An Efficient CDH-based Signature Scheme With a Tight Security Reduction[⋆]

Benoît Chevallier-Mames[1,2]

[1] Gemplus, Card Security Group
La Vigie, Avenue du Jujubier, ZI Athélia IV,
F-13705 La Ciotat Cedex, France

[2] École Normale Supérieure
Département d'Informatique
45 rue d'Ulm, F-75230 Paris 05, France
benoit.chevallier-mames@gemplus.com

Abstract. At Eurocrypt '03, Goh and Jarecki showed that, contrary to other signature schemes in the discrete-log setting, the *EDL* signature scheme has a *tight* security reduction, namely to the Computational Diffie-Hellman (CDH) problem, in the Random Oracle (RO) model. They also remarked that *EDL* can be turned into an off-line/on-line signature scheme using the technique of Shamir and Tauman, based on chameleon hash functions.

In this paper, we propose a new signature scheme that also has a tight security reduction to CDH but whose resulting signatures are smaller than *EDL* signatures. Further, similarly to the Schnorr signature scheme (but contrary to *EDL*), our signature is naturally efficient on-line: no additional trick is needed for the off-line phase and the verification process is unchanged.

For example, in elliptic curve groups, our scheme results in a 25% improvement on the state-of-the-art discrete-log based schemes, with the same security level. This represents to date the most efficient scheme of any signature scheme with a tight security reduction in the discrete-log setting.

Keywords: Public-key cryptography, signature schemes, discrete logarithm problem, Diffie-Hellman problem, *EDL*.

1 Introduction

In a signature scheme, a party, called *signer*, generates a signature using his own private key so that any other party, called *verifier*, can check the validity of the signature using the corresponding signer's public-key. Following the IEEE P1363 standard [P1363], there are two main settings commonly used to build signature schemes: the integer factorization setting and the discrete logarithm setting.

[⋆] This is the full version of [Che05].

A signature scheme should protect against *impersonation* of parties and *alteration* of messages. Informally, the security is assessed by showing that if an adversary can violate one of the two previous properties then the same adversary can also break the underlying cryptographic problem — for example, the integer factorization problem, the RSA problem [RSA78], the discrete logarithm problem or the Diffie-Hellman problem [DH76]. As the cryptographic problem is supposed to be intractable, no such adversary exists. This methodology for assessing the security is called *security reduction.* The "quality" of the reduction is given by the success probability of the adversary against a signature scheme to break the underlying intractable problem. A security reduction is said *tight* when this success probability is close to 1; otherwise it is said *close* or *loose* [MR02]. This notion of tightness is very important, and allows to distinguish between *asymptotic* security and *exact* security, the first one meaning that a scheme is secure *for sufficiently large parameters*, while the second one means that the underlying cryptographic problem is almost as hard to solve as the scheme to break.

The first efficient signature scheme *tightly* related to the RSA problem is due to Bellare and Rogaway [BR96]. The security stands in the Random Oracle (RO) model [BR93] where hash functions are idealized as random oracles. Their scheme, called RSA-PSS, appears in most recent cryptographic standards. Other RSA-based signature schemes shown to be secure in the standard model include [GHR99] and [CS00].

Amongst the signature schemes based on the discrete logarithm problem (or on the Diffie-Hellman problem), we quote the ElGamal scheme [ElG85], the Schnorr scheme [Sch91], and the Girault-Poupard-Stern scheme [Gir91,PS98]. The security of these schemes is assessed (in the RO model) thanks to the forking lemma by Pointcheval and Stern [PS96]. Basically, the idea consists in running the adversary twice with different hash oracles so that it eventually gets two distinct valid forgeries on the same message. The disadvantage of the forking lemma technique is that the so-obtained security reductions are loose.

Even if the security reductions are loose, those signature schemes present the nice feature that there are very efficient *on-line* [FS87] compared to RSA-based signature schemes. In the off-line phase, the signer precomputes a quantity (independent of the message) called a *coupon* that will be used in the on-line phase to produce very quickly a signature on an arbitrary message.

