Anonymous Signatures Made Easy

Marc Fischlin*

Darmstadt University of Technology, Germany
marc.fischlin@gmail.com
www.fischlin.de

Abstract. At PKC 2006, Yang, Wong, Deng and Wang proposed the notion of anonymous signature schemes where signatures do not reveal the signer's identity, as long as some parts of the message are unknown. They also show how to modify the RSA scheme and the Schnorr scheme to derive anonymous signatures in the random oracle model. Here we present a general and yet very efficient approach to build such anonymous schemes from ordinary signature schemes. When instantiated in the random oracle model, our solution is essentially as efficient as the original scheme, whereas our construction also supports an almost as efficient instantiation in the standard model.

Keywords: Anonymity, perfectly one-way hash function, randomness extractor, signature scheme.

1 Introduction

In an anonymous signature scheme, introduced by Yang et al. [9], a signature σ to a message m should hide the identity of a signer. That is, one should not be able to tell whether σ has been produced by the user with public key pk_0 or by the user with public key pk_1. This holds as long there is some hidden residual randomness in the signed message m, otherwise one can easily check the validity of m and σ with respect to the public keys.

Yang et al. discuss several applications of anonymous signature schemes such as authenticated key-transportation with client anonymity and anonymous paper reviewing. Another example are anonymous auctions where bidders can publish their bid and sign the bid prepended by some hidden random string, such that the bidder's identity remains secret and is only revealed if winning the auction. Yang et al. also show that well-known signatures schemes like RSA and Schnorr do not have the anonymity property, yet can be turned into anonymous ones (in the random oracle model).

Our Results. Here we give a very simple and yet general construction method for anonymous signatures from arbitrary signature schemes. Depending on the instantiation of the underlying tools in our transformation we either get an

* This work was supported by the Emmy Noether Program Fi 940/2-1 of the German Research Foundation (DFG).

T. Okamoto and X. Wang (Eds.): PKC 2007, LNCS 4450, pp. 31–42, 2007.

anonymous scheme in the random oracle model, which is essentially as efficient as the original signature scheme, or we get a solution in the standard model with a marginal loss in efficiency only (assuming the existence of regular collision-intractable hash functions[1]).

For the underlying idea suppose for the moment that we have an unforgeable but identity-revealing signature scheme producing signatures σ of length ℓ. Assume further that the unknown message m is distributed uniformly over ℓ-bit strings. If we now define a modified signature scheme where we let $\sigma' = \sigma \oplus m$, then the new scheme would clearly retain unforgeability. At the same time, signatures should still look random to an attacker who is oblivious about m and should thus provide anonymity. The fallacy in this argument —in addition to the overly optimistic assumption about completely random and unknown messages— is that the original signature value σ itself depends on m and thus σ' may not be uniformly distributed anymore.

The solution for the problem with arbitrary message distributions is to use randomness extractors [6,5,8]. Such extractors gather a sufficient amount of "smooth" randomness $\mathsf{Ext}(m)$ from an input m, as long as the input distribution has some intrinsic entropy. That is, if sufficiently large parts of the message are unknown to an attacker, the extracted value $\mathsf{Ext}(m)$ still looks like a uniformly distributed variable.[2] Hence, instead of using the message m to mask the signature we now add the value $\mathsf{Ext}(m)$.

For the second problem, dependencies between the signature of the message and the extracted randomness, we will introduce special randomness extractors whose output $\mathsf{Ext}(m)$ looks random, even if one sees an additional (possibly randomized) hash value $\mathsf{H}(m)$ of the message m. Given such a "good" hash function and extractor combination we can compute the signature σ for the hash value $\mathsf{H}(m)$, and then mask this signature with the extracted value $\mathsf{Ext}(m)$ of the original message:

$$\mathsf{Sig}'(sk, m) = \mathsf{Sig}(sk, \mathsf{H}(m)) \oplus \mathsf{Ext}(m).$$

We note that, if the hash function or the extractor are randomized, then the signature will also include the (public) randomness used to evaluate the functions. It is also worth noticing that signatures constructed as above actually achieve the stronger notion of being pseudorandom, and that this even holds if an attacker knows the secret signing key.

Instantiations. It remains to specify how to build a "good" hash function and randomness extractor pair. In the random oracle model this is very easy. Namely, for a random function H simply define the hash function to be $H(0, \cdot)$ and the randomness extractor to be $H(1, \cdot)$, such that both functions essentially yield

[1] A function is regular if any image has the same number of pre-images.

[2] In the literature randomness extractors are typically defined to produce an output that is statistically close to the uniform distribution. Here we merely need the relaxation to computational indistinguishability where the output *appears* to be random *for efficient observers*. We will use this algorithmic relaxation throughout the paper.

independent outputs $\mathsf{H}(m) = H(0, m)$ and $\mathsf{Ext}(m) = H(1, m)$ for non-trivially distributed messages m. Note that with this instantiation the derived signature scheme is basically as efficient as the original scheme.

