_A, X' \leftarrow \mathcal{D}'_A, \end{array}\right] + \frac{1}{2^t} + \varepsilon\right).$$

For every fixed distributions $\mathcal{D}_A$ and $\mathcal{D}'_A$, the probability $\Pr\left[X = X' \colon \begin{array}{l} X \leftarrow \mathcal{D}_A, \\ X' \leftarrow \mathcal{D}'_A, \end{array}\right]$ is equal to:

$$\sum_x \Pr\left[X = x \colon X \leftarrow \mathcal{D}_A\right] \Pr\left[X' = x \colon X' \leftarrow \mathcal{D}'_A,\right].$$

This can be upper bounded using Cauchy-Schwarz inequality, namely it is upper bounded by:

$$\sqrt{\sum_x \Pr\left[X = x \colon X \leftarrow \mathcal{D}_A\right]^2} \sqrt{\sum_x \Pr\left[X' = x \colon X \leftarrow \mathcal{D}'_A\right]^2}$$

As $\mathcal{D}_A$ and $\mathcal{D}_A$ has min-entropy at least m, this is smaller than $2^{-m/2} \cdot 2^{-m/2} \leq 2^{-m}$. As this is true for every fixed $\mathcal{D}_A$ and $\mathcal{D}'_A$, this is true for $\Pr\left[X = X' \colon \begin{array}{l} A \Rightarrow \mathcal{D}_A, X \leftarrow \mathcal{D}_A, \\ A \Rightarrow \mathcal{D}'_A, X' \leftarrow \mathcal{D}'_A, \end{array}\right]$ and we have:

$$\Pr\left[K = K' \wedge Y = Y' \colon E\right] \leq \frac{1}{2^k}\left(2^{-m} + \frac{1}{2^t} + \varepsilon\right)$$

Combining the previous equation with equation 3 we have:

$$\frac{1 + \delta^2}{2^k \cdot 2^t} \leq \frac{1}{2^k}\left(2^{-m} + \frac{1}{2^t} + \varepsilon\right)$$

It follows immediately:

$$\delta \leq \sqrt{2^t\,(2^{-m} + \varepsilon)}.\square$$

If $\varepsilon \leq 2^{-m}$, and we want to impose a 2^{-e} security, this result can be restated as:

$$m \geq t + 2e + 1.$$

C.2 The Prefix-Free cLHL

Lemma 10 (pf computational LHL). *Let H be a family of functions from $\{0,1\}^k \times \mathcal{D}om$ to $\{0,1\}^t$ such that for every au-adversary B, running in time T and producing a distribution over $\mathcal{D}om \times \mathcal{D}om$ of min-entropy at least $2m-2$, $\mathsf{adv}_H^{\mathtt{pf-cau}}(B) \leq 1/2^t + \varepsilon$. Then for every adversary A running in time $\mathcal{O}(T)$ producing a distribution of min-entropy at least m:*

$$\mathsf{adv}_H^{\mathtt{pf\text{-}cre}}(A) \leq \sqrt{2^t \cdot (3 \cdot 2^{-m} + \varepsilon)}.$$

Proof. The proof is similar to the previous proof, excepted the way the cau-adversary B is simulated.

Let denote by $\mathcal{D}_A$ and $\mathcal{D}'_A$ two distributions given by two independent runs of A and $\mathcal{D}_\pi$ the product distribution $\mathcal{D}_A \times \mathcal{D}'_A$. First note that since $\mathcal{D}_A$ may be different from $\mathcal{D}'_A$, for every $X \leftarrow \mathcal{D}_A$ and $X' \leftarrow \mathcal{D}'_A$ we may have $X \subset X'$ or $X' \subset X$ and $\mathcal{D}_\pi$ is not guaranteed to be prefix-free. Therefore the adversary B cannot generates the probability distribution $\mathcal{D}_\pi$.