To date, the only signature scheme whose security is tightly related to the discrete logarithm problem or to the Diffie-Hellman problem (in the RO model) is *EDL*, a scheme independently considered in [CP92] and [JS99]. Indeed, at Eurocrypt '03, Goh and Jarecki [GJ03] showed that the security of *EDL* can be reduced in a tight way to the Computational Diffie-Hellman (CDH) problem. Its on-line version as suggested in [GJ03] requires the recent technique by Shamir and Tauman [ST01] based on chameleon hash functions [KR00] and so is not as efficient as the aforementioned signature schemes: the resulting signatures are longer and the verification is slower.

It is to note that *EDL* was recently modified by Katz and Wang [KW03] into a scheme with shorter signatures and a tight security reduction but on a stronger assumption, namely the Decisional Diffie-Hellman (DDH) assumption. In the same paper, Katz and Wang also proposed an improvement to *EDL*, that uses a single bit instead of a long random, and which has a tight reduction to the CDH problem. The cost of this nice improvement is simply a decrease of the security parameter of one bit.

To finalize the related work part, we stress that the shortest signature scheme that is known today is a scheme of Boneh, Lynn and Shacham [BLS04]. This scheme is loosely related to the CDH problem, but gives very short signatures, as it consists in only one single group element. However, this scheme is limited to certain elliptic and hyper-elliptic curve groups, and so less general than *EDL*. Furthermore, the on-line version of the Boneh-Lynn-Shacham signature scheme requires the technique by Shamir and Tauman, which doubles the size of the signature, and hence is less interesting.

Our contribution. In this paper, we firstly review the definition of *EDL*, its proof by Goh and Jarecki, and the scheme of Katz and Wang. Secondly, we propose a new signature scheme which, similarly to *EDL*, features a *tight* security reduction relatively to the CDH problem but whose resulting signatures are smaller than *EDL* signatures. Furthermore, contrary to *EDL*, no additional trick is needed to turn our signature scheme in an off-line/on-line version.

Notably, in elliptic curve settings, our scheme supersedes other discrete logarithm based schemes with same security level, as it uses signatures that are 25% smaller.

Organization of the paper. The rest of this paper is organized as follows. In the next section, we give some background on signature schemes and provide a brief introduction to "provable" security. Then, in Section 3, we review the *EDL* signature scheme and its proof by Goh and Jarecki. Section 4 is the core of our paper. We describe our signature scheme, prove that its security is tightly related to CDH in the RO model and show how it outperforms *EDL*. Finally, we conclude in Section 5.

2 Definitions

In this section, we remind some background on signature schemes and on their security. We also define the Diffie-Hellman and the discrete logarithm problems. We then provide a brief introduction to provable security. Finally, we review the concept of *on-the-fly* signatures.

2.1 Signature Schemes

A signature scheme $\textsc{Sig} = (\textsc{GenKey}, \textsc{Sign}, \textsc{Verify})$ is defined by the three following algorithms:

- The *key generation algorithm* $\textsc{GenKey}$. On input 1^k, algorithm $\textsc{GenKey}$ produces a pair $(\mathsf{pk}, \mathsf{sk})$ of matching public (verification) and private (signing) keys.
- The *signing algorithm* $\textsc{Sign}$. Given a message m in a set of messages $\mathcal{M}$ and a pair of matching public and private keys $(\mathsf{pk}, \mathsf{sk})$, $\textsc{Sign}$ produces a signature σ. The signing algorithm can be probabilistic.
- The *verification algorithm* $\textsc{Verify}$. Given a signature σ, a message $m \in \mathcal{M}$ and a public key pk, $\textsc{Verify}$ 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, Micali and Rivest [GMR84,GMR88]. It is now customary to ask for the impossibility of existential forgeries, even against adaptive chosen-message adversaries:

- An *existential forgery* is a new message-signature pair, valid and generated by the adversary. The corresponding security notion 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 (EUF-CMA). This is measured by the following success probability, which should be negligibly small, for any adversary $\mathcal{A}$ which outputs a valid signature σ on a message m that was never submitted to the signature oracle,[3] within a "reasonable" bounded running-time and with at most q_s signature queries to the signature oracle:

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

In the *random oracle model* [BR93], adversary $\mathcal{A}$ has also access to a hash oracle: $\mathcal{A}$ is allowed to make at most q_h queries to the hash oracle.