To get a solution in the standard model we deploy so-called perfectly one-way hash functions [2,3] where it is infeasible to distinguish between randomized hash values $(\mathsf{H}(x; r), \mathsf{H}(x; r'))$ of the same pre-image x, and hashes $(\mathsf{H}(x; r), \mathsf{H}(x'; r'))$ of independent values x, x'. Take the first part of such a pair $(\mathsf{H}(m; r), \mathsf{H}(m; r'))$ for our message m as the hash input to the signature scheme, and the second part of the pair to be the output of our extractor (appropriately modified to yield pseudorandom outputs). Then the values appear to come from independent inputs m and m' and we get the desired computational independence of the two parts.

Very efficient instantiations of perfectly one-way hash function can de derived, for example, from regular collision-intractable hash functions, together with universal hash functions [3]. Namely, the randomized hash evaluation $\mathsf{H}(m)$ is described by picking an almost universal hash permutation π as public randomness and outputting $h(\pi(m))$ for a regular collision-intractable hash function h. According to our approach this hash function also defines the basic steps of our extractor, except that we have to produce a pseudorandom output. This additional property can be accomplished, for instance, by applying another almost universal hash function ρ to the $h(\pi(m))$ portion and by stretching the outcome with a pseudorandom generator G, i.e., the extractor's output for public randomness π, ρ equals $\mathsf{Ext}(m) = G(\rho(h(\pi(m))))$.

We remark that the informal discussion above hides some technical nuisances. For instance, if we use the suggested instantiation through the perfectly one-way hash functions, then the fact that we apply universal hash functions twice and stretch the final output with a pseudorandom generator, only yields a provably secure solution if we start with enough hidden entropy in the message. This entropy bound exceeds the one for the random-oracle based solution, but still appears to be within reasonable bounds for most applications.

Relationship to Ring Signatures. Ring signatures [7] allow each user from an "ad-hoc" group, the ring, to sign a message such that the signer's identity remains secret, yet everyone can verify that the message has been signed by someone in the ring. In this sense, anonymous signatures are an attenuation of ring signatures, because for anonymous schemes the signer's identity only remains undisclosed as long as the parts of the message are unknown. In fact, this weaker requirement allows us to give a simple and yet general construction of anonymous signatures, whereas ring signatures typically depend on specific assumptions (e.g. [7,4]) or are rather feasibility constructions as in [1]. One advantage of anonymous signatures over ring signature schemes is that anonymity is not bound to a certain group.

Our approach shows that there are anonymous signature schemes which are not ring signatures. Given the complete message m one can easily "peel off" the mask $\mathsf{Ext}(m)$ in our construction and figure out the signer's identity by checking the validity with respect to the keys. It remains an interesting open problem if

there is a general and efficient transformation from anonymous signatures to ring signatures (by that we refer to a transformation which does not involve general non-interactive zero-knowledge proofs as in [1]).

Organization. In Section 2 we introduce the notions of unforgeability and anonymity of signature schemes. In Section 3 we present the construction of the hash function and extractor pairs. In Section 4 we prove our derived anonymous signature scheme to be secure.

2 Preliminaries

For an algorithm A we write $x \leftarrow A(y)$ for a (possibly random) output x of A for input y. Likewise, $x \leftarrow X$ for a set X denotes a uniformly chosen element x from X, and with $x \leftarrow \mathcal{X}(y)$ we refer to x sampled according to distribution $\mathcal{X}$ (parameterized by input y). To make the random coins in probabilistic processes more specific we sometimes write $x \leftarrow A(y;\omega)$ for the output of algorithm A on input y for random coins ω. We say that an algorithm or a distribution is efficient if it runs in polynomial time in its input length (and, unless stated differently, we assume that efficient algorithms are probabilistic).

Signature Schemes. A signature scheme $\mathcal{S} = (\mathsf{SKGen}, \mathsf{Sig}, \mathsf{SVf})$ consists of efficient algorithms such that SKGen on input 1^n generates a key pair $(sk, pk) \leftarrow \mathsf{SKGen}(1^n)$, algorithm Sig for input sk and a message $m \in \{0,1\}^*$ outputs a signature $\sigma \leftarrow \mathsf{Sig}(sk, m)$, and algorithm SVf for input pk, m and σ returns a decision bit $d \leftarrow \mathsf{SVf}(pk, m, \sigma)$. Furthermore, for all security parameters $n \in \mathbb{N}$, all keys $(sk, pk) \leftarrow \mathsf{SKGen}(1^n)$, all messages $m \in \{0,1\}^*$ and all signatures $\sigma \leftarrow \mathsf{Sig}(sk, m)$ it holds $\mathsf{SVf}(pk, m, \sigma) = 1$.

A signature scheme $\mathcal{S}$ is *existentially unforgeable under adaptively chosen-message attacks* (or, for short, *unforgeable*) if for any efficient algorithm $\mathcal{A}$ the probability for $(sk, pk) \leftarrow \mathsf{SKGen}(1^n)$ and $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sig}(sk,\cdot)}(pk)$ such that $\mathsf{SVf}(pk, m^*, \sigma^*) = 1$ and m^* is not among the queries to oracle $\mathsf{Sig}(sk, \cdot)$, is negligible (as a function of n). We say that $\mathcal{S}$ is *strongly unforgeable* if we relax the requirement on the adversarial output (m^*, σ^*), such that $\mathsf{SVf}(pk, m^*, \sigma^*) = 1$ and m^* has never been answered with σ^* by oracle $\mathsf{Sig}(sk, \cdot)$, i.e., the message m^* may have been signed by $\mathsf{Sig}(sk, \cdot)$ previously but then the adversarial signature σ^* must be new.

