Key Derivation and Randomness Extraction

Olivier Chevassut[1], Pierre-Alain Fouque[2], Pierrick Gaudry[3], and David Pointcheval[2]

[1] Lawrence Berkeley National Lab. – Berkeley, CA, USA – OChevassut@lbl.gov
[2] CNRS-École normale supérieure – Paris, France – {Pierre-Alain.Fouque,David.Pointcheval}@ens.fr
[3] CNRS-École polytechnique – Palaiseau, France – gaudry@lix.polytechnique.fr

Abstract. Key derivation refers to the process by which an agreed upon large random number, often named master secret, is used to derive keys to encrypt and authenticate data. Practitioners and standardization bodies have usually used the random oracle model to get key material from a Diffie-Hellman key exchange. However, proofs in the standard model require randomness extractors to formally extract the entropy of the random master secret into a seed prior to derive other keys.

This paper first deals with the protocol Σ_0, in which the key derivation phase is (deliberately) omitted, and security inaccuracies in the analysis and design of the Internet Key Exchange (IKE version 1) protocol, corrected in IKEv2. They do not endanger the practical use of IKEv1, since the security could be proved, at least, in the random oracle model. However, in the standard model, there is not yet any formal global security proof, but just separated analyses which do not fit together well. The first simplification is common in the theoretical security analysis of several key exchange protocols, whereas the key derivation phase is a crucial step for theoretical reasons, but also practical purpose, and requires careful analysis. The second problem is a gap between the recent theoretical analysis of HMAC as a good randomness extractor (functions keyed with public but random elements) and its practical use in IKEv1 (the key may not be totally random, because of the lack of clear authentication of the nonces). Since the latter problem comes from the probabilistic property of this extractor, we thereafter review some *deterministic* randomness extractors and suggest the *'Twist-AUgmented'* technique, a new extraction method quite well-suited for Diffie-Hellman-like scenarios.

Key Words: Randomness extractors, Key derivation, Elliptic curve

1 Introduction

Key exchange is an important problem in practice and several schemes have been designed to solve it since the seminal work of Diffie and Hellman [20]. Recently, different works have been published in order to analyze the security of those schemes in various settings (password, public-key, hybrid setting) and security models (random oracle, common reference string, standard model). But for several years, efficiency and security in the standard model are one of the main goals to achieve in cryptography. The most widely used network security protocols nowadays are the TLS [47], a.k.a SSL, and the Internet Key Exchange (IKE) protocols [28, 35] from the IPSec standard of the IETF. Several papers [18, 19, 37, 25, 22] have studied the security of the IKE schemes and proposed a security analysis in the *standard model*. However, these papers do not precisely consider the protocol but discuss on either simplified descriptions of some parts, or isolated primitives, which do not *rigorously* match the protocol designed by the IETF. Therefore, there can be some gaps between the formal security analysis and the actual security of the protocol.

1.1 Motivation

The Key Derivation Problem. Diffie-Hellman (DH) based key exchanges establish a secure communication channel between two parties by securely negotiating a large random element in a given cyclic group, called master secret. Then, this secret is used to derive keys for encrypting and authenticating data. These keys must be bit-strings of some specific length uniformly distributed and used as input parameters to symmetric ciphers (for privacy), message authentication codes (for authentication), and pseudo-random functions (for expansion of a seed into a longer bit-string). However, they cannot be initialized with the *simple* bit-string encoding of the master secret. Even though this secret is indistinguishable from a random element in the cyclic group under some classical computational assumptions, such as the Decisional Diffie-Hellman assumption (DDH), its encoding is not indistinguishable from a random bit-string with a uniform distribution. The entropy of the bit-string encoded secret is indeed high but not high enough to immediately obtain an *almost* uniformly distributed random bit-string: pseudo-entropy generators are not pseudo-random generators even when only considering the property of computational indistinguishability [29].

Most of the cryptographic protocols do not take into account this practical problem since it only appears during the concrete implementation. Cryptographers indeed use "elements in sets" when designing their algorithms while standardization bodies represent and encode these elements. Engineers are left clueless when elements in a given set do not necessarily admit a compact encoding —in bijection with a set of ℓ-bit strings— even for a well-chosen ℓ. Practitioners have no choice but to make educated guesses on which encoding to use and so, may introduce security breaches. This is the case of the Diffie-Hellman version of the SSL protocol [47] where the binary encoding of the random element is used as it. This practice can not be proved secure in the standard model, although the protocol may be secure in the random oracle model. However, nobody claims any security proof despite the widely use of SSL. See also [14] for a similar remark on the ElGamal encryption scheme.

The Case of IKE. More interestingly, IKE raises this problem too. It can be emphasized in the security analysis of the basic SIGMA protocol Σ_0 provided by Canetti and Krawczyk [36, 19, 37]. SIGMA and IKE are well-designed protocols with many nice properties and all the previous analysis justify the whole design. In particular, IKE explicitly deals with the extraction issue via a mechanism analyzed in [22]. However, there is some inaccuracies when we want to go through the details. The authors [19] were definitely aware of the key derivation problem at the writing time for several years ago but they chose to eliminate the *mandatory* key derivation phase in this basic version of SIGMA, without any justification (maybe a simpler security proof). They just referred to it in some notes and section later in [19], and in the last appendix of the full version of the companion paper [37]. Nevertheless, some sentences of [19] are, to say the least, *ambiguous* about the key derivation problem, which also arises in the security analysis, even at a theoretical level, and not only for practitioners. They could lead to *annoying* misinterpretation. Furthermore, the same omission appeared elsewhere in the literature, and namely in the JFK protocol [2, 3]. Both of these protocols are the basis of the second version of IKE [35], however the latter avoids all the problems we mention here.

Since the key derivation problem is a very important step, we choose to draw attention to it. Even if known solutions exist, it is still possible to introduce security breaches, as it is the case in IKEv1. These breaches do not lead to practical attacks on the protocol, but they must be taken into account in a theoretical viewpoint.

1.2 Randomness Extraction and Key Derivation Phase

The protocol Σ_0 in [19] provides the two entities with a master secret (a Diffie-Hellman value) and uses it as a key to a pseudo-random function (PRF). As we previously said, even if this secret is indistinguishable from a truly random element in a group under the DDH assumption, it is not a random bit-string uniformly distributed in the key space of the PRF. Such a description of IKE is a simplified one, and thus the theoretical paper [19] does not provide *by itself* a full justification of IKE. More precisely, the actual IKE protocol does not directly use the master secret as a key of the PRF (as analyzed in [19]) and it is rather apparently well done:

Randomness Extraction in a first stage, the IKE standard uses a PRF keyed by *random and public nonces* and applies it to the master secret;

Key Derivation in the second stage, the output is used as a key to a PRF, with known inputs in order to derive further key material to create a secure channel.

This two-phases protocol also appears in the random generator architecture of Barak and Halevi [4]. The aim of the randomness extractor phase is to generate a short seed concentrating the entropy of the source and then in the key derivation, this seed will be used to generate keys. It is important to separate these stages, since different cryptographic primitives are needed: the PRF is used for two different tasks!

Before going into more details, let us review informally the main difference between randomness extractors and PRF. A PRF is a family of functions, from a set D on a set R, such that it is computationally hard to distinguish the inputs/outputs of a function taken at random from the set of all functions from D to R and of a function taken at random in the PRF family. It is important to note that the key, or the index of the function taken in the PRF family, must be kept secret, otherwise the distinction becomes easy. A randomness extractor has the property that the output distribution is close to the uniform one, if the input distribution has enough entropy. If the index is known, the randomness extractor is called a *strong* randomness extractor[1].

As a consequence, one can easily note that in the description of IKE the notation `prf` has two purposes: (1) first stage, `prf` is used as a randomness extractor, with public and random key and high-entropy input (but not as a PRF); (2) second stage, `prf` is used as a PRF, to build a PRG. The HMAC function [7], designed and analyzed as a secure MAC, is furthermore the default `prf` in IKE.

1.3 HMAC as a Randomness Extractor

HMAC, as well as some other constructions, have been recently studied as randomness extractors by Dodis *et al.* in [22]. This is the first formal analysis of practical randomness extractors. They namely prove that these constructions are almost universal hash functions under various assumptions. They basically show how to construct a variable-input length almost universal hash function family from a fixed-input length almost universal hash function family (or even random functions/permutations). Thereafter, a little modification of the Leftover Hash Lemma (LHL) [31] with a *randomly chosen function* from a family of (almost) universal hash functions can be used to extract the entropy of a random source. Such a theoretical result seems to match with the use of HMAC in IKE, as a randomness extractor. At least, the authors [22] claim to justify the design of IKE.

However, if one carefully looks at the description of IKEv1 (in the signature mode) in the standard [28], one can note that the LHL is not applied in an appropriate way: the key of the universal hash function may be chosen by the adversary. There is thus no guarantee of true randomness, while this is the basic assumption. More precisely, the random key is jointly chosen by the players, but not clearly authenticated. An adversary could take advantage of that (with some stupid but theoretically secure extractors) to control half of the bits. One could say that one honest player is enough to inject high entropy in this key. But a high-entropy bit-string *is not* a random bit-string. For extracting entropy, we are back to our initial problem... This security problem does not endanger the IKEv1 protocol in practice, due to the use of a specific extractor (HMAC), but it makes impossible any security proof under the sole assumption of "almost universal hash function". Note that the nonces are properly signed in IKEv2.

Furthermore, the universal-hashing property of HMAC is provided in three analyses [22]: the first one unfortunately considers a *variant* of HMAC, where large hash functions are used, with a truncated output. The theoretical analysis is indeed delicate since compression functions are families where the size of the key is the same as the size of the output. Hence, the second analysis, which is announced, but not yet available may be quite intricate. Anyway, the third one validates, at least for practical purpose, the use of HMAC, since this is an analysis of the real HMAC in the random oracle model [10].

[1] Hereafter, we only look at strong randomness extractors and we thus simply call them randomness extractors.

1.4 Randomness Extractors

The notion of randomness extractor is thus very important from a practical point of view and is often ignored or misused by cryptographers, since solutions are quite theoretical. Let us call the ratio k/n of a random source of blocklength n and of min-entropy k (basically the number of random bits) the *entropy rate*.

In complexity theory, randomness extraction from a distribution has been extensively studied (see [41] for a survey). For certain random sources, it has been shown that it is impossible to extract even one bit of randomness [39]. One way to solve this last problem is to use a small number of uniformly random bits as a *catalyst* in addition to the bits from the weak random source as in the LHL as said in [34]. However, in some cases, we can eliminate the need for the random catalyst by restricting the class of weak random sources. Trevisan and Vadhan and later Dodis [48, 21] have called such functions *deterministic extractors*. In cryptography, randomness extractors have been studied under different adversaries to construct truly random generators [5], and deterministic extractors have been used to built All-Or-Nothing-Transform (AONTs) schemes and Exposure-Resilient Function (ERF) [17, 23].

In the key exchange setting, the problem is to transform the randomness in a uniform source of small entropy rate into a source of entropy rate 1. For example, under the DDH assumption in a 160-bit prime order q subgroup in $\mathbb{Z}_p^\star$, we know that the input random source (in a DH-based key exchange protocol) has 160 bits of min-entropy. So, for 1024-bit prime p, the entropy rate of the initial source is 160/1024. Because of the specific structure of the source, deterministic extractors (which exploit the algebraic structure) may be used to derive cryptographic keys, they would avoid all the problems we presented before. We will thus introduce a new deterministic extractor, which takes advantage of the specific structure of elliptic curves.

1.5 Contribution and Organization

In this paper, we first focus on inaccuracies in the security analysis of Σ_0, and in the analysis/design of IKE, on the basis of the original documents [36, 28, 37]. We stress one more time that even if there is not yet any complete proof in the standard model, the IKE protocols have been carefully designed and there is no attack against them, just theoretical problems. The first one is about the theoretical analysis of Σ_0. The second one is about the actual use of HMAC in IKEv1, but at a theoretical level only. It has been introduced in the signature mode, for compatibility reasons with other authentication modes, in the IKE standard. Fortunately, it has been corrected in IKEv2 [35, 30] mainly by Krawczyk.

Then, we discuss in more details on various techniques to derive a uniformly distributed bit-string from a high-entropy bit-string source. More specifically, we apply the Kaliski's technique [33], with quadratic twists of elliptic curves, to this framework. It is quite well-suited to authenticated key exchange, since it already works on cyclic groups. Therefore, it is more efficient than the Leftover Hash Lemma while retaining the same security attributes, and being *deterministic*. This 'Twist AUgmented' (TAU) technique is provably secure assuming only the intractability of the decisional Diffie-Hellman problem on elliptic curves.

Even thought quadratic twists were previously introduced in the literature, their practical aspects were not fully studied [15, 16]. We thus also show that appropriate curves can be easily generated.

2 Security Problem in Security Proof of Σ_0

In this section, we precisely focus on the security proof of the basic SIGMA protocol, named Σ_0 in [19]. The description of this protocol[2] is given in figure 1.