2.2 The Diffie-Hellman and the Discrete Logarithm Problems

The security of signature schemes relies on problems that are supposed intractable, such as the *Diffie-Hellman problem* [DH76] or the *discrete logarithm problem.*

[3] When the signature generation is not deterministic, several signatures may correspond to the same message. In this case, we do not consider the attacker successful when it outputs a second signature on a message already submitted to the signature oracle. Being given a message-signature pair (m, σ), providing a second signature σ' on the same message m is captured by the adversarial goal of *malleability* [SPM+02].

Let $\mathbb{G}$ be a (multiplicatively written) abelian group. Given an element $g \in \mathbb{G}$ of prime order q, we let $G_{g,q} \subseteq \mathbb{G}$ denote the cyclic group generated by g, *i.e.*, $G_{g,q} = \{g^i, i \in \mathbb{Z}_q\}$.

Let x be a random number in $\mathbb{Z}_q$. Define $y = g^x$. Being given (g, y), the discrete logarithm problem in $G_{g,q}$ is defined as finding the value of x. In this paper, the discrete logarithm of y w.r.t. g will be denoted as $DL_g(y) = x$. On the other hand, being given (g, y, g^a), for an unknown random number a in $\mathbb{Z}_q$, the (computational) Diffie-Hellman problem is defined as returning $g^{ax} = y^a$.

For cryptographic applications, group $G_{g,q}$ is chosen so that the problems are (supposed) hard. A classical example is to choose $G_{g,q} \subseteq {\mathbb{F}_p}^*$, where q divides $(p-1)$. Another widely used group family is the one of elliptic curves over finite fields [Mil85,Kob87,BSS99].

There are plenty of such signature schemes, including the schemes by ElGamal [ElG85], Girault-Poupard-Stern [Gir91,PS98], Schnorr [Sch91], and particulary the one we are interested in this paper, the *EDL* scheme [CP92,JS99,GJ03].

2.3 Security Reduction and Provable Security

Today, schemes are "proved" secure, using what is called a *reduction.* For this reason, some authors prefer to use the term of reductionist security (*e.g.,* [KM04]) instead of provable security.

Basically, the idea is to prove that a scheme is secure by exhibiting a machine (the so-called reduction) that uses a chosen-message attacker on a given signature scheme, in order to solve a hard cryptographic problem. In the standard model, the attacker is used by simulating signature queries on q_s chosen-messages. In addition, in the random oracle mode, the simulator also simulates hash queries on q_h chosen data.

Two classes of provably secure signature schemes can be distinguished. The first class of provable signature schemes proposes reductions that are said *loose*, as they can turn an attacker into a machine to solve the cryptographic problem asymptotically. The second class of provable signature schemes features so-called *tight* reductions, using the attacker to solve the problem with almost the same probability.

Of course, tightly secure schemes are the preferred ones, but there are just few of them. Notably, RSA-PSS and its derivatives are tightly related to the RSA problem [RSA78,BR96,Cor02], and Rabin-PSS is equivalent to the factorisation problem [Rab79]. For a long time, no *tightly* secure schemes were known, based on the Diffie-Hellman or discrete logarithm problems, but only loosely secure schemes, as their security was shown thanks to the *forking lemma* technique by Pointcheval and Stern [PS96]. Proved recently at Eurocrypt '03, the *EDL* scheme is the first tight secure scheme, based on the computational Diffie-Hellman problem.

2.4 Signature with Coupons

Some signature schemes have the nice feature that one can precompute (off-line) some quantities, independent from the messages, called *coupons*, and use them in a very fast way to generate signatures once the message is received [FS87]. Such signature schemes are also known as *on-the-fly* signature schemes.