Anonymous Signatures. For anonymity we adopt the strongest notion given by Yang et al. [9], called anonymity under chosen-message attacks. This notion basically says that no efficient algorithm $\mathcal{D}$ should be able to distinguish whether a message m (generated secretly according to a distribution $\mathcal{M}$) has been signed with secret key sk_0 or sk_1. This should even hold if $\mathcal{D}$ gets to learn other signatures for chosen messages. See [9] for a discussion of this notion.

In comparison to the original definition we consider here the most simple case of two users and public keys, respectively, among which $\mathcal{D}$ must distinguish (instead of polynomially many users). Security for the case of two users implies anonymity for polynomially many users, because the two "target keys" can

always be guessed among the polynomially many keys (with sufficiently large probability).

In addition, as for ring signatures [1] we also consider the notion of anonymity with respect to *full key exposure* where the signer's identity cannot be determined even if one knows the signing keys of the two users. This guarantees anonymity even if the adversary corrupts the users and gets to know the secret key.

Definition 1. *A signature scheme $\mathcal{S}$ is called* signer anonymous under adaptive chosen-message attacks *(or simply* anonymous*) with respect to distribution $\mathcal{M}$ if for any efficient algorithm $\mathcal{D}$ the random variables $\boldsymbol{Exp}^{anon,b}_{\mathcal{S},\mathcal{M},\mathcal{D}}(n)$ for $b = 0, 1$ are computationally indistinguishable:*

Experiment $\boldsymbol{Exp}^{anon,b}_{\mathcal{S},\mathcal{M},\mathcal{D}}(n)$*:*
let $(sk_0, pk_0) \leftarrow \mathsf{SKGen}(1^n)$ *and* $(sk_1, pk_1) \leftarrow \mathsf{SKGen}(1^n)$
sample $m \leftarrow \mathcal{M}(pk_b)$ *and compute* $\sigma \leftarrow \mathsf{Sig}(sk_b, m)$
let $d \leftarrow \mathcal{D}^{\mathsf{Sig}(sk_0,\cdot),\mathsf{Sig}(sk_1,\cdot)}(pk_0, pk_1, \sigma)$
output d

The scheme is called anonymous with respect to full key exposure *if the random variables are still computationally indistinguishable, even if $\mathcal{D}$ gets the secret keys sk_0, sk_1 as additional input.*

The definition above considers anonymity with respect to designated distributions $\mathcal{M}$, i.e., the signature scheme itself may depend on the distribution in question. Such schemes may be sufficient in some settings, but it often seems be desirable to have schemes which are anonymous with respect to any distributions from a larger class $\mathcal{C}_{\mathcal{M}}$, e.g., including all efficient distributions with non-trivial entropy. The definition extends straightforwardly to this case by demanding anonymity with respect to any distribution $\mathcal{M}$ from $\mathcal{C}_{\mathcal{M}}$. For the constructions we mostly focus on the case of designated distributions and briefly discuss how our solutions extend to classes of distributions.

3 Constructing Hash-and-Extractor Combinations

Recall from the introduction that our goal is to design a (probabilistic) randomness extractor whose output still looks random, even if one sees an additional hash value of the extractor's input. We first recall the two required primitives, hash functions and randomness extractors. Both algorithms will be randomized in the sense that they get an auxiliary random input and compute the output from the input and this random string, and the random string becomes part of the output (public randomness).

Hash Functions and Extractors. A (probabilistic) hash function $\mathcal{H} = (\mathsf{HKGen}, \mathsf{H})$ consists of efficient algorithms such that HKGen on input 1^n returns a key K and H on input a key K and a string $x \in \{0,1\}^{i(n)}$ picks a random string $r \leftarrow \{0,1\}^{t(n)}$ and outputs an image $y \leftarrow \mathsf{H}(K, x; r)$ (to which one appends the randomness r). The hash function $\mathcal{H}$ is called *collision-intractable* if for any

efficient algorithm $\mathcal{C}$ the probability that for $K \leftarrow \mathsf{HKGen}(1^n)$ and $(r, x, x') \leftarrow \mathcal{C}(K)$ it holds $x \neq x'$ but $\mathsf{H}(K, x; r) = \mathsf{H}(K, x'; r)$, is negligible (as a function of n). Note that we define such collisions x, x' with respect to the same random string r, as required for our applications.

We next define randomness extractors [6,5,8]. Recall that we want to combine a hash function and an extractor and we therefore extend the basic definition of extractors and allow the key generation algorithm of the extractor to depend on hash function keys. Namely, a (strong[3]) extractor $\mathcal{E} = (\mathsf{EKGen}, \mathsf{Ext})$ *associated to hash function* $\mathcal{H}$ consists of two probabilistic algorithms such that EKGen for input $K \leftarrow \mathsf{HKGen}(1^n)$ returns a random key $E \leftarrow \mathsf{EKGen}(K)$, and algorithm Ext for input E and $x \in \{0,1\}^{i(n)}$ picks a random string $u \leftarrow \{0,1\}^{d(n)}$ and outputs an $\ell(n)$-bit string $e \leftarrow \mathsf{Ext}(E, x; u)$ (to which one appends again the randomness u).