[2] As suggested in [19], we enhanced it with a fourth flow, which is an acknowledgment from the receiver which means that everything is consistent to him. Without such a flow, it would be easy to break the usual mutual authentication notion, by simply not relaying the third flow. For the semantic security, it just helps in the proof, since active attacks can then be easily detected.

Alice		Bob
	Common group : $\mathbb{G} = \langle g \rangle$ of prime order q	
Signing Key : sk_A Verification Key : pk_A $\mathsf{accept} \leftarrow \mathsf{false}; \mathsf{terminate} \leftarrow \mathsf{false}$		Signing Key : sk_B Verification Key : pk_B $\mathsf{accept} \leftarrow \mathsf{false}; \mathsf{terminate} \leftarrow \mathsf{false}$
$s \overset{R}{\leftarrow} \{0,1\}^\star, x \overset{R}{\leftarrow} \mathbb{Z}_q, X = g^x$	$\xrightarrow{s, X}$	$y \overset{R}{\leftarrow} \mathbb{Z}_q, Y = g^y$ $\sigma_B = \mathsf{SIG.Sign}(\mathsf{sk}_B; (\text{"1"}, s, X, Y))$ $K = X^y, k_m = \mathsf{MacKey}(K)$
$K = Y^x, k_m = \mathsf{MacKey}(K)$ Check μ_B, σ_B $\sigma_A = \mathsf{SIG.Sign}(\mathsf{sk}_A; (\text{"0"}, s, Y, X))$ $\mu_A = \mathsf{MAC.Sign}(k_m; (\text{"0"}, s, \mathit{Alice}))$	$\xleftarrow{s, Y, \mathit{Bob}, \sigma_B, \mu_B}$	$\mu_B = \mathsf{MAC.Sign}(k_m; (\text{"1"}, s, \mathit{Bob}))$
$\mathsf{accept} \leftarrow \mathsf{true}$	$\xrightarrow{s, \mathit{Alice}, \sigma_A, \mu_A}$	Check μ_A, σ_A $\mathsf{accept} \leftarrow \mathsf{true}$
wait for a valid μ_C $\mathsf{terminate} \leftarrow \mathsf{true}$	$\xleftarrow{\mu_C}$	$\mu_C = \mathsf{MAC.Sign}(k_m; \mathsf{Ack})$ $\mathsf{terminate} \leftarrow \mathsf{true}$

$\mathsf{sk} = \mathsf{SessionKey}(K)$
$\mathsf{sid} = s, X, Y, \mathit{Alice}, \mathit{Bob}, \sigma_A, \sigma_B, \mu_A, \mu_B$

where $\mathsf{SessionKey}(K) = \mathsf{PRF}_K(0), \mathsf{MacKey}(K) = \mathsf{PRF}_K(1)$
and when a check fails, one stops the execution: $\mathsf{terminate} \leftarrow \mathsf{true}$

Fig. 1. An honest execution of the protocol Σ_0.

2.1 Security Proof of Σ_0 [19]

We only focus on the semantic security (privacy of the session key) of a key exchange protocol and we quickly explain the adversarial game. A passive adversary is allowed to see passive executions between two honest participants of his choice and to see old session keys. His goal is to distinguish a random key from the real session key in an execution of the protocol for which he cannot ask to see the real session keys to any partner. Moreover, if active adversaries have to be taken into account, the adversary can inject in the protocol his own messages, and receives the corresponding answer according to the protocol. The security goal is the same. More details are given in the appendix A.

A security proof consists in showing that the adversary advantage in winning the security game is negligible. To this end, the general technique is to describe a polynomial sequence of hybrid games [45]. The first one is usually the real security game while the last one is a security game where the advantage of the adversary can "easily" be proved to be zero. Finally, it remains to show that the statistical or computational distance between each pair of sequential games is negligible.

Let us recall the definition of pseudo-random functions according to Goldreich [26] since this will be a crucial tool (other more classical primitives and assumptions are reviewed in appendix B.)

Definition 1 (Pseudo-Random Functions). A pseudo-random function family (PRF) is a family of functions $\mathcal{F} = (f_k)_k$ in $\mathcal{F}_{n,m}$, the set of the functions from $\{0,1\}^n$ into $\{0,1\}^m$, indexed by a key $k \in \{0,1\}^\ell$, so that for a randomly chosen ℓ-bit string key k, no adversary can distinguish the function f_k from a truly random function in $\mathcal{F}_{n,m}$: $\mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(\mathcal{D}, q) = |\Pr_k[1 \leftarrow \mathcal{D}^{f_k}] - \Pr_f[1 \leftarrow \mathcal{D}^f]|$ must be small, where $\mathcal{D}$ is a distinguisher, with an oracle access to either a random instance f_k in the given family $\mathcal{F}$ or a truly random function f in $\mathcal{F}_{n,m}$, and must distinguish the two cases with at most q queries to the function oracle. We say that such a family is a (q, ε, t)-PRF if for any distinguisher asking at most q queries to the oracle, its advantage is less than ε, after a running time bounded by t.

Let us now review the proof of semantic security Σ_0, depicted on Figure 1. While [19] is a theoretical paper, assumptions in the security claim and the security proof remain quite informal since at some place one can find "*we will assume the security of the other underlying cryptographic primitives in the protocol (...) under the standard security notions in the cryptographic literature*", and "*We use the notation random() to represent a random (and independent) choice of a string of some appropriate length*". The latter statement allows proofs in the standard model, *i.e.* without making any unrealistic assumption, such as in the random oracle model [10]. However, the actual security result relies on ***non-standard security notions in the cryptographic literature*** contrary to the former statement. The main problem comes from the PRF which is usually assumed to be indexed by keys uniformly distributed in $\{0,1\}^\ell$, for some length ℓ (as reviewed above).

The security proof [19] uses the following hybrid sequence of games where we only define the difference with the original security game REAL. In fact, the security games are identical, except the generation of some keys which varies between the games.

1. REAL: $k_m = \mathsf{MacKey}(K) = \mathsf{PRF}_{g^{xy}}(1)$ and $\mathsf{sk} = \mathsf{SessionKey}(K) = \mathsf{PRF}_{g^{xy}}(0)$
2. RPRF: $k_m = \mathsf{PRF}_K(1)$ and $\mathsf{sk} = \mathsf{PRF}_K(0)$, for $K \leftarrow random_1()$
3. ALLR: $k_m \leftarrow random_2()$ and $\mathsf{sk} \leftarrow random_2()$
4. HYBR: $k_m = \mathsf{PRF}_K(1)$ for $K \leftarrow random_1()$, and $\mathsf{sk} \leftarrow random_2()$
5. RAND: $k_m = \mathsf{PRF}_{g^{xy}}(1)$ and $\mathsf{sk} \leftarrow random_2()$

However, in the security proof of Canetti and Krawczyk [19], we can show that under classical assumptions, the distance between some games cannot be negligible, but even close to 1 and consequently *no* security can hold in the standard model. More precisely, we show that we can construct PRF for which the scheme becomes insecure while the authors of [19] claims security for any PRF.

2.2 Informal Evidence that $\boldsymbol{\Sigma_0}$ (without Randomness Extraction) is Insecure

In the following, we present a distance between game REAL and game ALLR, but the same gap appears between games ALLR and RAND. The REAL game is the game the adversary actually plays, while in the RAND game, it is clear that its advantage is exactly zero since the session key is random and independent of the challenger bit. According to the "appropriate" range of the random generation $random_1()$, either the gap from RPRF to ALLR or from REAL to RPRF may not be indistinguishable.

Case 1: *random*$_1$() outputs random elements in $\{0,1\}^\ell$. In this first case, RPRF and ALLR are computationally indistinguishable under the assumption that $\mathcal{F}$ is a PRF in $\mathcal{F}_{n,m}$ for $k \stackrel{R}{\leftarrow} \{0,1\}^\ell$. However, REAL and RPRF are not indistinguishable with the above parameters: in the REAL game, $k = g^{xy} \in \mathbb{G}$, while in the RPRF game, $k \stackrel{R}{\leftarrow} \{0,1\}^\ell$. The latter is in $\mathbb{G}$ with negligible probability, and thus the Decisional Diffie-Hellman assumption is not enough.

Case 2: *random*$_1$() outputs random elements in $\mathbb{G}$. In this case, the two first games REAL and RPRF are computationally indistinguishable under the assumption of the intractability of the decisional Diffie-Hellman problem. However, RPRF and ALLR are not indistinguishable using classical pseudo-random functions, where the keys (the indices) must be uniformly distributed ℓ-bit strings to provide "almost-random" functions as we will illustrate in the following subsection.

2.3 Formal Evidence that $\boldsymbol{\Sigma_0}$ (without Randomness Extraction) is Insecure

The following theorem contradicts the main theorem 6 from [19]:

Theorem 2. *There exists PRF for which Σ_0 (without any randomness extraction) can be broken.*

Proof. Let us show with the following easy **counter-example** that the above standard assumption about pseudo-random functions is not enough here: let us assume, as suggested in [19], that $\mathbb{G}$ is the subgroup of prime order q in $\mathbb{Z}_p^\star$, where $q|p-1$. This usually means that p is an ℓ-bit prime, with $\ell \geq 1024$, while q is 160-bit long. For clarity, let us assume that $2^{\ell-1} < p < 2^\ell$. Let $\mathcal{F}$ be a PRF in $\mathcal{F}_{n,m}$, for k randomly drawn from $\{0,1\}^\ell$. Let us consider the following family of functions $\mathcal{G} = (g_k)_k$: if $k \in \mathbb{G}$, then $g_k(x) = 0^m$ for any $x \in \{0,1\}^n$, else $g_k = f_k$. Since we modify a negligible subfamily of $\mathcal{F}$, the family $\mathcal{G}$ is also a PRF for k in $\{0,1\}^\ell$ or k in $\mathbb{Z}_p^\star$, but clearly not for k in $\mathbb{G} = \{k \in \{0,1\}^\ell \mid k^q = 1 \bmod p\}$.

Even a passive adversary can learn the session keys, since they are all equal to 0^m. Indeed, the master secret is always a Diffie-Hellman value, and thus in $\mathbb{G}$. □

The same argument prohibits the use of the ElGamal encryption [24] for messages which do not lie in the group. The conclusion of this section is that new tools are needed to fill the theoretical gap. The designers of SIGMA/IKE were aware of that, as explained in the appendices of the full version of [37]. It has also been discussed in the IETF mailing list [30]. However, one can explicitly read in [19] the two following statements. First, in the notes right after the description of the protocol Σ_0, one reads

> In practice, it is recommended not to use the plain value g^{xy} of the DH key but a hashed value $H(g^{xy})$ where H is a hash function (e.g. a cryptographic hash function such as SHA or a universal hash function, etc.). This has the effect of producing a number of bits as required to key the PRF, and (depending on the properties of the hash function) may also help to "extracting the security entropy" from the g^{xy} output. **If the plain g^{xy} is used, our security results hold under the DDH assumption.** Using a hashed value of g^{xy} is secure under the (possibly weaker) HDH assumption.

It then refers to the subsection 5.6, where we can find

> We mentioned in Section 3 [the above note] that it is **advisable** in practice to hash the DH value g^{xy} to the length of the PRF's key from which further keys are derived. In particular, **this may result in better security** of the resultant hashed bits relative to the initial plain string g^{xy}. **The use of all of g^{xy} as if they were all perfectly random is generally justified by the DDH assumption.** However, while this assumption is considered "standard" these days, it actually constitutes a very strong conjecture about the strength of the DH key g^{xy}: namely that *all* bits in this key are simultaneously indistinguishable from random for an observer of g^x and g^y.

Contrary to these claims, the DDH assumption does not say that **all bits in this key are simultaneously indistinguishable from random**, but just that the min-entropy is of $\log q$, where q is the prime order of the group. Our first goal in this paper is thus to insist on this usual simplification. The practical solution is thus the use of a randomness extractor, and namely the use of the Leftover Hash Lemma [32] with (almost) universal hash functions [32, 22].

3 Towards a Justification of IKE

Contrary to the above Σ_0 protocol, IKE indeed applies a preliminary transformation randomness extraction) to the master secret, also denoted `prf`. It furthermore suggests to implement `prf` with HMAC. Dodis *et al.* [22] analyzed usual MACs, namely the CBC-MAC and HMAC, and iterated hash functions (Cascade) as randomness extractors (RE). In fact, they prove that NMAC is a good randomness extractor. But, what does happen if a PRF is used. We show here that some PRF are good RE. It is worthwhile to say here that as noted with an example in [22], not all PRF are good RE. Moreover, we recall some results in order to better explain the gap between theory and practice in the following section.

3.1 Almost Universal Hash Functions and the Leftover Hash Lemma

More precisely, in [22] the authors proved the property of good randomness extractors for several constructions under various kinds of assumptions, and namely the universal hashing property of the compression functions. Under such an assumption, they manage to prove that the global construction provides an almost universal hash function family (AUH).