This coupon technique is very useful, especially in constrained environments such as smart cards and finds numerous applications. Most signature schemes based on discrete logarithm or Diffie-Hellman problems allow the use of coupons. However, as previously explained, they do not offer a tight security reduction. To our knowledge, the only exception is the *EDL* signature scheme using a technique proposed by Shamir and Tauman, based on chameleon hashes by Krawczyk and Rabin [ST01,KR00]. However, this use of chameleon hashes is at the price of a slower verification, as the verifier must compute chameleon hashes (which are multi-exponentiations) before verifying the signature.

3 The *EDL* Signature

3.1 The Scheme

The *EDL* signature scheme, independently proposed in [CP92,JS99], is defined as follows.

Global set-up: Let ℓ_p, ℓ_q, and ℓ_r denote security parameters.[4] Let also a cyclic group $G_{g,q}$ of order q, generated by g, where q is a ℓ_q-bit prime and the representation of the elements of $G_{g,q}$ is included in $\{0,1\}^{\ell_p}$. Finally, let two hash functions, $\mathcal{H} : \mathcal{M} \times \{0,1\}^{\ell_r} \to G_{g,q}$ and $\mathcal{G} : (G_{g,q})^6 \to \mathbb{Z}_q$.

Key generation: The private key is a random number $x \in \mathbb{Z}_q$. The corresponding public key is $y = g^x$.

Signature: To sign a message $m \in \mathcal{M}$, one first randomly chooses $r \in \{0,1\}^{\ell_r}$, and computes $h = \mathcal{H}(m,r)$ and $z = h^x$. Follows a proof of logarithm equality that $DL_h(z) = DL_g(y)$: for a random number $k \in \mathbb{Z}_q$, one computes $u = g^k$, $v = h^k$, $c = \mathcal{G}(g,h,y,z,u,v)$ and $s = k + cx \bmod q$. The signature on m is $\sigma = (z,r,s,c)$.

Verification: To verify a signature $\sigma = (z,r,s,c) \in G_{g,q} \times \{0,1\}^{\ell_r} \times (\mathbb{Z}_q)^2$ on a message $m \in \mathcal{M}$, one computes $h' = \mathcal{H}(m,r)$, $u' = g^s\, y^{-c}$ and $v' = h'^s\, z^{-c}$. The signature σ is accepted iff $c = \mathcal{G}(g,h',y,z,u',v')$.

In *EDL*, the only quantity that can be precomputed in off-line signature phase is u. The on-line part is so two hash function evaluations plus two modular exponentiations.

[4] For normal use-cases, $\ell_r \leq \ell_q$.

3.2 Security of *EDL*

In this section, we reduce the security of *EDL* to the security of the computational Diffie-Hellman problem. The proof basically follows the one originally presented in [GJ03] by showing that the *EDL* scheme is a proof that $DL_h(z) = DL_g(y) = x$.

Theorem 1 ([GJ03]). *Let $\mathcal{A}$ be an adversary which can produce, with success probability ε, an existential forgery under a chosen-message attack within time τ, after q_h queries to the hash oracles and q_s queries to the signing oracle, in the random oracle model. Then the computational Diffie-Hellman problem can be solved with success probability ε' within time τ', with*

$$\varepsilon' \geq \varepsilon - q_s \left(\frac{q_s + q_h}{q^2} + \frac{q_s + q_h}{2^{\ell_r}} \right) - \frac{q_h}{q}$$

and

$$\tau' \lesssim \tau + (6 q_s + q_h) \tau_0$$

where τ_0 is the time for an exponentiation in $G_{g,q}$.

Proof. We are given a group $G_{g,q}$ and a CDH challenge (g, g^x, g^a). We will use an attacker $\mathcal{A}$ against the *EDL* signature scheme to solve this challenge, *i.e.*, to find g^{ax}. Our attacker $\mathcal{A}$, after $q_{\mathcal{H}}$ (resp. $q_{\mathcal{G}}$) hash queries to $\mathcal{H}$ (resp. $\mathcal{G}$) oracle and q_s signature queries, is able to produce a signature forgery with probability ε within time τ. We let $q_h = q_{\mathcal{H}} + q_{\mathcal{G}}$.