The extractor $\mathcal{E}$ (associated to $\mathcal{H}$) is called *pseudorandom* for distribution $\mathcal{X}$ if the following two random variables (one describing a hash value and the related extractor output, and the other one a hash value and an independent random output) are computationally indistinguishable:

- Let $K \leftarrow \mathsf{HKGen}(1^n)$, $x \leftarrow \mathcal{X}(1^n)$, $y \leftarrow \mathsf{H}(K, x; r)$, and $E \leftarrow \mathsf{EKGen}(K)$, $u \leftarrow \{0,1\}^{d(n)}$ and $e \leftarrow \mathsf{Ext}(E, x; u)$. Output the tuple $(K, r||y, E, u||e)$.
- Let $K \leftarrow \mathsf{HKGen}(1^n)$, $x \leftarrow \mathcal{X}(1^n)$, $y \leftarrow \mathsf{H}(K, x; r)$, and $E \leftarrow \mathsf{EKGen}(K)$, $u \leftarrow \{0,1\}^{d(n)}$ and $v \leftarrow \{0,1\}^{\ell(n)}$. Output the tuple $(K, r||y, E, u||v)$.

In the literature it is usually assumed that the extractor's output is statistically close to uniform. For our purpose it suffices that the output cannot be efficiently distinguished from random. This also requires a form of non-triviality of the distribution $\mathcal{X}$, usually demanding that the min-entropy $H_\infty(\mathcal{X}) = \min_x -(\log \mathrm{Prob}[\mathcal{X}(1^n) = x])$ of $\mathcal{X}$ is super-logarithmic (so called *well-spread* distributions). We also note that we get the regular definition of extractors by setting $K = 1^n$ and letting $\mathsf{H}(K, x; r)$ and r be the empty strings. In this case we drop the addendum "associated to $\mathcal{H}$" and simply speak of regular extractors.

Instantiations. As for the existence of such extractors we give two examples. Assume that we work in the random oracle model, for random function $H : \{0,1\}^* \to \{0,1\}^{\ell(n)}$. Define $H(0, \cdot)$ as the collision-intractable hash function. Then it is easy to see that $\mathsf{Ext}(\cdot) = H(1, \cdot)$ is a (deterministic) extractor (associated to $H(0, \cdot)$) which is pseudorandom for any fixed well-spread distribution $\mathcal{X}$. This is so because the super-logarithmic min-entropy of $\mathcal{X}$ prevents a distinguisher to query $H(0, \cdot)$ or $H(1, \cdot)$ about a randomly sampled and secret pre-image x, except with negligible probability, making the hash values independent and uniformly distributed.

To get a solution in the standard model, which is only slightly less efficient, assume that we have a 2-value perfectly one-way hash function (with public randomness) [2,3], i.e., where hash value pairs $(\mathsf{H}(K, x; r), \mathsf{H}(K, x; r'))$ of the same

[3] The term "strong" typically refers to extractors that give the auxiliary random input as part of the output. Since this is always the case here we usually do not mention this explicitly.

pre-image x are indistinguishable from hash value pairs $(\mathsf{H}(K,x;r),\mathsf{H}(K,x';r'))$ of independent pre-images x,x'. Formally, a *perfectly one-way hash function* (with respect to distribution $\mathcal{X}$) is a probabilistic collision-resistant hash function $\mathcal{H}$ such that the following random variables are computationally indistinguishable:

- Let $K \leftarrow \mathsf{HKGen}(1^n)$, $x \leftarrow \mathcal{X}(1^n)$ and $r,r' \leftarrow \{0,1\}^{t(n)}$. Compute $y \leftarrow \mathsf{H}(K,x;r)$ and $y' \leftarrow \mathsf{H}(K,x;r')$. Output the tuple (K,r,r',y,y').
- Let $K \leftarrow \mathsf{HKGen}(1^n)$, $x,x' \leftarrow \mathcal{X}(1^n)$ and $r,r' \leftarrow \{0,1\}^{t(n)}$. Compute $y \leftarrow \mathsf{H}(K,x;r)$ and $y' \leftarrow \mathsf{H}(K,x';r')$. Output the tuple (K,r,r',y,y').

Very efficient perfectly one-way hash functions (for any fixed well-spread distribution $\mathcal{X}$) can be derived from any regular collision-resistant hash function [3].

The perfectly one-way hash function basically allows us to compute two hashes of the same input but such the hash values appear to originate from independent inputs. Hence, if we now take the first hash value for the signing process and apply a regular extractor $\mathcal{E}_{\mathsf{reg}}$ to the second hash value, the result will almost look as if we have run both algorithms on independent inputs.

On a technical side, we note that the regular extractor $\mathcal{E}_{\mathsf{reg}}$ (not associated to a hash function) gets as input a hash value sampled according to the distribution which picks $x \leftarrow \mathcal{X}(1^n)$, $K \leftarrow \mathsf{HKGen}(1^n)$ and $r \leftarrow \{0,1\}^{t(n)}$ and which returns $\mathsf{H}(K,x;r)$. We denote this distribution by $\mathcal{H}(\mathcal{X})$, and we say that such an extractor is pseudorandom with respect to $\mathcal{H}(\mathcal{X})$ if the extractor's output is indistinguishable from random, even when given K and r in clear.