Definition 3 (Statistical Distance). If $\mathcal{D}$ is a distribution over some finite set S and $s \in S$, then we denote by $\mathcal{D}(s)$ the probability of s according to $\mathcal{D}$ and similarly, if $X \subseteq S$, then $\mathcal{D}(X) = \sum_{s \in X} \mathcal{D}(s)$.

Let $\mathcal{D}_1, \mathcal{D}_2$ be two distributions over some finite set S. The statistical distance between $\mathcal{D}_1, \mathcal{D}_2$, is defined as

$$|\mathcal{D}_1 - \mathcal{D}_2| = \frac{1}{2} \cdot \sum_{s \in S} |\mathcal{D}_1(s) - \mathcal{D}_2(s)| = \max_{X \subseteq S} |\mathcal{D}_1(X) - \mathcal{D}_2(X)|.$$

We say that a random variable X on S is δ-**uniform** if the statistical distance between X and the uniform distribution on S is equal to δ.

Definition 4 (Almost Universal Hash Functions). Let $\mathcal{H} = (h_k)_k$ be a family of functions in $\mathcal{F}_{n,m}$, the set of the functions from $\{0,1\}^n$ into $\{0,1\}^m$, indexed by a key $k \in \{0,1\}^\ell$. We say that $\mathcal{H}$ is a δ-almost universal hash (δ-AUH) function family if

$$\text{for any } x, y \in \{0,1\}^n, x \neq y, \Pr_k[h_k(x) = h_k(y)] \leq \delta.$$

Note that such a family is called a universal hash function family if $\delta = 1/2^m$. Consequently, we set $\delta = 1/2^m + \varepsilon$.

Definition 5 (Min and Renyi Entropy [44]). Let X be a random variable taking values on a finite set $\mathcal{S}$. We define the **guessing probability** $\gamma(X)$ of X and the **collision probability** $\kappa(X)$ of X as $\gamma(X) = \max\{\Pr[X = s] : s \in \mathcal{S}\}$ and $\kappa(X) = \sum_{s \in \mathcal{S}} \Pr[X = s]^2$. The **min entropy** of X is $\mathbf{H}_\infty = \log_2(1/\gamma(X))$ while the **Renyi entropy** is $\mathbf{H}_2 = \log_2(1/\kappa(X))$, and we have the following inequality $\gamma(X)^2 \leq \kappa(X) \leq \gamma(X)$.

The Leftover Hash Lemma [32, 31] provides the most well-known *probabilistic randomness extractor*:

Lemma 6 ([32]). *Let $\mathcal{D}$ be a probabilistic distribution over $\{0,1\}^n$ with min-entropy at least σ. Let e be an integer and $m = \sigma - 2e$. Let $\mathcal{H} = \{h_k\}_k$, with $h_k \in \mathcal{F}_{n,m}$ for any $k \in \{0,1\}^\ell$, be an almost universal hash function family. Let H be a random variable uniformly distributed on $\mathcal{H}$, X denotes a random variable taking value in $\{0,1\}^n$, and H, X are independent. Then, $(H, H(X))$ is $2^{-(e+1)}$-uniform on $\mathcal{H} \times \{0,1\}^m$.*

Impagliazzo and Zuckerman in [32] prove the lemma with almost universal hash function where $\varepsilon = 1/2^n$. In [22], it is proved for any δ-almost universal hash function family for $\varepsilon \ll 1/2^m$. See also [44] for a proof. Therefore, combined with the analysis of NMAC as a δ-AUH function, this may justify the design of IKE when HMAC is used under a specific assumption on the independence of the two keys in NMAC. We show in the following that the same result holds for some PRF provided ε be taken into account to estimate the size of the output. However, we begin to prove a slight generalization of the LHL, similar to [22].

Lemma 7 (LHL with $(1/2^m + \varepsilon)$-AUH). *Let $\mathcal{D}$ be a probabilistic distribution over $\{0,1\}^n$ with min-entropy at least σ. Let e be an integer and $m \leq \alpha - 2e$ where $\alpha = \min(\sigma, \log_2(1/\varepsilon))$. Let $\mathcal{H} = \{h_k\}_k$, with $h_k \in \mathcal{F}_{n,m}$ for any $k \in \{0,1\}^\ell$, be a $(1/2^m + \varepsilon)$-almost universal hash function family. Let H be a random variable uniformly distributed on $\mathcal{H}$, X denotes a random variable taking value in $\{0,1\}^n$, and H, X are independent. Then, $(H, H(X))$ is 2^{-e}-uniform on $\mathcal{H} \times \{0,1\}^m$.*

Proof. The proof relies on two claims.

Claim ([44]). Let X be a random variable distributed according a distribution $\mathcal{D}$, taking values on the finite set S and of collision probability $\kappa = \kappa(X)$. If X is δ-uniform on S, then $\kappa \geq (1 + 4\delta^2)/|S|$.

The second claim studies the collision probability $\kappa = \kappa(H, H(X))$ where H denotes a random variable with uniform probability on $\mathcal{H}$, X denotes a random variable on the set $\{0,1\}^n$, and H and X are independent. We can easily adapt the proof of [44] to prove that the statistical distance between the distribution of $(H, H(X))$ and the uniform distribution on $\mathcal{H} \times \{0,1\}^m$ is δ, which is at most $(1/2) \cdot \sqrt{2^m \cdot (\kappa + \varepsilon)}$. So it can be upper-bounded by $(1/2) \cdot \sqrt{2^m \cdot (2^{-\sigma} + \varepsilon)}$, since the collision probability κ is less than the guessing probability γ as noted in definition 5.

If we denote by $\alpha = \min(\sigma, \log_2(1/\varepsilon))$, then we can upper-bound δ by $(1/2) \cdot \sqrt{2^m \cdot 2 \cdot 2^{-\alpha}}$ and so if we want a bias of 2^{-e} we need $m \leq \alpha - 2e$. □

Remark 8. This requires $\varepsilon \ll 1/2^m$ as it is observed in [22], but $\varepsilon \leq 1/2^{m+2e}$ is enough. Anyway, this definitely excludes functions families where the key-length is the same as the output-length (as compression functions).

3.2 Pseudo-Random Functions, Almost Universal Hash Functions and Randomness Extractors

We have already discussed the practical meaning of the universal hashing property for compression functions. However, the design of IKE [28, 35] uses the acronym `prf` at several places, for different purposes: randomness extractors and actual PRF. However, we recall here the *crucial* difference between pseudo-random function and randomness extraction: the former uses random *secret* keys, while the latter uses random *but known* keys. We thus show below that the *strong* assumption of PRF implies the almost universal hashing property. Therefore, the Leftover Hash Lemma 7 applied with some PRF (namely keyed with uniform random bit-strings and with advantage sufficiently small) provides a good randomness extractor.

Theorem 9. *If a family of functions $\mathcal{F}$ is a $(2, \varepsilon, 2T_f)$-PRF in $\mathcal{F}_{n,m}$, then it is a $(1/2^m + \varepsilon)$-AUH function family, where T_f denotes the maximal time to evaluate an instance of $\mathcal{F}$ on any $x \in \{0,1\}^n$.*

Proof. We want to show that if an universal hash function family is not $(1/2^m + \varepsilon)-$secure, *i.e.* there exist x, y such that $\Pr_k[h_k(x) = h_k(y)] > 1/2^m + \varepsilon$, then we can find an adversary against a PRF with advantage at least ε.

Let us consider the following family of distinguishers, $\mathcal{D}_{x,y}$ for each pair (x, y) of elements in $\{0,1\}^n$. The distinguisher $\mathcal{D}_{x,y}$ queries the oracle (either f_k for a random k or a random function) to get $X = f(x)$ and $Y = f(y)$, and simply answers 1 if $X = Y$ and 0 otherwise.

Suppose that $\mathcal{F}$ is not a $(1/2^m + \varepsilon)$-AUH function family. It means there exists a pair (x, y) for which $\Pr_k[f_k(x) = f_k(y)] > 1/2^m + \varepsilon$. Let us consider the advantage of the corresponding distinguisher $\mathcal{D}_{x,y}$: if f is a truly random function in $\mathcal{F}_{n,m}$, the set of all functions from $\{0,1\}^n$ to $\{0,1\}^m$, then $\Pr[\mathcal{D}_{x,y} = 1] = 1/2^m$; if f is a randomly chosen f_k in $\mathcal{F}$, then $\Pr[\mathcal{D}_{x,y} = 1] > 1/2^m + \varepsilon$. As a consequence, the advantage of $\mathcal{D}_{x,y}$ is not less than ε, which is in contradiction with the above PRF property. □

Therefore, we have the following corollary by combining lemma 7 with the previous theorem.

Corollary 10. *Let $\mathcal{F}$ be a family of functions in $\mathcal{F}_{n,m}$, and T_f denote the maximal time to evaluate an instance of $\mathcal{F}$ on any $x \in \{0,1\}^n$. If $\mathcal{F}$ is a $(2, \varepsilon, 2T_f)$-PRF, when applied on a random source with min-entropy at least σ, then it is a good randomness extractor, of bias bounded by $1/2^e$, as soon as*

$$m \leq \min(\sigma, \log_2(1/\varepsilon)) - 2e.$$

Remark 11. This result is not in contradiction with the example described in [22], since if $\varepsilon = 1/2^m$ with m bits of output, then clearly $\min(\sigma, \log_2(1/\varepsilon)) \leq m$. The above corollary just claims that the bias is less than 1. As a consequence, we cannot extract m bits.

4 Mismatches between Theory and Practice

The above arguments all together seem to constitute a strong justification of the design and the security of IKE as it fills the gap from the security proof of Σ_0 (without randomness extraction) with the use of HMAC, as a randomness extractor. However, let take a closer look at the actual use of `prf` in IKEv1 [28]. We stress again that these mismatches do not endanger the use of IKE and are only of theoretical interest.

4.1 Security Inaccuracy in the Randomness Extraction Phase of IKE

Informal Evidence that NO Security Proof of IKE can be Simply Based on the LHL. In IKE (in the signature mode), the master key `SKEYSEED` (which should be a pseudo-uniformly distributed bit-string) is derived with the above δ-AUH construction, `SKEYSEED = prf(Ni | Nr, g^ir)`, where `Ni` and `Nr` are random nonces *chosen* by the participants, and `g^ir` is the common Diffie-Hellman secret.

More precisely, the random nonces specifying the choice of the function in the δ-AUH function family are chosen by the players and never signed: the adversary could edit one of the nonces—`Ni` or `Nr`—and thus have some control over the value `SKEYSEED`.

Indeed, the security results provided in [22] guarantee the quality of `SKEYSEED` if one uses **random** *but known* keys. In the specification of IKEv1, the adversary has some control over this random key (which is no longer the case in IKEv2.)

Formal Evidence of the Security Level of the LHL with Partially Random Keys. In order to have a good picture of that, let us first apply the LHL with a *perfect* universal hash function family. The linear congruential hashing is a well-known method: it works in $\mathbb{F}_{2^n}$ in order to output n-bit strings, which are thereafter truncated to the m-rightmost bits. More precisely, the key is a pair $(a, b) \in \mathbb{F}_{2^n}^2$, and on the input $x \in \{0,1\}^n$, the function $h_{a,b}$ outputs $\mathsf{Trunc}_m(ax + b)$. This family of 2^{2n} functions is a universal hash function family in $\mathcal{F}_{n,m}$. And thus, the above LHL applies perfectly to extract random bits.

Theorem 12. *There exist universal hash functions and pseudo-random functions which lead to insecure randomness extractors, when the key is only partially random.*

Proof. Let us consider the above linear congruential hash. By making $a = 0$, the final key is equal to b. Since the latter is public, the adversary knows the final key.

Maybe, a PRF (as required by IKE) would avoid this weakness. But let us exhibit another counter-example, starting from any PRF $\mathcal{F} = (f_k)_k$ (possibly the above $\mathcal{G}$ one, defined in the previous counter-example), with ℓ-bit keys, where $\ell = 2\lambda$. Let us define the new family $\mathcal{H} = (h_{a||b})_{a|b}$, by $h_{a||b}(x) = f_{a||b}(x)$ if $a \neq 0^\lambda$, and $h_{0^\lambda||b}(x) = 0^n$. We again modify a negligible fraction of $\mathcal{F}$, and thus $\mathcal{H}$ is still a PRF. However, with the control of half of the bits, the adversary can make the master key to be always 0^n: all the derived keys are thus known to him. □

In IKEv1, the nonces are not signed, but some MAC are applied later which makes the above counter-example irrelevant. However, some other counter-examples exist:

$$h_{(a_0,a_1),(b_0,b_1)}(x) = \mathsf{Trunc}_m(a_0 b_0 x + a_1 b_1), \text{with } \mathtt{Ni} = (a_0, a_1) \text{ and } \mathtt{Nr} = (b_0, b_1), \text{ in } \mathbb{F}_{2^n}.$$

This result clearly shows that the security of IKEv1 cannot only rely on the general definition of universal hash functions or pseudo-random functions.