Attacker $\mathcal{A}$ is run with the following simulation:

Initialization: $\mathcal{A}$ is initialized with public key $y = g^x$ and public parameters $(g, q, G_{g,q})$.

Answering new $\mathcal{G}(g, h, y, z, u, v)$ query: The simulator returns a random number in $\mathbb{Z}_q$.

Answering new $\mathcal{H}(m, r)$ query: The simulator generates a random number $d \in \mathbb{Z}_q$, and returns $(g^a)\, g^d$.

Answering signature query on $m \in \mathcal{M}$: The simulator generates a random number $r \in \{0,1\}^{\ell_r}$. If $\mathcal{H}(m, r)$ is already set, the simulator fails and stops (Event 1). Else, the simulator generates a random number $\kappa \in \mathbb{Z}_q$, sets $h = \mathcal{H}(m, r) = g^\kappa$ and computes $z = (g^x)^\kappa$ —remark that $DL_h(z) = DL_g(y)\ (= x)$. Then, the simulator randomly picks $(s, c) \in \mathbb{Z}_q \times \mathbb{Z}_q$ and computes $u = g^s\, y^{-c}$ and $v = h^s\, z^{-c}$. If $\mathcal{G}(g, h, y, z, u, v)$ is already set, the simulator fails and stops (Event 2). Else, the simulator sets $\mathcal{G}(g, h, y, z, u, v) = c$ and returns the valid signature (z, r, s, c).

As we can see, the simulation is valid and indistinguishable from an actual signer, except for some events:

- Event 1: As r is a random number in $\{0,1\}^{\ell_r}$, the probability that the $\mathcal{H}(m,r)$ is already set is less than $\frac{q_{\mathcal{H}}+q_s}{2^{\ell_r}}$, for one signature query. For q_s signature queries, the failure probability is thus upper bounded by $\frac{q_s \cdot (q_{\mathcal{H}}+q_s)}{2^{\ell_r}}$.
- Event 2: From the simulation, the input tuples to the $\mathcal{G}$ oracle are of the form $(g,h,y,z,u,v) = (g, g^{\kappa}, y, y^{\kappa}, g^{k}, g^{\kappa k})$ with $(k,\kappa) \in \mathbb{Z}_q \times \mathbb{Z}_q$. Furthermore, as $h = g^{\kappa} = \mathcal{H}(m,r)$ is not known by the attacker (else, Event 1 would have happened), κ is absolutely random for the attacker. Hence, the probability that $\mathcal{G}(g,h,y,z,u,v)$ is already set is less than $\frac{q_{\mathcal{G}}+q_s}{q^2}$. For q_s signature queries, the failure probability is thus upper bounded by $\frac{q_s \cdot (q_{\mathcal{G}}+q_s)}{q^2}$.

Solving the CDH challenge (g, g^x, g^a): Except when these two rare events occur, attacker $\mathcal{A}$ returns, with probability ε, a valid signature forgery $\sigma = (z,r,s,c)$ on a message m that was not submitted to the signature oracle, with $h = \mathcal{H}(m,r) = (g^a)\, g^d$ for some d known to the simulator. Provided that $DL_h(z) = DL_g(y) = x$, the solution to the CDH challenge is $z\,(g^x)^{-d}$.

Now, we calculate the probability that the attacker outputs a valid forgery but with $DL_h(z) \neq DL_g(y) = x$. Letting $u = g^k$, $v = h^{k'}$ and $z = h^{x'}$ ($\Leftrightarrow x' = DL_h(z) \neq x$), it follows, as the forgery is valid, that $k = s - cx \bmod q$ and $k' = s - cx' \bmod q$. Hence, we get $c = \mathcal{G}(g,h,y,h^{x'},g^k,h^{k'}) = \frac{k-k'}{x'-x} \bmod q$. As $\mathcal{G}(g,h,y,h^{x'},g^k,h^{k'})$ is not defined (else, Event 2 would have happened), it follows that the relation $c = \mathcal{G}(g,h,y,h^{x'},g^k,h^{k'}) = \frac{k-k'}{x'-x} \bmod q$ is never satisfied, except with probability $\frac{q_{\mathcal{G}}}{q}$.