We remark that the distribution $\mathcal{H}(\mathcal{X})$ "essentially preserves" the entropy of the input distribution $\mathcal{X}$. That is, if $\mathcal{X}$ is well-spread and efficient, then with overwhelming probability over the choice $K \leftarrow \mathsf{HKGen}(1^n)$ and $r \leftarrow \{0,1\}^{t(n)}$, the min-entropy of $\mathsf{H}(K,\mathcal{X}(1^n);r)$ remains super-logarithmically. Else, for a random input key K, sampling $r \leftarrow \{0,1\}^{t(n)}$ and $x,x' \leftarrow \mathcal{X}(1^n)$ would yield a non-trivial collision with noticeable probability (i.e., because of the min-entropy of $\mathcal{X}$ the values x,x' will be different with overwhelming probability, whereas the hash values collide with noticeable probability by presumption about the entropy loss of H). The entropy of $\mathcal{H}(\mathcal{X})$ can be determined explicitly in terms of the entropy of $\mathcal{X}$ and the "entropy loss" of $\mathcal{H}$. In particular, if we use the construction of $\mathcal{H}$ via regular collision-resistant hash functions [3] then a (fixed) min-entropy $\lambda(n)$ of $\mathcal{X}$ yields a distribution $\mathcal{H}(\mathcal{X})$ with min-entropy at least $\lambda(n)/6+3$.

Recall that we usually consider an extractor $\mathcal{E}_{\mathsf{reg}}$ as the composition of a statistical randomness extractors, producing output which is statistically close to the uniform distribution, and a cryptographically-secure pseudorandom generator G. Note that the pseudorandom generator G needs to be able to stretch the short random input of, say, super-logarithmically many bits, into a pseudorandom output of polynomially many bits. Whether G achieves such an expansion factor or not depends on the concrete implementation. But we can safely assume for any pseudorandom generator that, if G takes n^c inputs bits (for some constant $c>0$), it can stretch this input to any output of polynomial size. Thus, using the [3] perfectly one-way hash function, we get a secure construction if the starting distribution has min-entropy $\Omega(n^c)$. Below, however, we still state our

result in its general form, assuming that we have a good extractor with respect to the distribution $\mathcal{H}(\mathcal{X})$.

Construction 1. *Let $\mathcal{H}$ be a hash function and $\mathcal{E}_{reg}$ be a regular extractor (for distribution $\mathcal{H}(\mathcal{X})$). Define extractor $\mathcal{E} = (\mathsf{EKGen}, \mathsf{Ext})$ associated to $\mathcal{H}$ as follows:*

- *The key generator* EKGen *on input K generates $E_{reg} \leftarrow \mathsf{EKGen}_{reg}(1^n)$ and outputs $E \leftarrow (E_{reg}, K)$.*
- *The extraction procedure* Ext *on input E, $x \in \{0,1\}^{i(n)}$ and $u = r||u_{reg} \in \{0,1\}^{t(n)+d(n)}$ computes $e \leftarrow \mathsf{Ext}_{reg}(E_{reg}, \mathsf{H}(K,x;r); u_{reg})$ and outputs e.*

We next prove that the derived extractor is pseudorandom:

Proposition 1. *Let $\mathcal{H}$ be a perfectly one-way hash function (for distribution $\mathcal{X}$) and $\mathcal{E}_{reg}$ be a pseudorandom extractor (for distribution $\mathcal{H}(\mathcal{X})$). Then $\mathcal{E}$ in Construction 1 is an extractor associated to $\mathcal{H}$ which is pseudorandom (with respect to distribution $\mathcal{X}$).*

Proof. Consider the random variable

> Let $K \leftarrow \mathsf{HKGen}(1^n)$, $x \leftarrow \mathcal{X}(1^n)$ and $r, r' \leftarrow \{0,1\}^{t(n)}$. Let $E_{\mathsf{reg}} \leftarrow \mathsf{EKGen}_{\mathsf{reg}}(1^n)$ and $u_{\mathsf{reg}} \leftarrow \{0,1\}^{d(n)}$. Compute $y \leftarrow \mathsf{H}(K,x;r)$ and $e_{\mathsf{reg}} \leftarrow \mathsf{Ext}_{\mathsf{reg}}(E_{\mathsf{reg}}, \mathsf{H}(K,x;r'); u_{\mathsf{reg}})$. Output $(K, r||y, (K, E_{\mathsf{reg}}), r'||u_{\mathsf{reg}}||e_{\mathsf{reg}})$.

which describes the output of our extractor $\mathcal{E}$ for a random sample x (together with the additional hash value). By the computational indistinguishability of the perfectly one-way hash function this variable is indistinguishable from the following random variable, where we pick an independent input x' for the "extractor's hash value":

> Let $K \leftarrow \mathsf{HKGen}(1^n)$, $x, x' \leftarrow \mathcal{X}(1^n)$ and $r, r' \leftarrow \{0,1\}^{t(n)}$. Let $E_{\mathsf{reg}} \leftarrow \mathsf{EKGen}_{\mathsf{reg}}(1^n)$ and $u_{\mathsf{reg}} \leftarrow \{0,1\}^{d(n)}$. Compute $y \leftarrow \mathsf{H}(K,x;r)$ as well as $e_{\mathsf{reg}} \leftarrow \mathsf{Ext}_{\mathsf{reg}}(E_{\mathsf{reg}}, \mathsf{H}(K,x';r'); u_{\mathsf{reg}})$. Output $(K, r||y, (K, E_{\mathsf{reg}}), r'||u_{\mathsf{reg}}||e_{\mathsf{reg}})$.