4.2 Countermeasure: Certified Keys

For the IKE application, a much more simple alternative consists in signing the nonces. It is almost at no cost to include the nonces in the values to be signed by the users. This was proposed in SIGMA, and included in IKEv2.

Another solution to cope with the randomness extraction error is as noticed by Shoup [44] and also by Barak *et al.* in [5] to use the same "certified key" or the same hard-coded key in the software. Indeed, they suggest an extension of the LHL which allows the derivation of many random bit-strings with a *unique random* key, and thus a *public and fixed* hash function. However, the quality of the extracted randomness decreases linearly with the number of extractions – due to the hybrid technique. Nevertheless, this is often the unique solution.

5 Deterministic Randomness Extractors

Other alternatives to the LHL are also available, namely when no certification is available, as in the password-based setting, by using deterministic randomness extractors. Several of them exist in the literature and have already been employed by standardization bodies to convert a random element of a group into a random bit-string as in [42].

5.1 Hash-Diffie-Hellman

The first one, from a practical point of view, is the use of a hash function. In the random oracle model [10], this gives a perfect random bit-string, under the so-called computational Diffie-Hellman assumption. In the standard model, a weaker assumption has been defined, the Hash Diffie-Hellman assumption [1, 25]. But this assumption is, in some sense, the assumption that a hash function is perfectly suited to this goal, while this is not the applications that designers of hash functions have in mind. Everybody may agree on the practical validity of such a construction, but it definitely requires non-standard assumptions, from a theoretical point of view.

5.2 A Simple Deterministic Extractor

Basically, what we want is an extractor of the entropy from a random (uniformly distributed) element in a cyclic group $\mathbb{G}$ of order q. A bijection from $\mathbb{G}$ to $\mathbb{Z}_q$ would do the job, since it would transfer the uniform distribution $\mathbb{G}$ into a uniform distribution in $\mathbb{Z}_q$ (an appropriate choice for q thereafter allows the truncation to the $\log q$-rightmost bits to get a uniformly distributed bit-string).

Theorem 13. *There is an efficient bijection from a subgroup $\mathbb{G}$ of prime order q in $\mathbb{Z}_p^\star$ to $\mathbb{Z}_q$, when $p = 2q + 1$.*

Proof. Let us use a finite field $\mathbb{Z}_p$, with $p = 2q+1$ (a safe prime) and work in the cyclic group of order q: the group $\mathbb{G}$ of the quadratic residues modulo p. Since $p = 3 \bmod 4$, this is a Blum prime, and thus -1 does not lie in $\mathbb{G}$.

We can define the following extractor, for any $x \in \mathbb{G}$: if $y \leq q$, then $f(y) = f_1(y) = y$, else $f(y) = f_2(y) = p - y$. Since -1 is not in $\mathbb{G}$, and $p - y = -y = (-1) \times y \bmod p$, f_1 maps $\mathbb{G}$ to $\mathbb{G}$ (the identity function) and f_2 maps $\mathbb{G}$ to $\mathbb{Z}_p \backslash \mathbb{G}$. Therefore, f is an injective mapping and for $y \in \mathbb{G}$, $f_1(y), f_2(y)$ are in $\mathbb{Z}_q$. A simple counting argument proves that this is a bijection. □

The following lemma analyzes the security when truncation is used in order to get ℓ bits uniformly distributed. We assume that the distance between q and the smaller and closest power of two is not too large, say $2^{\ell/2}$ if we want to extract ℓ bits. The proof of the lemma is done in appendix F.

Lemma 14. *Let us denote by $\mathcal{U}_q$ the uniform distribution on the space $\mathbb{Z}_q$ and by $\mathcal{U}_{2^\ell}$ the uniform distribution on the space $\{0,1\}^\ell \sim \{0,\ldots,2^\ell-1\}$. Then the statistical distance is bounded by $2/\sqrt{2^\ell}$.*

Corollary 15. *Since the statistical distance between the uniform distribution on $\mathcal{U}_\ell$ and $\mathsf{Trunc}_\ell(f(\mathbb{G}))$, where f is the previous bijection, $|q| = \ell$ and $|q - 2^\ell| \leq 2^{\ell/2}$, is upper bounded by $2/\sqrt{2^\ell}$ according to lemma 14, the mapping $\text{TRUNC}_\ell \circ f$ is a good deterministic randomness extractor.*

Therefore, the truncation of f gives a deterministic randomness extractor from $\mathbb{G}$ onto $\mathbb{Z}_q$. However, this requires the use of a safe prime, and thus quite large exponents: 1023 bits instead of 160 bits, which makes any Diffie-Hellman key agreement unpractical. In the next subsection, we describe an efficient deterministic randomness extraction.

5.3 The 'Twist-AUgmented' Technique

In the early 90's Kaliski [33] used elliptic curves and their twists for making a random permutation from a random function. This construction can be used to make a uniform distribution in $\mathbb{Z}_{2q}$ from points uniformly distributed on a curve or its quadratic twist, both on the finite field $\mathbb{F}_q$. More recently, quadratic twists have also been used in the context of password-authenticated key exchange [16]. The goal was to make the Bellovin et al.'s encrypted key exchange protocol [6] immune to partition attacks but did not explain how to specify the key-derivation function. It has also been applied to the context of public-key encryption [15].

We can take advantage of elliptic curves and their quadratic twists, as done by Kaliski [33], to come up with a technique not requiring stronger assumptions. This technique, called 'Twist-AUgmented' (TAU), uses the fact that a random point on a curve over $\mathbb{F}_p$ has an abscissa uniformly distributed in a set E and that a random point over its twisted has an abscissa uniformly distributed in the set $\tilde{E}$ as well, *i.e.* it is the complementary set of E in $\mathbb{F}_p$. Therefore by choosing one of the two abscissae at random, we will get an element almost uniformly distributed in $\mathbb{F}_p$. For well-chosen fields, we thus efficiently get an almost uniformly distributed bit-string, which may be 256-bit long: it is enough to derive two keys (for privacy and for authentication) without any pseudo-random function by simply splitting this bit-string.

Quadratic Twist of an Elliptic Curve. Let $p > 3$ be a prime number. An elliptic curve is a set of points $\mathbb{E} = \mathbb{E}_{a,b} = \{(x, y) : y^2 = x^3 + ax + b\} \cup \{\infty_\mathbb{E}\}$, where a and b are elements of $\mathbb{F}_p$ and $\infty_\mathbb{E}$ is a symbol for the point at infinity. It is well known that an elliptic curve $\mathbb{E}$ can be equipped with a group law − the so-called chord and tangent group law − such that the computational and decisional Diffie-Hellman problems are believed to be hard problems in general.

Let c be a quadratic non-residue in $\mathbb{F}_p$, and define the **quadratic twist** of $\mathbb{E}_{a,b}$ to be the curve given by the following equation: $\tilde{\mathbb{E}}_{a,b} = \{(x, y) : cy^2 = x^3 + ax + b\} \cup \{\infty_{\tilde{\mathbb{E}}}\}$.

The change of variables $x' = cx$ and $y' = c^2 y$ transforms the equation of $\tilde{\mathbb{E}}_{a,b}$ into $y'^2 = x'^3 + ac^2x' + bc^3$. This demonstrates that $\tilde{\mathbb{E}}_{a,b}$ is isomorphic to an elliptic curve and can therefore be equipped with a group law. The main interest of the introduction of the quadratic twist here follows directly from the definition: if x is not the abscissa of a point of $\mathbb{E}_{a,b}$, then $x^3 + ax + b$ is not a square in $\mathbb{F}_p$ and therefore $(x^3 + ax + b)/c$ is a square in $\mathbb{F}_p$. Then it is the abscissa of a point of $\tilde{\mathbb{E}}_{a,b}$. The converse is also true.

Cardinalities. Hasse-Weil's theorem gives good bound on the group order of an elliptic curve [46]. Let us write $q = \#\mathbb{E} = p + 1 - t$, then we have $|t| < 2\sqrt{p}$. We could apply the same result to $\tilde{\mathbb{E}}$, but in fact the number of points of a curve and its twist are far from being independent. Starting with the fact that a scalar is either a point on $\mathbb{E}$ or a point on $\tilde{\mathbb{E}}$, it is easy to derive that $\tilde{q} = \#\tilde{\mathbb{E}} = p + 1 + t$. For maximal security, it is desirable that the group orders are prime numbers. In particular, since p is odd, this implies that t is odd. Then both q and $\tilde{q}$ are odd.

Choice of the Prime Field. We have restricted ourselves to curves defined over prime fields. The notion of quadratic twist of an elliptic curve also exists for more general finite fields and in particular for fields of characteristic 2. However, they are of less interest in our context where we want to use the property that the abscissae of the points of the groups we are dealing with cover the whole finite field. In characteristic 2, all the non-super-singular curves have a group order that is divisible by (at least) 2. Hence keeping the covering property would imply to work with non-prime order groups. Even if it looks feasible to patch the protocol for that situation, it is certainly less elegant than using prime-order group with curves over prime fields.

To achieve our goal, we need that the abscissa of a point taken randomly in $\mathbb{E}$ or in $\tilde{\mathbb{E}}$ behaves like a random bit-string of length ℓ. Since all the elements of $\mathbb{F}_p$ are obtainable as abscissae of points of $\mathbb{E}$ and $\tilde{\mathbb{E}}$, we will be able to show that the random abscissa in $\mathbb{E}$ or $\tilde{\mathbb{E}}$ gives a random element in $\mathbb{F}_p$ (see Lemma 16 in Appendix F.) To convert this element to a bit-string of length ℓ without any further device and keeping the randomness unbiased, it is necessary to have p very close to 2^ℓ. Hence we propose to use a prime p which can be written $p = 2^\ell - \varepsilon$, where ε is an integer less than $2 \cdot 2^{\ell/2}$ (see Lemma 14.)

This extra-condition on p is not a practical inconvenience. In fact, the primes that are used in practice are almost always of this form, because they allow a faster arithmetic than more general primes. For instance, the curves proposed by the NIST are defined over finite field with primes which are often suitable to our case (the prime field, not the curves!).

TAU is a Good Randomness Extractor. Now, we show that the distribution of the master secret key K if we take it at random either on the curve $\mathbb{E}$ or $\tilde{\mathbb{E}}$ is statistically almost uniformly distributed on $\{0,1\}^\ell$. On the one hand, we prove that it is statistically indistinguishable from the uniform distribution on $\{0,\dots,p-1\}$ and then that the latter distribution is statistically indistinguishable from the uniform distribution on $\{0,1\}^\ell$ by using lemma 14 by replacing q by p. The proof of the following lemmas are done in appendix F.

Let us denote by $\mathcal{D}$ the distribution of K:

$$\mathcal{D} = \{b \stackrel{R}{\leftarrow} \{0,1\}, \mathbf{R}_0 \stackrel{R}{\leftarrow} \mathbb{E}, \mathbf{R}_1 \stackrel{R}{\leftarrow} \tilde{\mathbb{E}} : K = [\mathbf{R}_b]_{\mathsf{abs}}\} = \{b \stackrel{R}{\leftarrow} \{0,1\}, x_0 \stackrel{R}{\leftarrow} [\mathbb{E}]_{\mathsf{abs}}, x_1 \stackrel{R}{\leftarrow} [\tilde{\mathbb{E}}]_{\mathsf{abs}} : K = x_b\}.$$

Lemma 16. *The distribution $\mathcal{D}$ is statistically close to the uniform distribution $\mathcal{U}_p$ in $\mathbb{F}_p \sim \mathbb{Z}_p$:*

$$\delta = \sum_{x \in \mathbb{F}_p} \left| \Pr_{K \stackrel{R}{\leftarrow} \mathcal{U}_p}[K = x] - \Pr_{K \stackrel{R}{\leftarrow} \mathcal{D}}[K = x] \right| \leq \frac{2\sqrt{2}}{\sqrt{2^\ell}}.$$

Corollary 17. *The statistical distance between the uniform distribution on $\mathcal{U}_\ell$ and the* TAU *technique if $|p - 2^\ell| \leq 2^{\ell/2}$, is upper bounded by $2/\sqrt{2^\ell} + \frac{2\sqrt{2}}{\sqrt{2^\ell}}$ according to lemma 16 and 14.*

Finding a Suitable Elliptic Curve and Twist. The basic approach for construction a curve $\mathbb{E}$ over $\mathbb{F}_p$ such that both q and $\tilde{q}$ are primes is to pick random curves, count their cardinalities with the SEA algorithm, and keep only the good ones. With this strategy, if number of points were completely independent and behaved like random numbers in the Hasse-Weil interval, we would expect to have to have to build $O(\log^2 p)$ curves before finding a good one. If $\log p \approx 200$, it means that we have to run the SEA algorithm about 20000 times to construct a good curve, which is prohibitive.