Putting all together, we can conclude that the *EDL* signature scheme is tightly as secure as the Diffie-Hellman problem: the success probability ε' of our reduction satisfies

$$\varepsilon' \geq \varepsilon - q_s \left(\frac{q_s + q_h}{q^2} + \frac{q_s + q_h}{2^{\ell_r}} \right) - \frac{q_h}{q}$$

and the running time τ' satisfies

$$\tau' \lesssim \tau + (6q_s + q_h)\tau_0$$

where τ_0 is the time required for an exponentiation. □

3.3 Features of the *EDL* Signature

The *EDL* signature scheme is proven secure relatively to the computational Diffie-Hellman problem, with a tight reduction. Hence, its security is a strong point.

The scheme yields signatures of $(\ell_p + 2\ell_q + \ell_r)$ bits. This may appear somewhat long but actually it is not, given such a strong security.[5]

[5] In [GJ03], the authors estimate that if the discrete logarithm problem is supposed to be infeasible for 1000-bit primes, the forking lemma's technique tells that Schnorr signatures are secure in a field modulo a 8000-bit prime.

In its classical use, the scheme cannot be used with coupons, but, as noted by Goh and Jarecki, one can use the technique of [ST01] based on chameleon hash functions [KR00] to transform this signature into a signature with coupons, what we will call *EDL-CH* in the sequel. Producing a *EDL-CH* signature forgery is equivalent to produce a signature forgery in the regular *EDL* signature scheme, or to find a collision in the chameleon hash function. Hence, the natural way to get a signature with coupons and with a tight security reduction to the computational Diffie-Hellman problem is to use a chameleon hash function whose collision-resistance is also based on discrete logarithm or Diffie-Hellman problem (*e.g.,* $\mathcal{H}(m,r) = \mathcal{H}_0(g^m\, y^r)$, where $\mathcal{H}_0 : G_{g,q} \rightarrow G_{g,q}$ is a hash function). But the cost of this way to create coupons is a slower verification. Further, using the chameleon hash $\mathcal{H}(m,r) = \mathcal{H}_0(g^m\, y^r)$ implies that one needs to define random number $r \in \mathbb{Z}_q$ (and not in $\{0,1\}^{\ell_r}$). This makes the *EDL-CH* signatures slightly longer: $(\ell_p + 3\ell_q)$ bits.

3.4 Katz-Wang Signature Scheme

In [KW03], Katz and Wang proposed two modifications of *EDL*, one which consists in a scheme with short signatures tightly based on the DDH assumption, and one another that uses signature shorter than *EDL* but keeps tightly related to the CDH problem. In this section, we briefly remind the second scheme.

The idea of Katz and Wang is to remove the *randomness* of r, and to replace it by *unpredictability.* Namely, r is replaced by a bit b that can only be computed by the signer (*e.g.,* b is the result of a PRF, under a secret key included in the signing key):[6] the signatures are then (z,s,c,b), and so are shorter than *EDL* signatures by 110 bits. The proof of *EDL* is then slightly modified for Katz-Wang scheme. For $\mathcal{H}(m,b)$ queries, the simulator computes the bit value corresponding to m, then:

- if this value is b, the returned value is of the form g^κ, which allows to compute corresponding z very simply: $z = (g^x)^\kappa$;
- if this value is not b, the returned value is of the form $(g^a)\, g^d$.

Consequently, it is simple for the simulator to reply to signature queries, as it knows the right value b for each message m. On the contrary, as b cannot be guessed by the forger better than randomly for any new message m, its forge will be with the wrong b with a probability $\frac{1}{2}$, and with this probability, the CDH problem will be solved by the simulator.

Hence, this modification gives a signature scheme with a signature length of $(\ell_p + 2\ell_q + 1)$ bits, and which is just one bit less secure than *EDL* when taking same parameters. Unfortunately, in this scheme, only u can be computed off-line, and so the on-line part of the signature is two modular exponentiations in $G_{g,q}$.