It next follows from the pseudorandomness of the extractor $\mathcal{E}_{\mathsf{reg}}$ that the previous random variable with independent inputs x, x' is indistinguishable from the following random variable, where we replace the extractor's output by a random value:

> Let $K \leftarrow \mathsf{HKGen}(1^n), x \leftarrow \mathcal{X}(1^n), r, r' \leftarrow \{0,1\}^{t(n)}$ and $E_{\mathsf{reg}} \leftarrow \mathsf{EKGen}_{\mathsf{reg}}(1^n)$. Pick $u_{\mathsf{reg}} \leftarrow \{0,1\}^{d(n)}$ as well as $v_{\mathsf{reg}} \leftarrow \{0,1\}^{\ell(n)}$. Compute $y \leftarrow \mathsf{H}(K,x;r)$. Output $(K, r||y, (K, E_{\mathsf{reg}}), r'||u_{\mathsf{reg}}||v_{\mathsf{reg}})$.

The indistinguishability of this final variable from the starting case proves the claim. □

Our extractors so far work for specific distributions $\mathcal{H}(\mathcal{X})$. In particular, they depend (only) on the knowledge of the min-entropy of distribution $\mathcal{H}(\mathcal{X})$. Hence, such extractors also work with classes $\mathcal{C}_{\mathcal{H}(\mathcal{X})}$ of distributions, as long as any such distribution $\mathcal{H}(\mathcal{X}) \in \mathcal{C}_{\mathcal{H}(\mathcal{X})}$ obeys a fixed lower bound $\lambda(n)$ on the

min-entropy (e.g., $\lambda(n) = \omega(\log n)$ if one assumes a strong pseudorandom generator G, or $\lambda(n) = n^c$ for some constant $c > 0$ if we assume standard pseudorandom generators).

4 Constructing Anonymous Signatures

With the primitives of the previous section we can now give the formal description of our transformation from any regular signature scheme to an anonymous one. We assume without loss of generality that the signature size is bounded by some publicly known polynomial $\ell(n)$ (such a bound exists by the limited running time of the signature algorithms), and that the extractor $\mathsf{Ext}(E, m; u)$ produces $\ell(n)$-bit outputs e. Below, if we mask the signature σ with e it is understood that the signature is padded with zeros if necessary, i.e., $\sigma \oplus e = (\sigma || 0^{\ell - |\sigma|}) \oplus e$.

Note that our construction of the extractor (associated to a hash function) requires that the message has some fixed input length $i(n)$ (which nonetheless can depend on the security parameter). We therefore assume that messages to be signed have exactly $i(n)$ bits, and that the distribution $\mathcal{M}$ itself is defined over such bit strings. This requirement can be implemented by hashing longer messages first with some collision-intractable hash function. Accordingly, we have to consider the distribution of hashed messages then (which, by the collision-intractability, is also well-spread if the original message distribution is).

Construction 2. *Let $\mathcal{S}$ be a signature scheme, let $\mathcal{H}$ be a hash function and $\mathcal{E}$ be an extractor (associated to $\mathcal{H}$). Define the following signature scheme $\mathcal{S}' = (\mathsf{SKGen}', \mathsf{Sig}', \mathsf{SVf}')$:*

- *The key generation algorithm $\mathsf{SKGen}'(1^n)$ runs $\mathsf{SKGen}(1^n)$ to get a key pair (sk, pk). It also runs $\mathsf{HKGen}(1^n)$ to generate a key K for the hash function, as well as a key $E \leftarrow \mathsf{EKGen}(K)$ for the extractor. It outputs $sk' \leftarrow (sk, K, E)$ and $pk' \leftarrow (pk, K, E)$.*
- *The signing algorithm $\mathsf{Sig}'(sk, m)$ samples $r \leftarrow \{0,1\}^{t(n)}$ and $u \leftarrow \{0,1\}^{d(n)}$, computes a signature $\sigma \leftarrow \mathsf{Sig}(sk, \mathsf{H}(K, m; r))$ as well as $\tau \leftarrow \sigma \oplus \mathsf{Ext}(E, m; u)$ and finally outputs $\sigma' \leftarrow \tau || r || u$.*
- *The verification algorithm $\mathsf{SVf}'(pk', m, \sigma')$ for $\sigma' = \tau || r || u$ first computes $\sigma \leftarrow \tau \oplus \mathsf{Ext}(E, m; u)$ and then outputs $\mathsf{SVf}(pk, \mathsf{H}(K, m; r), \sigma)$.*

Proposition 2. *Let $\mathcal{S}$ be an unforgeable signature scheme, let $\mathcal{H}$ be a collision-intractable hash function and $\mathcal{E}$ be an extractor (associated to $\mathcal{H}$). Then $\mathcal{S}'$ in Construction 2 is an unforgeable signature scheme.*

Note that we do not need to assume that $\mathcal{E}$ is a good extractor for proving unforgeability. This property will only be required for the anonymity proof.