Fortunately, the SEA algorithm [40] is suited for this kind of search, since it computes the order of $\mathbb{E}$ modulo small primes and recombine the group order by Chinese Remainder. Hence as soon as we know the order of $\mathbb{E}$ modulo a small prime ℓ, we abort the computation if this is zero. Furthermore, the group order of $\tilde{\mathbb{E}}$ modulo ℓ is readily deduced from $\#\mathbb{E} \bmod \ell$, and similar abortion can be played also with the twist. As a consequence, most of the curves are very quickly detected as bad curves, because either the curve of its twist has a non-prime group order.

In fact, the situation is more tricky, since the order of the curve and the order of its twist are not independent. For instance, imagine that $p \equiv 2 \bmod 3$, then the condition $\#\mathbb{E} \equiv 0 \bmod 3$ is equivalent to $t \equiv 0 \bmod 3$, which in turn is equivalent to $\#\tilde{\mathbb{E}} \equiv 0 \bmod 3$. A rigorous estimation of the running time of the SEA algorithm equipped with the early-abort strategy is out of the scope of this work. We just propose some numerical experiments to justify the claim that the construction of secure pairs of curve and twist is easily feasible on a reasonable computer.

We picked randomly about 30000 200-bit primes, and for each of them we picked a random curve and computed its cardinality and the cardinality of its twist. In the following table, we summarize the percentage of the curves for which both number of points are not divisible by all primes up to P_{max}.

P_{max}	1	2	3	5	7	11	13	17	19
remaining curves	100 %	33 %	12 %	7.2 %	4.9 %	3.9 %	3.3 %	3.0 %	2.7 %

From this data, we see that for 97.3 % of the curves, the SEA algorithm will be stopped at a very early stage, thus spending only a tiny fraction of the running time of the whole computation. With usual reasonable heuristics, it is expected that about 500 full computations are required on average before finding a good pair of curve and twist. A single SEA computation takes about 20 seconds for this size on a personal computer, hence in about 3 hours, we expect to build good parameters for a key-size of 200 bits. An example curve is given in Appendix G.

If there is a need to construct the curves in constraint environment, then it is probably a better idea to use the theory of Complex Multiplication. We will not give the details here, since the construction is well described both in the literature and in the standards. For our purpose, it suffices to choose a group order and a twisted group order which are both primes.

6 The 'Twist-AUgmented' Authenticated Diffie-Hellman Protocol

6.1 Description

Using the properties of 'Twist-AUgmented' deterministic randomness extractor, we then convert any Diffie-Hellman-like protocol, which provides a random element in a cyclic group, into a protocol which provides a random bit-string, without any additional assumptions. See figure 2 for the description.

6.2 Semantic Security

On Figure 2, we present the TAU-enhancement of Σ_0: some flows are doubled, on each curve. During his key-confirmation phase, Bob randomly chooses the curve which will be used for the Diffie-Hellman computation. This protocol achieves the property of semantic security under the elliptic-curve decisional Diffie-Hellman assumption and does not use ideal-hash functions. In order to prove this claim (the full proof is postpone to the appendix C) we consider games that have distances that can be measured easily. We use Shoup's lemma to bound the probability of events in successive games [43, 45]. The first game $\mathbf{G}_1$ allows us to avoid active attacks so that in the following games we only have to worry about replay attacks. Proving the claim boils down to coming up with the appropriate games $\mathbf{G}_2$ through $\mathbf{G}_5$. In these games we obtain a random master key K uniformly distributed in $\{0, \ldots, 2^\ell - 1\}$ which corrects the uncertainties of previous proofs. The games $\mathbf{G}_6$ and $\mathbf{G}_7$ are easy to come up with and therefore the proof of the claim easily follows. In the last game $\mathbf{G}_7$, the adversary has clearly no means to get any information about the random bit involved in the Test-query except to flip a coin.

Theorem 18. *For any adversary $\mathcal{A}$ running within time bound t, with less than q_s different sessions*

$$\mathsf{Adv}^{\mathsf{ake}}_{\mathsf{TAU}}(\mathcal{A}) \leq 4 \cdot \mathsf{Succ}^{\mathsf{euf-cma}}_{\mathsf{AUTH}}(t, q_s, q_s) + 2 \cdot \mathsf{Adv}^{\mathsf{ecddh}}_{\mathbf{P},\langle\mathbf{P}\rangle}(t') + 2 \cdot \mathsf{Adv}^{\mathsf{ecddh}}_{\mathbf{Q},\langle\mathbf{Q}\rangle}(t') + 2q_s\mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(t, 2) + \frac{10q_s}{\sqrt{2^\ell}},$$

where $t' \leq t + 8 \times q_s T_m$, and T_m is an upper-bound on the time to compute the multiplication of a point by a scalar.

Alice		Bob
Common twisted curves : $\mathbb{E} = \langle \mathbf{P} \rangle, \tilde{\mathbb{E}} = \langle \mathbf{Q} \rangle$ of respective prime orders $q, \tilde{q}$ over the finite field $\mathbb{F}_p$		
Signing Key : sk_A		Signing Key : sk_B
Verification Key : vk_A		Verification Key : vk_B
$\mathsf{accept} \leftarrow \mathsf{false}$		$\mathsf{accept} \leftarrow \mathsf{false}$
$\mathsf{terminate} \leftarrow \mathsf{false}$		$\mathsf{terminate} \leftarrow \mathsf{false}$
$s \stackrel{R}{\leftarrow} \{0,1\}^\star, x_0 \stackrel{R}{\leftarrow} \mathbb{Z}_q, X_0 = x_0 \cdot \mathbf{P}$		
$x_1 \stackrel{R}{\leftarrow} \mathbb{Z}_{\tilde{q}}, X_1 = x_1 \cdot \mathbf{Q}$		
$\sigma_A = \mathsf{AUTH.Sign}(\mathsf{sk}_A; (s, X_0, X_1))$	$\xrightarrow{Alice, s, X_0, X_1, \sigma_A}$	Check σ_A
		$y_0 \stackrel{R}{\leftarrow} \mathbb{Z}_q, Y_0 = y_0 \cdot \mathbf{P}$
		$y_1 \stackrel{R}{\leftarrow} \mathbb{Z}_{\tilde{q}}, Y_1 = y_1 \cdot \mathbf{Q}$
		$\beta \stackrel{R}{\leftarrow} \{0,1\}$
		$K_\beta = [y_\beta \cdot X_\beta]_{\mathsf{abs}}, k_m = \mathsf{MacKey}(K_\beta)$
		$\sigma_B = \mathsf{AUTH.Sign}(\mathsf{sk}_B; (s, X_0, X_1, Y_0, Y_1))$
	$\xleftarrow{Bob, s, Y_0, Y_1, \sigma_B, \mu_B}$	$\mu_B = \mathsf{MAC.Sign}(k_m; (\text{"1"}, s, Bob))$
$d \stackrel{R}{\leftarrow} \{0,1\}$		
$K = [x_d \cdot Y_d]_{\mathsf{abs}}, k_m = \mathsf{MacKey}(K)$		
Try to check μ_B : in case of failure $d = 1 - d, K = [x_d \cdot Y_d]_{\mathsf{abs}}$, $k_m = \mathsf{MacKey}(K)$		
Check μ_B, σ_B		
$\mu_A = \mathsf{MAC.Sign}(k_m, (\text{"0"}, s, Alice))$		
$\mathsf{accept} \leftarrow \mathsf{true}$	$\xrightarrow{s, \sigma_A, \mu_A}$	Check μ_A
		$\mathsf{accept} \leftarrow \mathsf{true}$
$\mathsf{terminate} \leftarrow \mathsf{true}$		$\mathsf{terminate} \leftarrow \mathsf{true}$

$\mathsf{sk} = \mathsf{SessionKey}(K)$
$\mathsf{sid} = s, Alice, Bob, X_0, X_1, Y_0, Y_1, \sigma_A, \sigma_B, \mu_A, \mu_B$

where $\mathsf{SessionKey}(K) = \mathsf{PRF}_K(0), \mathsf{MacKey}(K) = \mathsf{PRF}_K(1)$
$[\mathbf{R}]_{\mathsf{abs}}$ is the abscisse of the point $\mathbf{R}$ in $\mathbb{F}_p$
and when a check fails whithout being caught, one stops the execution: $\mathsf{terminate} \leftarrow \mathsf{true}$

Fig. 2. An honest execution of the 'Twist-AUgmented' Authenticated Diffie-Hellman protocol.

Conclusion

This paper shows that it is important to prove the security of a whole protocol and not to make the proof in several and independent works. Some gaps may indeed appear, unless theorems are applied, with clear assumptions and conclusions. OAEP is also a good example, where the semantic security against adaptive chosen-ciphertext attacks was widely assumed, without a global security proof, but only two unrelated results (OAEP is IND-CPA and PA94 [13], and IND-CPA combined with PA98 implies IND-CCA2 [9]), as noticed by Shoup in [43].

Finally, we want to stress that next versions of IKE should precisely refer to PRF or randomness extractor and not use the same acronym for the two different goals: randomness extraction and key derivation.

Acknowledgement

The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT. The first author is supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, Mathematical Information and Computing Sciences Division, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098. This document is report LBNL-54709. Disclaimer available at `http://www-library.lbl.gov/disclaimer`.

References

1. M. Abdalla, M. Bellare, and P. Rogaway. The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES. In *CT – RSA '01*, LNCS 2020, pages 143–158. Springer-Verlag, Berlin, 2001.
2. W. Aiello, S. M. Bellovin, M. Blaze, J. Ioannidis, O. Reingold, R. Canetti, A. D. Keromytis. Efficient, DoS-resistant, Secure Key Exchange for Internet Protocols. In *Proc. of the 9th CCS*, pages 48–58. ACM Press, New York, 2002.
3. W. Aiello, S. M. Bellovin, M. Blaze, J. Ioannidis, O. Reingold, R. Canetti, A. D. Keromytis. Just Fast Keying: Key Agreement In A Hostile Internet. In *ACM Transactions on Information and System Security*, Vol. 7, No. 2, pages 1–30. ACM Press, May 2004.
4. B. Barak and S. Halevi. An architecture for robust pseudo-random generation and applications to `/dev/random`. Available at `http://eprint.iacr.org/`.
5. B. Barak, R. Shaltiel and E. Tromer. True Random Number Generators Secure in a Changing Environment. In *CHES '03*, pages 166–180. LNCS 2779, 2003.
6. S. M. Bellovin and M. Merritt. Encrypted Key Exchange: Password-Based Protocols Secure against Dictionary Attacks. In *Proc. of the Symposium on Security and Privacy*, pages 72–84. IEEE, 1992.
7. M. Bellare, R. Canetti and H. Krawczyk. Keying Hash Functions for Message Authentication. In *Crypto '96*, LNCS 1109, pages 1–15. Springer-Verlag, Berlin, 1996.
8. M. Bellare, R. Canetti and H. Krawczyk. Pseudorandom Functions Revisited: The Cascade Construction and Its Concrete Security. In *Proc. of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 514–523, 1996.
9. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among Notions of Security for Public-Key Encryption Schemes. In *Crypto '98*, LNCS 1462, pages 26–45. Springer-Verlag, Berlin, 1998.
10. M. Bellare and P. Rogaway. Random Oracles Are Practical: a Paradigm for Designing Efficient Protocols. In *Proc. of the 1st CCS*, pages 62–73. ACM Press, New York, 1993.
11. M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *Crypto '93*, LNCS 773, pages 232–249. Springer-Verlag, Berlin, 1994.
12. M. Bellare and P. Rogaway. Provably Secure Session Key Distribution: the Three Party Case. In *Proc. of the 27th STOC*. ACM Press, New York, 1995.
13. M. Bellare and P. Rogaway. Optimal Asymmetric Encryption – How to Encrypt with RSA. In *Eurocrypt '94*, LNCS 950, pages 92–111. Springer-Verlag, Berlin, 1995.
14. D. Boneh. The Decision Diffie-Hellman problem. In *Proc. of Third Number Theory Symposium*, LNCS 1423, pages 48–63. Springer-Verlag, Berlin, 1998.
15. B. Möller. A Public-Key Encryption Scheme with Pseudo-Random Ciphertexts. In *ESORICS '04*, LNCS 3193, pages 335–351. Springer-Verlag, Berlin, 2004.
16. C. Boyd, P. Montague, and K. Nguyen. Elliptic Curve Based Password Authenticated Key Exchange Protocols. In *ACISP '01*, LNCS 2119, pages 487–501. Springer-Verlag, Berlin, 2001.