Let $\mathcal{D}_B$ be the following distribution: choose (X, X') following $\mathcal{D}_\pi$, if $X \subset X'$ or $X' \subset X$, then choose Y uniformly at random in $\mathcal{D}om$ and output (Y, Y), else output (X, X'). Let show that this efficiently samplable probability distribution has min-entropy at least $2m - 2$. By definition of prefix-freeness for every x' there is at most one x such that $x \subset x'$, $\Pr_{\mathcal{D}_a}[X = x] > 0$ and $\Pr_{\mathcal{D}'_a}[X' = x'] > 0$. Therefore, the probability that $X \subset X'$ is equal to $\sum_{x'} \Pr[X' = x'] \sum_x \Pr[X = x] \Pr[x \subset x']$. Due to the previous remark, there is at most one term in the sum $\sum_x \Pr[X = x] \Pr[x \subset x']$ and the sum is upper bounded by the min-entropy 2^{-m}. Thus, $\Pr[X \subset X']$ is upper bounded by 2^{-m}. Therefore the probability to obtain a couple (x, x') which is not prefix-free is smaller than 2^{-2m} and the probability of a couple (y, y) is upper bounded by $2^{-2m} + 2 \cdot 2^{-m} \cdot |\mathcal{D}om|^{-1}$, which is upper bounded by $4 \cdot 2^{-2m}$. The min-entropy of $\mathcal{D}_B$ is thus greater than $2m - 2$ and $\mathcal{D}_B$ is prefix-free.

The time complexity of B is $2T_A + \mathcal{O}(1)$ and its cau-advantage is exactly:

$$\Pr\left[\begin{array}{c} X \neq X' \\ H(K,X) = H(K,X') \end{array} : \begin{array}{c} A \Rightarrow \mathcal{D}_A, A \Rightarrow \mathcal{D}'_A, \\ (X, X') \leftarrow \mathcal{D}_B, \\ K \leftarrow U_{\mathcal{K}eys} \end{array}\right].$$

This cau-advantage is upper bounded by $1/2^t + \varepsilon$. Besides, it is equal to

$$\begin{array}{l} \sum_{\substack{(x,x') \\ x \not\subset x' \\ x' \not\subset x}} \Pr\left[H(K,x) = H(K,x') : K \leftarrow U_{\mathcal{K}eys}\right] \\ \qquad \cdot \quad \Pr_{\mathcal{D}_B}\left[(X, X') = (x, x')\right]. \end{array}$$

If (x, x') is prefix-free, $\Pr_{\mathcal{D}_\pi}[(X, X') = (x, x')]$ is equal to $\Pr_{\mathcal{D}_B}[(X, X') = (x, x')]$, therefore previous formula equals

$$\begin{array}{l} = \sum_{(x,x')} \Pr\left[\begin{array}{c} x \neq x' \\ H(K,x) = H(K,x') \end{array} : K \leftarrow U_{\mathcal{K}eys}\right] \\ \qquad \cdot \quad \Pr_{\mathcal{D}_\pi}\left[(X, X') = (x, x')\right] \\ - \sum_{\substack{(x,x') \\ x \subset x' \\ x' \subset x}} \Pr\left[\begin{array}{c} x \neq x' \\ H(K,x) = H(K,x') \end{array} : K \leftarrow U_{\mathcal{K}eys}\right] \\ \qquad \cdot \quad \Pr_{\mathcal{D}_\pi}\left[(X, X') = (x, x')\right] \end{array}$$

Therefore the probability

$$\Pr\left[\begin{array}{c} X \neq X' \\ H(K,X) = H(K,X') \end{array} : \begin{array}{c} A \Rightarrow \mathcal{D}_A, A \Rightarrow \mathcal{D}'_A, \\ (X, X') \leftarrow \mathcal{D}_\pi, \\ K \leftarrow U_{\mathcal{K}eys} \end{array}\right]$$

is equal to

$$\begin{array}{rc} \displaystyle\sum_{\substack{(x,x') \\ x \subset x' \\ x' \subset x}} \Pr\left[\begin{array}{c} x \neq x' \\ H(K,x) = H(K,x') \end{array} : K \leftarrow U_{\mathcal{K}eys}\right] & \\ \cdot & \Pr_{\mathcal{D}_\pi}\left[(X, X') = (x, x')\right] \\ + & \mathsf{adv}^{\mathsf{cau}}_H(B). \end{array}$$

It is smaller than $\Pr[(X \subset X' \vee X' \subset X) \wedge X \neq X'] + 2^t + \varepsilon$, which is upper bounded by $2 \cdot 2^{-m} + 2^t + \varepsilon$. With the upper bound for this probability, the proof is similar to the previous one. □