Proof. We show that we can transform any forger $\mathcal{A}'$ on the derived scheme $\mathcal{S}'$ into one on the original scheme, essentially preserving the running time and success probability of $\mathcal{A}'$. We assume without loss of generality that $\mathcal{A}'$ always outputs a new message m^* in the forgery attempt (i.e., such that m^* has never been signed by the signing oracle before).

For transforming the attacker $\mathcal{A}'$ into one for the underlying signature scheme we let $\mathcal{A}^{\mathsf{Sig}(sk,\cdot)}(pk)$ run a black-box simulation of $\mathcal{A}'$ for input $pk' = (pk, K, E)$ where keys K and E are generated by $\mathcal{A}$ by running $\mathsf{HKGen}(1^n)$ and $\mathsf{EKGen}(K)$. Then, $\mathcal{A}$ simulates the signing oracle Sig' for $\mathcal{A}'$ as follows:

> Each time $\mathcal{A}'$ submits a message $m \in \{0,1\}^{i(n)}$ to its (putative) signing oracle attacker $\mathcal{A}$ first picks $r \leftarrow \{0,1\}^{t(n)}$ and $u \leftarrow \{0,1\}^{d(n)}$ and forwards $\mathsf{H}(K, m; r)$ to its oracle Sig to get a signature σ. Algorithm $\mathcal{A}$ next computes $\tau \leftarrow \sigma \oplus \mathsf{Ext}(E, m; u)$ and $\sigma' \leftarrow \tau||r||u$ and returns σ' on behalf of Sig' to attacker $\mathcal{A}'$.

When $\mathcal{A}'$ eventually outputs a forgery attempt $(m^*, \tau^*||r^*||u^*)$ we let $\mathcal{A}$ compute $\sigma^* \leftarrow \tau^* \oplus \mathsf{Ext}(E, m^*; u^*)$ and let it return $(\mathsf{H}(K, m^*; r^*), \sigma^*)$.

It is easy to see that the simulation above perfectly mimics an actual attack. Hence, in the simulation above $\mathcal{A}'$ outputs a successful forgery with the same probability as in an attack on the derived scheme. By the collision-intractability of $\mathcal{H}$ we can also conclude that, with overwhelming probability, $\mathsf{H}(K, m^*; r^*)$ is different from all hash values that $\mathcal{A}$ has passed to its oracle Sig previously (else, since m^* is different from all previously signed messages, it would be straightforward to derive a successful collision-finder against the hash function). It follows that, if $\mathcal{A}'$ produces a successful forgery against the derived scheme with noticeable probability, then so does $\mathcal{A}$ in the attack on the underlying signature scheme. □

Theorem 3. *Let $\mathcal{S}$ be a signature scheme, let $\mathcal{H}$ be a hash function and $\mathcal{E}$ be an extractor (associated to $\mathcal{H}$) which is pseudorandom with respect to distribution $\mathcal{M}$. Then $\mathcal{S}'$ in Construction 2 is an anonymous signature scheme (with respect to $\mathcal{M}$). It is even anonymous with respect to full key exposure.*

Here we merely require that the extractor is pseudorandom; the original signature scheme and the hash function only need to be efficient. This fact also shows anonymity against full key exposure.

Proof. Fix an arbitrary attacker $\mathcal{D}$ against the (basic) anonymity property and some distribution $\mathcal{M}$. We need to show that the outputs of the random variables $\mathbf{Exp}^{\text{anon},b}_{\mathcal{S}',\mathcal{M},\mathcal{D}}(n)$ for $b = 0, 1$ are indistinguishable. In the sequel we also fix the bit b.

In experiment $\mathbf{Exp}^{\text{anon},b}_{\mathcal{S}',\mathcal{M},\mathcal{D}}(n)$ we now change the way the challenge signature for $m \leftarrow \mathcal{M}(pk_b)$ is computed as follows. As before we sample $r \leftarrow \{0,1\}^{t(n)}$ and $u \leftarrow \{0,1\}^{d(n)}$ and compute a signature $\sigma \leftarrow \mathsf{Sig}(sk, \mathsf{H}(K, m; r))$. But now we let $\tau \leftarrow \sigma \oplus v$ for an independent random value v, instead of computing $\tau \leftarrow \sigma \oplus \mathsf{Ext}(E, m; u)$ as before. We output $\sigma' \leftarrow \tau||r||u$ for the modified value τ. We denote this experiment by $\mathbf{Exp}^{\text{mod-anon},b}_{\mathcal{S}',\mathcal{M},\mathcal{D}}(n)$.

It follows from the pseudorandomness of the extractor (associated to $\mathcal{H}$) that the way we compute the signature in the modified experiment cannot change the output behavior of experiment $\mathbf{Exp}^{\text{anon},b}_{\mathcal{S}',\mathcal{M},\mathcal{D}}(n)$ noticeably. Else it would be easy to construct an algorithm $\mathcal{B}_b$ (with b hardwired into its description) which gets

$(K, r||y, E, u||v)$ for $v = \mathsf{Ext}(E, m; u)$ or random v as input, and which successfully distinguishes these two cases (by simulating $\mathcal{D}$ in experiment $\mathbf{Exp}^{\text{anon},b}_{\mathcal{S}',\mathcal{M},\mathcal{D}}(n)$ for fixed bit b and using the given values to prepare the challenge signature). Hence, $\mathbf{Exp}^{\text{anon},b}_{\mathcal{S}',\mathcal{M},\mathcal{D}}(n)$ and $\mathbf{Exp}^{\text{mod-anon},b}_{\mathcal{S}',\mathcal{M},\mathcal{D}}(n)$ are computationally indistinguishable for both $b = 0, 1$.