17. R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz and A. Sahai. Exposure-Resilient Functions and All-Or-Nothing Transforms. In *Eurocrypt '00*, LNCS 1807, pages 453–469. Springer-Verlag, Berlin, 2000.
18. R. Canetti and H. Krawczyk. Universally Composable Notions of Key Exchange and Secure Channels. In *Eurocrypt '02*, LNCS 2332, pages 337–351. Springer-Verlag, Berlin, 2002.
19. R. Canetti and H. Krawczyk. Security Analysis of IKE's Signature-based Key-Exchange Protocol. In *Crypto '02*, LNCS 2442, pages 143–161. Springer-Verlag, Berlin, 2002. Cryptology ePrint Archive 2002/120. Full Version. Available at `http://eprint.iacr.org/`.
20. W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT–22(6):644–654, November 1976.
21. Y. Dodis. Exposure-Resilient Cryptography. *PhD Thesis*, MIT, August 2000.
22. 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.
23. Y. Dodis, A. Sahai, A. Smith. On perfect and adaptive security in exposure-resilient cryptography. In *Eurocrypt '01*, LNCS 2405, pages 301–324. Springer-Verlag, Berlin, 2001.
24. T. El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, IT–31(4):469–472, July 1985.
25. 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.
26. O. Goldreich. Foundations of Cryptography (Fragments of a Book). 1995.
27. S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure Against Adaptative Chosen-Message Attacks. *SIAM Journal of Computing*, 17(2):281–308, April 1988.
28. D. Harkins and D. Carrel. The Internet Key Exchange (IKE). RFC 2409, November 1998.
29. 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.
30. IETF. Archive of IETF-IPSEC. Available at `http://www.vpnc.org/ietf-ipsec/`.
31. I. Impagliazzo, L. Levin, and M. Luby. Pseudo-Random Generation from One-Way Functions. In *Proc. of the 21st STOC*, pages 12–24. ACM Press, New York, 1989.
32. I. Impagliazzo and D. Zuckerman. How to Recycle Random Bits. In *Proc. of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 248–253, 1989.
33. B. Kaliski. One-Way Permutations on Elliptic Curves. *Journal of Cryptology*, 3(3):187–199, 1991.
34. J. Kamp and D. Zuckerman. Deterministic Extractors for Bit-Fixing Sources and Exposure-Resilient Cryptography. In *Proc. of the 44th Annual IEEE Symposium on Foundations of Computer Science*, 2003.
35. C. Kaufman. The Internet Key Exchange (IKEv2) Protocol. INTERNET-DRAFT draft-ietf-ipsec-ikev2-17.txt, September 23, 2004. Available at `http://www.ietf.org/internet-drafts/draft-ietf-ipsec-ikev2-17.txt`
36. H. Krawczyk. The SIGMA Family of Key-Exchange Protocols. Available at `http://www.ee.technion.ac.il/~hugo/sigma.html`.
37. H. Krawczyk. SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols. In *Crypto '03*, LNCS 2729, pages 400–425. Springer-Verlag, Berlin, 2003. Extended version available at `http://www.ee.technion.ac.il/~hugo/sigma.html`.
38. M. Naor and O. Reingold. From Unpredictability to Indistinguishability: A Simple Construction of Pseudo-Random Functions from MACs. In *Crypto '98*, LNCS 1462. Springer-Verlag, Berlin, 1998.
39. M. Santha and U. V. Vazirani. Generating quasi-random sequences from semi-random sources. In *J. of Computer and System Sciences*, 63:612–626, 1986.
40. R. Schoof. Counting Points on Elliptic Curves over Finite Fields. In *J. Théor. Nombres Bordeaux*, 7:219–254, 1995.
41. R. Shaltiel. Recent developments in Extractors. In *Bulletin of the European Association for Theoretical Computer Science*, Volume 77, June 2002, pages 67–95. Available at `http://www.wisdom.weizmann.ac.il/~ronens/papers/survey.ps`, 2002.
42. V. Shoup. A Proposal for an ISO Standard for Public-Key Encryption, december 2001. ISO/IEC JTC 1/SC27.
43. V. Shoup. OAEP Reconsidered. In *Crypto '01*, LNCS 2139, pages 239–259. Springer-Verlag, Berlin, 2001.
44. V. Shoup. A Computational Introduction to Number Theory Algebra. In *Cambridge University Press*, 2005. Freely available at `http://www.shoup.net/ntb/`.
45. V. Shoup. Sequences of Games: A Tool for Taming Complexity in Security Proofs. Available at `http://www.shoup.net/papers/`, 2004.
46. J. H. Silverman. *The Arithmetic of Elliptic Curves*, volume 106 of *Graduate Texts in Mathematics*. Springer-Verlag, 1986.
47. T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246, January 1999. OpenSSL. version 0.9.7e
48. L. Trevisan and S. Vadhan. Extracting Randomness from Samplable Distributions. In *Proc. of the 41st Annual IEEE Symposium on Foundations of Computer Science*, 2000.

A Authenticated Key Exchange

An algorithm for key exchange is an interactive protocol between two players A (for Alice) and B (for Bob) at the end of which they both share a session key sk. Each of the players may have several *instances* involved in distinct, possibly concurrent, executions of the protocol. Instances of party A (resp. B) are modeled by oracles [11, 12], denoted Π_A^i (resp. Π_B^j), or by Π when we consider any player's instance.

A.1 The Communication Model

During executions of this protocol, the adversary has the entire control of the network, and tries to break the privacy of the key (*semantic security*) or the authentication of the players (*mutual authentication*). To model the various capabilities of the adversary, several queries are available to the latter:

- $\mathsf{Execute}(A, i, B, j)$: This query models passive attacks, where the adversary gets access to honest executions of the protocol between the i-th instance of A (Π_A^i) and the j-th instance of B (Π_B^j), by eavesdropping for example.
- $\mathsf{Reveal}(U, i)$: This query models the misuse of the session key by the i-th instance of U (either A or B). Our model thus encompasses the so-called *known-key attacks*. The query is only available to the adversary if the attacked instance actually "holds" a session key. It then releases the latter. During the protocol, the instance will claim that it actually holds a session key when it flips the flag accept to true. This may never happen if the instance detects that the other party does not behave honestly: it then terminates without accepting (the flag $\mathsf{terminate}$ changes to true, while the flag accept remains to false.) A Reveal-query asked to such a player is answered by $\perp$.

A.2 Session Key Privacy

The first goal of an adversary is to break the privacy of the session key (a.k.a., semantic security): it wants to learn some information about it. Such a security notion is modeled by the game $\mathbf{Game}^{\mathsf{ake}}(\mathcal{A})$, in which one more query is available to the adversary $\mathcal{A}$: the Test-query. This query $\mathsf{Test}(U, i)$ can be asked at most once, on an instance of any party which actually holds a *fresh* session key. The freshness notion (which will be defined more precisely later, with the partnering relation) roughly means that the session key is not "obviously" known to the adversary. This query is answered as follows: one flips a (private) coin b and forwards sk (the value $\mathsf{Reveal}(U, i)$ would output) if $b = 1$, or a uniformly distributed random value if $b = 0$.

When playing this game, the goal of the adversary is to guess the bit b involved in the Test-query, by outputting its guess b'. We denote the **AKE advantage** against a protocol P as the probability that $\mathcal{A}$ correctly guesses the value of b. More precisely, we define $\mathsf{Adv}_P^{\mathsf{ake}}(\mathcal{A}) = 2\Pr[b = b'] - 1$.

A.3 Active Adversaries

Classical attacks in key exchange protocols are the so-called "man-in-the-middle" attacks. They do not involve a simple passive adversary, but an adversary which intercepts, replays, modifies or creates flows from/to Alice to/from Bob. We thus consider the powerful query

- $\mathsf{Send}(U, i, m)$: this allows the adversary $\mathcal{A}$ to send a message to the i-th instance of U. The adversary $\mathcal{A}$ gets back the response this instance generates in processing the message m according to the protocol and its current state. A query $\mathsf{Send}(A, i, \mathtt{Start})$ initiates a key exchange execution, and thus the adversary receives the initial flow the player A should send out to the player B (we assume here that Alice is the initiator.)

When considering active adversaries, the Execute-query becomes useless, since using the Send-query, and relaying the flows, the adversary has the ability to carry out honest executions among parties. We can thus forget the former one in our model. Note however that Execute-queries are of major interest when dealing with password-based authentication. In such a case, it is indeed important to distinguish passive and active attacks.

A.4 Authentication

Another goal for an adversary may also be to break the authentication of the players (impersonate a player, or simply make a player to share a key with nobody —unknown-key attacks—, or a non-intended partner —miss-binding identity attacks—). The *mutual authentication* is the formal security notion which prevents all these kinds of attacks. More precisely, a key exchange scheme achieves mutual authentication if any party who terminates has an accepting partner. Combined with semantic security which roughly means that nobody except the intended partners knows the key, it guarantees any terminating party that the intended partner actually knows the key, and nobody else has any information about it. This is usually achieved by additional rounds in which parties prove their knowledge of the key material to their partners: key confirmation rounds.

Note however that even if one is only interested in the privacy of the keys under active attacks, players have to authenticate themselves in some way. Otherwise, it would be easy for the adversary to impersonate Bob to Alice, and thus finally share a key with Alice, while the latter has no partner (except the adversary). Since the adversary knows the key, he definitely can distinguish it in the Test-query asked to Alice. Therefore, while semantic security does not guarantee a strong authentication (a.k.a. explicit authentication) it still ensures an implicit one: when Alice accepts a key, it can also be known to Bob only (but to nobody else, and maybe Bob neither.)

A.5 Freshness and Partnering

We restricted the Test-query on *fresh* keys. Indeed, if Π_A^i and Π_B^j agreed on a session key sk, a query $\mathsf{Reveal}(A, i)$ provides this session key sk to the adversary. Thereafter, a $\mathsf{Test}(B, j)$ (or *a fortiori* $\mathsf{Test}(A, i)$) would immediately leak all the information about the bit b. This is however the only restriction: a key is said to be *fresh* if neither the instance or its partner has been asked for a Reveal-query.

Therefore, a new notion of *partnership* appears. We say that two instances are *partners* if they have been involved in the same session of the protocol, which is named by its **session ID** or sid, defined as the (common) view of the execution: the concatenation of the *crucial* flows. The *crucial* flows are the required flows for achieving acceptance from both sides.

B Security Notions and Computational Assumptions

In this section we review the cryptographic primitives (Signatures, Message Authentication Codes (MACs)) and the Diffie-Hellman intractability assumptions.

B.1 Signature Schemes

A signature scheme $\mathsf{SIG} = (\mathsf{SIG.Key}, \mathsf{SIG.Sign}, \mathsf{SIG.Verify})$ is defined by the three following algorithms:

- The *key generation algorithm* SIG.Key. On input 1^k, the algorithm SIG.Key produces a pair $(\mathsf{pk}, \mathsf{sk})$ of matching public (verification) and private (signing) keys.
- The *signing algorithm* SIG.Sign. Given a message m and a pair of matching public and private keys $(\mathsf{pk}, \mathsf{sk})$, SIG.Sign produces a signature σ. The signing algorithm might be probabilistic.

– The *verification algorithm* $\mathsf{SIG.Verify}$. Given a signature σ, a message m and a public key pk, $\mathsf{SIG.Verify}$ tests whether σ is a valid signature of m with respect to pk.

Several security notions have been defined about signature schemes, mainly based on the seminal work of Goldwasser *et al* [27]. It is now classical to ask for the impossibility of existential forgeries, even for adaptive chosen-message adversaries:

– An *existential forgery* is a new message-signature pair, valid and generated by the adversary. The corresponding security level is called *existential unforgeability* (EUF).
– The verification key is public, including to the adversary. But more information may also be available. The strongest kind of information is definitely formalized by the *adaptive chosen-message attacks* (CMA), where the attacker can ask the signer to sign any message of its choice, in an adaptive way.

As a consequence, we say that a signature scheme is secure if it prevents existential forgeries, even under adaptive chosen-message attacks.

B.2 Message Authentication Codes

A Message Authentication Code $\mathsf{MAC} = (\mathsf{MAC.Sign}, \mathsf{MAC.Verify})$ is defined by the two following algorithms, with a secret key sk uniformly distributed in $\{0,1\}^\ell$:

– The *MAC generation algorithm* $\mathsf{MAC.Sign}$. Given a message m and secret key $\mathsf{sk} \in \{0,1\}^\ell$, $\mathsf{MAC.Sign}$ produces an authenticator μ. This algorithm might be probabilistic.
– The *MAC verification algorithm* $\mathsf{MAC.Verify}$. Given an authenticator μ, a message m and a secret key sk, $\mathsf{MAC.Verify}$ tests whether μ has been produced using $\mathsf{MAC.Sign}$ on inputs m and sk.

As for signature schemes, the classical security level for MAC is to prevent existential forgeries, even for an adversary which has access to the generation and the verification oracles.