But in experiment $\mathbf{Exp}^{\text{mod-anon},b}_{\mathcal{S}',\mathcal{M},\mathcal{D}}(n)$ the signature $\tau||r||u$ for $\tau \leftarrow \sigma \oplus v$ is now independently distributed of σ and it follows that the output $\mathbf{Exp}^{\text{mod-anon},b}_{\mathcal{S}',\mathcal{M},\mathcal{D}}(n)$ for both $b = 0, 1$ is identical. In conclusion, the random variables $\mathbf{Exp}^{\text{anon},0}_{\mathcal{S}',\mathcal{M},\mathcal{D}}(n)$ and $\mathbf{Exp}^{\text{anon},1}_{\mathcal{S}',\mathcal{M},\mathcal{D}}(n)$ must be computationally indistinguishable.

Note that the proof still works if $\mathcal{D}$ knows the signing keys since we merely need the pseudorandomness of the extractor. This shows that the scheme remains anonymous with respect to full key exposure. □

Some remarks follow. First, note that our proof actually shows that signatures in our scheme are *pseudorandom*, even when knowing the signing keys. Clearly, such pseudorandom signatures imply anonymity (with respect to full key exposure), because it is hard to tell such signatures apart from random strings.

Second, we can modify our signature scheme to get a strongly unforgeable scheme, given that the starting scheme is strongly unforgeable. To this end we let the signature algorithm sign $\mathsf{H}(K, m; r)||r||u$ instead of the hash value only. It follows similarly to the unforgeability proof above that the scheme is strongly unforgeable.

As a proof outline of the strong unforgeability of our modified scheme, assume that the adversary outputs a valid forgery $(m^*, \tau^*||r^*||u^*)$ such that the values (m^*, r^*, u^*) have never appeared before. Then this would contradict the unforgeability of the original signature scheme. Assume, on the other hand, that such values have appeared before (in which case there is a unique signature reply $\tau||r^*||u^*$ in which they appear, with overwhelming probability over the random choices of r, u in the signing process). This implies that the adversary has only modified τ to a different τ^*. But then the validity of the forgery attempt would imply that $\sigma^* \leftarrow \tau^* \oplus \mathsf{Ext}(E, m^*; u^*)$ is different from σ in the original signature, and that this value σ^* together with "message" $\mathsf{H}(K, m^*; r^*)||r^*||u^*$ contradicts the strong unforgeability of the underlying scheme. And this modified scheme is still anonymous with respect to full key exposure.

Third, we finally notice that our result extends to classes $\mathcal{C}_{\mathcal{M}}$ of message distributions, if the underlying extractor is pseudorandom with respect to this class. Hence, we get a provably secure construction assuming that $\mathcal{C}_{\mathcal{M}}$ only contains distributions of min-entropy at least $\lambda(n)$, where the fixed bound $\lambda(n)$ depends on the extractor in question (see Section 3).

References

1. Adam Bender, Jonathan Katz, and Ruggero Morselli. *Ring Signatures: Stronger Definitions, and Constructions Without Random Oracles.* Theory of Cryptography Conference (TCC) 2006, Volume 3876 of Lecture Notes in Computer Science, pages 60–79. Springer-Verlag, 2006.

2. Ran Canetti. *Towards Realizing Random Oracles: Hash Functions That Hide All Partial Information.* Advances in Cryptology — Crypto'97, Volume 1294 of Lecture Notes in Computer Science, pages 455–469. Springer-Verlag, 1997.
3. Ran Canetti, Daniele Micciancio, and Omer Reingold. *Perfectly One-Way Probabilistic Hash Functions.* Proceedings of the Annual Symposium on the Theory of Computing (STOC)'98, pages 131–140. ACM Press, 1998.
4. Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. *Anonymous Identification in Ad Hoc Groups.* Advances in Cryptology — Eurocrypt 2004, Volume 3027 of Lecture Notes in Computer Science, pages 609–626. Springer-Verlag, 2004.
5. Noam Nisan and Amnon Ta-Shma. *Extracting Randomness: A Survey and New Constructions. Journal of Computer and System Science*, 58(1):148–173, 1999.
6. Noam Nisan and David Zuckerman. *Randomness is Linear in Space. Journal of Computer and System Science*, 52(1):43–52, 1996.
7. Ronald Rivest, Adi Shamir, and Yael Tauman. *How to Leak a Secret.* Advances in Cryptology — Asiacrypt 2001, Volume 2248 of Lecture Notes in Computer Science, pages 552–565. Springer-Verlag, 2001.
8. Ronen Shaltiel. *Recent Developments in Extractors — a Survey. Bulletin of the European Association for Theoretical Computer Science*, 77:67–95, 2002.
9. Guomin Yang, Duncan Wong, Xiaotie Deng, and Huaxiong Wang. *Anonymous Signature Schemes.* Public-Key Cryptography (PKC) 2006, Volume 3958 of Lecture Notes in Computer Science, pages 347–363. Springer-Verlag, 2006.