B.3 Authentication Schemes

In this section, we simply unify the two above primitives, so that analyses in this paper are quite general (in the symmetric or the asymmetric settings.) We thus define an authentication scheme by three algorithms $\mathsf{AUTH} = (\mathsf{AUTH.Key}, \mathsf{AUTH.Sign}, \mathsf{AUTH.Verify})$:

– The *key generation algorithm* $\mathsf{AUTH.Key}$. On input 1^k, the algorithm $\mathsf{AUTH.Key}$ produces a pair $(\mathsf{vk}, \mathsf{sk})$ of matching verification and signing keys (they can be either the same or different.)
– The *signing algorithm* $\mathsf{AUTH.Sign}$. Given a message m and the signing key sk, $\mathsf{AUTH.Sign}$ produces an authenticator σ.
– The *verification algorithm* $\mathsf{AUTH.Verify}$. Given an authenticator σ, a message m and a verification key vk, $\mathsf{AUTH.Verify}$ tests whether σ is a valid authenticator of m with respect to vk.

Such an authentication scheme is said to be secure if it prevents existential forgeries, even for an adversary which has access to the signing and the verification oracles. This is measured by

$$\mathsf{Succ}^{\mathsf{euf-cma}}_{\mathsf{AUTH}}(\mathcal{A}, q_s, q_v) = \Pr\left[\begin{array}{c}(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{AUTH.Key}(1^k), (m, \sigma) \leftarrow \mathcal{A}^{\mathsf{AUTH.Sign}(\mathsf{sk};\cdot), \mathsf{AUTH.Verify}(\mathsf{vk};\cdot,\cdot)} : \\ \mathsf{AUTH.Verify}(\mathsf{vk}; m, \sigma) = 1\end{array}\right],$$

where the adversary can ask up to q_s and q_v queries to the signing and verification oracles $\mathsf{AUTH.Sign}$ and $\mathsf{AUTH.Verify}$ respectively.

B.4 Computational Assumptions

When one deals with key exchange, the classical problem which arises is the problem introduced by Diffie-Hellman in the seminal paper about asymmetric cryptography [20]. More formally, we consider a finite cyclic group $\mathbb{G}$ of prime order q with a generator g, which we denote multiplicatively in this definition: $\mathbb{G} = (\langle g \rangle, \times)$. Two problems are usually assumed to be intractable, in well-chosen groups:

– the *Computational Diffie-Hellman Problem*, in which given random elements g^x and g^y in $\mathbb{G}$, one wants to find $\mathsf{DH}(g^x, g^y) = g^{xy}$. The actual intractability is measured, for any adversary $\mathcal{A}$, by

$$\mathsf{Succ}^{\mathsf{cdh}}_{g,\mathbb{G}}(\mathcal{A}) = \Pr[x, y \stackrel{R}{\leftarrow} \mathbb{Z}_q : \mathsf{DH}(g^x, g^y) \leftarrow \mathcal{A}(g^x, g^y)].$$

– the *Decisional Diffie-Hellman Problem*, in which given random elements g^x and g^y in $\mathbb{G}$, and a candidate g^z for the value $\mathsf{DH}(g^x, g^y)$, one should guess whether this is the actual solution or not. The actual intractability is measured, for any distinguisher $\mathcal{D}$, by

$$\mathsf{Adv}^{\mathsf{ddh}}_{g,\mathbb{G}}(\mathcal{D}) = \left| \Pr[x, y \stackrel{R}{\leftarrow} \mathbb{Z}_q : 1 \leftarrow \mathcal{D}(g^x, g^y, g^{xy})] - \Pr[x, y, z \stackrel{R}{\leftarrow} \mathbb{Z}_q : 1 \leftarrow \mathcal{D}(g^x, g^y, g^z)] \right|.$$

In the following, we work on elliptic curves, which groups are usually denoted in an additive way: $\mathbb{G} = (\langle \mathbf{P} \rangle, +)$. The latter problem can be stated as follows by adapting the notations: in the *Elliptic Curve Decisional Diffie-Hellman Problem*, given random elements $x \cdot \mathbf{P}$ and $y \cdot \mathbf{P}$ in the group of points $\mathbb{G}$, of order q, and a candidate $z \cdot \mathbf{P}$ for the value $\mathsf{ECDH}(x \cdot \mathbf{P}, y \cdot \mathbf{P})$, one should guess whether this is the actual solution or not. The intractability is measured, for any distinguisher $\mathcal{D}$, by the advantage $\mathsf{Adv}^{\mathsf{ecddh}}_{\mathbf{P},\mathbb{G}}(\mathcal{D})$ defined as above.

Note 19. We insist here on the well-known fact that the intractability of any decisional Diffie-Hellman problem just means that the Diffie-Hellman value is indistinguishable from a random element in the cyclic group. It does not mean that the encoding of a Diffie-Hellman value is indistinguishable from a random bit-string [42].

B.5 Upper-Bounds for Time-Constrained Adversaries

As usual, for all the above success probabilities or advantages, we denote by $\mathsf{Succ}(t, \ldots)$ and $\mathsf{Adv}(t, \ldots)$ the maximal probabilities over all the adversaries which running time is bounded by t.

C Proof of Theorem 18

Let $\mathcal{A}$ be an adversary, and let $\mathbf{G}_0$ be the original AKE attack game. Let b and b' be as defined in the security model (see appendix A), and let S_0 be the event that $b = b'$.
Game $\mathbf{G}_0$: This is the real protocol. In this game, we are interested in the event S_0, which occurs if $b = b'$ in this game, where b is the bit involved in the Test-query and b' is the output of the adversary $\mathcal{A}$.
Game $\mathbf{G}_1$: We modify the oracle instances as follows. If the adversary submits a new authenticator (σ_A or σ_B) which has not been previously generated by an oracle, and thus our simulation, then in game $\mathbf{G}_1$, we reject it and the instance we are simulating (and which receives such a *forged* message), stops: it terminates without accepting.

Let F_1 be the event that in game $\mathbf{G}_1$ an authenticator is rejected that would not have been rejected under the rules of game $\mathbf{G}_0$. Since these two games proceed identically until F_1 occurs, we have $\Pr[S_0 \wedge \neg F_1] = \Pr[S_1 \wedge \neg F_1]$, and applying Lemma 1 of [43, 45] with (S_0, S_1, F_1), we have $|\Pr[S_0] - \Pr[S_1]| \leq \Pr[F_1]$. From the following lemma, one immediately gets:

$$|\Pr[S_0] - \Pr[S_1]| \leq 2 \cdot \mathsf{Succ}^{\mathsf{euf-cma}}_{\mathsf{AUTH}}(t, q_s, q_s). \qquad (1)$$

Lemma 20.

$$\Pr[F_1] \leq 2 \cdot \mathsf{Succ}^{\mathsf{euf-cma}}_{\mathsf{AUTH}}(t, q_s).$$

Proof. We want to bound $\Pr[F_1]$. This probability is bounded by the probability that an adversary $\mathcal{A}'$ —running in expected time nearly twice the same as the running time of the original adversary $\mathcal{A}$— can forge an authenticator under a chosen-message attack. In this case, $\mathcal{A}'$ accesses a signing oracle and tries to forge a new authenticator. At the beginning, $\mathcal{A}'$ picks at random a bit b and according to this bit, it plays the role of an adversary against either A or B authentication. If $b = 0$, then $\mathcal{A}'$ uses the signing oracle to simulate A authenticators, but knows the signing key of B, and if $b = 1$, $\mathcal{A}'$ uses the signing oracle to simulate B authenticators, but knows the signing key of A. It is easy to check that all the $\mathsf{Reveal}(U, i)$, $\mathsf{Send}(U, i, m)$ and $\mathsf{Test}(U, i)$ queries will be perfectly simulated and there is no way for $\mathcal{A}$ to guess which of the two signing keys are known, and thus which of the two authentication schemes we try to break. Consequently, if the event F_1 happens, the adversary $\mathcal{A}$ has been able to forge an authenticator either for A or B. If the forgery is an authenticator under the unknown signing key, then $\mathcal{A}'$ can win the game; otherwise he cannot do anything with the forgery. On average, by running twice the algorithm $\mathcal{A}'$, he will forge an authenticator. If $\mathcal{A}$ makes q_s Send-queries and runs within time t, then $\mathcal{A}'$ runs in the same time, since exactly the same number of authenticators have to be generated. Therefore,

$$\begin{aligned}
\mathsf{Succ}^{\mathsf{euf-cma}}_{\mathsf{AUTH}}(t, q_s, q_s) &\geq \mathsf{Succ}(\mathcal{A}', q_s) = \Pr[\mathcal{A}' \text{ forges}] \\
&\geq \Pr[\mathcal{A}' \text{ forges } A \wedge b = 0] + \Pr[\mathcal{A}' \text{ forges } B \wedge b = 1] \\
&\geq \frac{1}{2} \cdot \Pr[\mathcal{A}' \text{ forges } A] + \frac{1}{2} \cdot \Pr[\mathcal{A}' \text{ forges } B] = \frac{1}{2} \cdot \Pr[\mathcal{A} \text{ forges}] = \frac{1}{2} \cdot \Pr[F_1].
\end{aligned}$$

□

Game $\mathbf{G}_2$: In this game, we try to avoid the use of the discrete-log of the elements X_0, X_1, Y_0, Y_1. We thus introduce two random DDH triples (X, Y, Z) and $(\tilde{X}, \tilde{Y}, \tilde{Z})$: the first one on the elliptic curve $\mathbb{E}$ and the second on the twisted curve $\tilde{\mathbb{E}}$. Then, using the classical random self-reducibility of the Diffie-Hellman problem, one can introduce the above triples in all the sessions which can be tested by the adversary. We do not need to modify the other sessions.

The complete behavior of our simulation in this game is described in Appendix D. It is then clear that games $\mathbf{G}_1$ and $\mathbf{G}_2$ are equivalent, since we have consistently replaced one set of random variables by another set of identically distributed random variables. In particular, $\Pr[S_1] = \Pr[S_2]$.

Game $\mathbf{G}_3$: Game $\mathbf{G}_3$ is exactly the same as game $\mathbf{G}_2$, except that in all the rules, we use a random triple (X, Y, Z) coming from a random distribution $(x \cdot \mathbf{P}, y \cdot \mathbf{P}, z \cdot \mathbf{P})$, instead of a DDH triple. The distance between the two games is clearly bounded by the advantage of any adversary against the DDH (see Appendix E):

$$|\Pr[S_2] - \Pr[S_3]| \leq \mathsf{Adv}^{\mathsf{ecddh}}_{\mathbf{P},\langle\mathbf{P}\rangle}(t + 8q_sT_m). \tag{2}$$

Game $\mathbf{G}_4$: The modification between games $\mathbf{G}_4$ and $\mathbf{G}_3$ is the same that between $\mathbf{G}_3$ and $\mathbf{G}_2$, except that instead of replacing a DDH triple by a random triple on the elliptic curve $\mathbb{E}$, we do the same on the triple on the twisted $\tilde{\mathbb{E}}$. Hence, we have

$$|\Pr[S_3] - \Pr[S_4]| \leq \mathsf{Adv}^{\mathsf{ecddh}}_{\mathbf{Q},\langle\mathbf{Q}\rangle}(t + 8q_sT_m). \tag{3}$$

Game $\mathbf{G}_5$: In this game, we modify the generation of the master key K in each session by picking at random in $\mathbb{F}_p$ instead of as $[Z]_{\mathsf{abs}}$. Granted to the random-self reducibility property used in game $\mathbf{G}_2$ (described in Appendix D), the Z's are random elements on the curves. According to Lemma 16,

$$|\Pr[S_4] - \Pr[S_5]| \leq q_s \times \frac{2\sqrt{2}}{\sqrt{2^\ell}}. \tag{4}$$

Game $\mathbf{G}_6$: In this game, we modify the generation of the master key K in each session by picking at random in $\{0,1\}^\ell$ instead of as random in $\mathbb{F}_p$. This is done independently for each session. According to Lemma 14,

$$|\Pr[S_5] - \Pr[S_6]| \leq q_s \times \frac{2}{\sqrt{2^\ell}}. \tag{5}$$

Game $\mathbf{G}_7$: In this game, instead of using the PRF in order to generate the MAC key k_m and the session key sk, we pick random values in $\{0,1\}^n$. We use a classical hybrid argument [26] in order to prove that the difference between game $\mathbf{G}_6$ and $\mathbf{G}_7$ is $q_s \times \mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(\mathcal{A}, 2)$. Hybrids consist of a sequence of random variables V_i, $1 \leq i \leq q_s + 1$, such that

1. the random variable V_i is constructed as follows : in the first $(i-1)$ sessions, the session and MAC keys are generated according to game $\mathbf{G}_7$ and in the $(q_s - i + 1)$ sessions, they are generated according the $\mathbf{G}_6$;
2. extreme hybrids $(i = 1)$ and $(i = q_s + 1)$ collide with $\mathbf{G}_6$ and $\mathbf{G}_7$ respectively;
3. random values of each hybrid can be produced by a probabilistic polynomial time algorithm and the session that we modify is independent of the other sessions;
4. there are only polynomially many hybrids.

The hybrids allow us to define q_s different games and in each of the hybrids we only ask 2 queries to the PRF. The difference between two consecutive hybrid games can be shown to be less than $\mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(\mathcal{A}, 2)$, as we did between games $\mathbf{G}_2$ and $\mathbf{G}_3$, but here by constructing an adversary $\mathcal{A}'$ against the PRF security game.

$$|\Pr[S_6] - \Pr[S_7]| \leq q_s \times \mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(\mathcal{A}, 2). \tag{6}$$

It is also clear that in game $\mathbf{G}_7$, the hidden bit b of the Test-query in independent of all values directly or indirectly accessible to the adversary. Hence, $\Pr[S_7] = 1/2$. Combined with Equations (1), (2), (3), (4), (5) and (6), it gives the expected result.

D Random Self-Reducibility

Game $\mathbf{G}_2$ is identical to game $\mathbf{G}_1$, except that we apply the following special rules when dealing with the $\mathsf{Reveal}(U, i)$, $\mathsf{Test}(U, i)$ and $\mathsf{Send}(U, i, m)$ queries :

R1: When processing a $\mathsf{Send}(A, i, \mathtt{Start})$ query, the simulator picks four random values $a_0, x_0 \overset{R}{\leftarrow} \mathbb{Z}_q$ and $a_1, x_1 \overset{R}{\leftarrow} \mathbb{Z}_{\tilde{q}}$, computes $X_0 = a_0 \cdot X + x_0 \cdot \mathbf{P}$ and $X_1 = a_1 \cdot \tilde{X} + x_1 \cdot \mathbf{Q}$, and stores in some $\mathcal{X}$-table (a_0, x_0, X_0) and (a_1, x_1, X_1).

R2: When processing a $\mathsf{Send}(B, j, (s, X_0, X_1))$ query,
- if the two elements X_0 and X_1 have been computed by our simulator and thus have been stored in the $\mathcal{X}$-table, then it generates the same way its answer by choosing four random values $b_0, y_0 \overset{R}{\leftarrow} \mathbb{Z}_q$ and $b_1, y_1 \overset{R}{\leftarrow} \mathbb{Z}_{\tilde{q}}$, it computes $Y_0 = b_0 \cdot Y + y_0 \cdot \mathbf{P}$ and $Y_1 = b_1 \cdot \tilde{Y} + y_1 \cdot \mathbf{Q}$, and stores in some $\mathcal{Y}$-table (b_0, y_0, Y_0) and (b_1, y_1, Y_1). It can now compute $Z_0 = a_0 b_0 \cdot Z + x_0 b_0 \cdot Y + a_0 y_0 \cdot X + x_0 y_0 \cdot \mathbf{P}$ and $Z_1 = a_1 b_1 \cdot \tilde{Z} + x_1 b_1 \cdot \tilde{Y} + a_0 y_1 \cdot \tilde{X} + x_1 y_1 \cdot \mathbf{Q}$.
- if one of the elements X_0 or X_1 has not been previously computed by our A-simulation, then it proceeds as in the game $\mathbf{G}_1$.

In the first case, the simulator uses the key Z_0 or Z_1 as a master key according to be bit β whereas in the second case, the master key will be calculated as in the previous game.

R3: When processing a $\mathsf{Send}(A, i, (s, Y_0, Y_1, Bob, \sigma_B, \mu_B))$, then if the authenticator is correct, we can assume that the corresponding values $(X_0, X_1, Y_0, Y_1, Z_0, Z_1)$ have been computed by the simulator: we can compute the master key, and thus compute and check the MAC to determine the bit d, if it exists.

R4: When processing a $\mathsf{Test}(U,i)$-query, we know that such a query can only be asked on accepting instance, and accepting session can only happen when the simulator knows the correct value Z_d and can answer such query as in the game $\mathbf{G}_1$.

R5: When processing a $\mathsf{Reveal}(U,i)$-query, as in the rule **R4**, the simulator is able to answer such queries as in the previous game.

It is easy to see that in the second case of rule **R2**, as in game $\mathbf{G}_1$, the adversary will not been able to forge an authenticator, and then he will not be able to generate a correct third message. Consequently, the session will not be accepted by any party and so the adversary will not be able to send a Test-query to any instance. Hence, the simulation will be consistent.

E The DDH Distinguisher

We assume that $\mathcal{A}$ is an attacker that breaks the AKE security game with a different advantage in Game $\mathbf{G}_3$ than in Game $\mathbf{G}_2$, then we construct an adversary $\mathcal{A}'$ which is able to distinguish triples coming from either a DDH or a random distribution: at the beginning of the experiment, $\mathcal{A}'$ receives a triple (X,Y,Z) which is a DDH triple if $b=0$ or a random triple if $b=1$. Then $\mathcal{A}'$ runs the attacker $\mathcal{A}$ using this triple to simulate all the queries as in the previous game (with is actually either the previous game if $b=0$ or the current game if $b=1$).. When the $\mathsf{Test}(U,i)$-query happens, $\mathcal{A}'$ picks a bit at random b' and sends according to b' either the real session key if $b'=0$ or a random session key. Eventually, $\mathcal{A}$ will reply with a bit b''. Finally, if $b'=b''$, then $\mathcal{A}'$ returns a bit $b^\star=1$, else it returns $b^\star=0$.

$$\begin{aligned}\Pr[b^\star=b] &= \frac{1}{2}\cdot\left(\Pr[b^\star=0|b=0]+\Pr[b^\star=1|b=1]\right) = \frac{1}{2}\cdot\big(\Pr[b'=b''|\mathsf{DDH}]+\Pr[b'\neq b''|\mathsf{Rand}]\big)\\ &= \frac{1}{2}\cdot\big(\Pr[b'=b''|\mathsf{DDH}]+1-\Pr[b'=b''|\mathsf{Rand}]\big) = \frac{1}{2}\cdot\left(\Pr[S_2]+1-\Pr[S_3]\right).\end{aligned}$$

Note that the running time t' of $\mathcal{A}'$ is the same as t plus the time for the computations of the random self-reducibility: $t+8q_sT_m$.

F Distribution of the Preliminary Secret

In this section we show that the distribution of the preliminary secret key K is statistically indistinguishable from the uniform distribution on $\{0,1\}^\ell$. On the one hand, we prove that it is statistically indistinguishable from the uniform distribution on $\{0,\dots,p-1\}$ and then that the latter distribution is statistically indistinguishable from the uniform distribution on $\{0,1\}^\ell$.

Let us denote by $\mathcal{D}$ the distribution of K:

$$\mathcal{D}=\{b\overset{R}{\leftarrow}\{0,1\},\mathbf{R}_0\overset{R}{\leftarrow}\mathbb{E},\mathbf{R}_1\overset{R}{\leftarrow}\tilde{\mathbb{E}}:K=[\mathbf{R}_b]_{\mathsf{abs}}\}=\{b\overset{R}{\leftarrow}\{0,1\},x_0\overset{R}{\leftarrow}[\mathbb{E}]_{\mathsf{abs}},x_1\overset{R}{\leftarrow}[\tilde{\mathbb{E}}]_{\mathsf{abs}}:K=x_b\}.$$

F.1 Proof of Lemma 16

In this proof, we note $E_0=[\mathbb{E}]_{\mathsf{abs}}$ and $E_1=[\tilde{\mathbb{E}}]_{\mathsf{abs}}$. Then, we have $\mathbb{F}_p=E_0\cup E_1$. As already noticed, $\#\mathbb{E}=p+1-t=q$ and $\#\tilde{\mathbb{E}}=p+1+t=\tilde{q}$, where t is less than $2\sqrt{p}$. Then $\#E_0=q/2$ and $\#E_1=\tilde{q}/2$, since one abscissa corresponds to two points on the elliptic curves. We thus have

$$\begin{aligned}\delta &= \sum_{x\in\mathbb{F}_p}\left|\Pr_{K\overset{R}{\leftarrow}\mathcal{U}_p}[K=x]-\Pr_{K\overset{R}{\leftarrow}\mathcal{D}}[K=x]\right| = \sum_{x\in\mathbb{F}_p}\left|\frac{1}{p}-\Pr_{b\overset{R}{\leftarrow}\{0,1\}}[x_0\overset{R}{\leftarrow}E_0,x_1\overset{R}{\leftarrow}E_1:x=x_b]\right|\\ &= \sum_{x\in E_0}\left|\frac{1}{p}-\Pr_{b\overset{R}{\leftarrow}\{0,1\}}[x_0\overset{R}{\leftarrow}E_0,x_1\overset{R}{\leftarrow}E_1:x=x_b]\right| + \sum_{x\in E_1}\left|\frac{1}{p}-\Pr_{b\overset{R}{\leftarrow}\{0,1\}}[x_0\overset{R}{\leftarrow}E_0,x_1\overset{R}{\leftarrow}E_1:x=x_b]\right|\end{aligned}$$

$$
\begin{aligned}
&= \sum_{x \in E_0} \left| \frac{1}{p} - \frac{1}{2} \times \Pr[x_0 \xleftarrow{R} E_0 : x = x_0] \right| + \sum_{x \in E_1} \left| \frac{1}{p} - \frac{1}{2} \times \Pr[x_1 \xleftarrow{R} E_1 : x = x_1] \right| \\
&= \frac{q}{2} \times \left| \frac{1}{p} - \frac{1}{2} \times \frac{2}{q} \right| + \frac{\tilde{q}}{2} \times \left| \frac{1}{p} - \frac{1}{2} \times \frac{2}{\tilde{q}} \right| = \frac{q}{2} \times \left(\frac{1}{q} - \frac{1}{p} \right) + \frac{\tilde{q}}{2} \times \left(\frac{1}{p} - \frac{1}{\tilde{q}} \right) \\
&= \left(\frac{1}{2} - \frac{q}{2p} \right) + \left(\frac{\tilde{q}}{2p} - \frac{1}{2} \right) = \frac{\tilde{q} - q}{2p} = \frac{2t}{2p} \leq \frac{2\sqrt{p}}{p} \leq \frac{2}{\sqrt{p}} \leq \frac{2}{\sqrt{2^{\ell-1}}} \leq \frac{2\sqrt{2}}{\sqrt{2^\ell}}.
\end{aligned}
$$

□

F.2 Proof of Lemma 14

Now, we prove that the statistical distance between the uniform distribution in the space $\mathbb{F}_p \sim \mathbb{Z}_p$ and the uniform distribution in the space $\{0,1\}^\ell \sim \{0, \ldots, 2^\ell - 1\}$, where $2^\ell - \varepsilon \leq p < 2^\ell$ and $0 < \varepsilon \leq 2^{\ell/2}$, is less than $1/\sqrt{2^\ell}$.

$$
\begin{aligned}
\delta' &= \sum_{x \in \{0,1\}^\ell} \left| \Pr_{X \xleftarrow{R} \mathcal{U}_{2^\ell}} [X = x] - \Pr_{X \xleftarrow{R} \mathcal{U}_p} [X = x] \right| \\
&= \sum_{\substack{x \in \{0,1\}^\ell \\ x < p}} \left| \Pr_{X \xleftarrow{R} \mathcal{U}_{2^\ell}} [X = x] - \Pr_{X \xleftarrow{R} \mathcal{U}_p} [X = x] \right| + \sum_{\substack{x \in \{0,1\}^\ell \\ x \geq p}} \left| \Pr_{X \xleftarrow{R} \mathcal{U}_{2^\ell}} [X = x] - \Pr_{X \xleftarrow{R} \mathcal{U}_p} [X = x] \right| \\
&= \sum_{\substack{x \in \{0,1\}^\ell \\ x < p}} \left| \frac{1}{2^\ell} - \frac{1}{p} \right| + \sum_{\substack{x \in \{0,1\}^\ell \\ x \geq p}} \left| \frac{1}{2^\ell} - 0 \right| = p \times \left| \frac{1}{2^\ell} - \frac{1}{p} \right| + (2^\ell - p) \times \frac{1}{2^\ell} \leq \frac{2(2^\ell - p)}{2^\ell} \leq \frac{2\varepsilon}{2^\ell} \leq \frac{2}{\sqrt{2^\ell}}.
\end{aligned}
$$

□

G An Example 200-bit Pair of Curve and Twist

We give a pair of curve and twist suitable for implementing the TAU protocol. This curve was produced using the method sketched in Section 5.3. We choose a curve with $a = -3$, to allow the use of the fast projective group law.

Let $\ell = 200$, and let $p = 2^\ell - 978579$. Let b in $\mathbb{F}_p$ be given by

$$b = 386119362724722930774569388602676779780560253666503462427823.$$

The trace of the curve $\mathbb{E}$ of equation $y^2 = x^3 - 3x + b$, is

$$t_{\mathbb{E}} = -1864972684066157296039917581949.$$

Hence, the group orders of $\mathbb{E}$ and of its twist $\tilde{\mathbb{E}}$ are $p + 1 \pm t_{\mathbb{E}}$, which are both prime numbers.