[6] In other words, in *EDL*, signing few times the same message would result in different random numbers r, while doing the same with Katz-Wang scheme would give always the same bit b.

4 Our Signature Scheme

Looking at the description of *EDL*, we can see that basically two random values are used: k is used to generate a proof of knowledge of the discrete logarithm while r is used to ensure that the attacker cannot predict the value of h, that will be used during simulations.

More precisely, in *EDL*, h is taken equal to $\mathcal{H}(m, r)$, with a sufficiently large random number r. As RSA-PSS does in a certain sense, the goal is to avoid, with overwhelming probability, that the attacker requests the value of $\mathcal{H}(m, r)$ with a random number r that will afterwards appear during signature queries on m. Indeed, we want to build the $\mathcal{H}(m, r)$'s involved in signature simulations in a certain form and the $\mathcal{H}(m, r)$'s returned to direct queries (and susceptible to be used in the final forgery) in another form (see Section 3.2 for more detail).

Our first idea is the following: *Why not trying to put the randomness of k inside $\mathcal{H}(m, \cdot)$ instead of using another random number r that increases the size of the signature?* Clearly, one cannot use $\mathcal{H}(m, k)$ directly, but $\mathcal{H}(m, u)$ looks promising (and appears to be secure, as proven in Appendix C). As a result, the size of the so-constructed signature is reduced.

Our second idea is the following: *Would it be possible to put m inside $\mathcal{G}(\cdot)$ rather than in $\mathcal{H}(\cdot)$, as done in [Sch91] or in [KW03]?* The goal here is to allow as many precomputations as possible. This trick does not apply to *EDL*, but when combined with the previously suggested technique, the answer appears to be positive.

Intuitively, using $z = \mathcal{H}(r)^x$ and putting m in $\mathcal{G}(\cdot)$ in *EDL* is insecure because an attacker could easily reuse a z returned by the signer, and so a simulator would not solve a CDH problem. On the contrary, in our construction, we will show that using $z = \mathcal{H}(u)^x$ remains secure, as an attacker could not reuse an $\mathcal{H}(u)^x$ returned by the signer, unless the discrete logarithm is revealed: indeed, u satisfies a certain relation ($u = g^s\, y^{-c}$) that cannot be given for two different c's for the same u without revealing the discrete logarithm.

In this section, we describe more formally our scheme and prove strictly the intuition that we have just given.

4.1 Description

Our scheme goes as follows:

Global set-up: Let ℓ_p and ℓ_q denote security parameters. Let also a cyclic group $G_{g,q}$ of order q, generated by g, where q is a ℓ_q-bit prime and the representation of the elements of $G_{g,q}$ is included in $\{0,1\}^{\ell_p}$. Finally, let two hash functions, $\mathcal{H} : G_{g,q} \to G_{g,q}$ and $\mathcal{G} : \mathcal{M} \times (G_{g,q})^6 \to \mathbb{Z}_q$.

Key generation: The private key is a random number $x \in \mathbb{Z}_q$. The corresponding public key is $y = g^x$.

Signature: To sign a message $m \in \mathcal{M}$, one first randomly chooses $k \in \mathbb{Z}_q$, and computes $u = g^k$, $h = \mathcal{H}(u)$, $z = h^x$ and $v = h^k$. Next, one computes $c = \mathcal{G}(m, g, h, y, z, u, v)$ and $s = k + cx \bmod q$. The signature on m is $\sigma = (z, s, c)$.

Verification: To verify a signature $\sigma = (z, s, c) \in G_{g,q} \times (\mathbb{Z}_q)^2$ on a message $m \in \mathcal{M}$, one computes $u' = g^s\, y^{-c}$, $h' = \mathcal{H}(u')$, and $v' = h'^s\, z^{-c}$. The signature σ is accepted iff $c = \mathcal{G}(m, g, h', y, z, u', v')$.

As an advantage, our signatures are smaller than the *EDL*'s ones: they are only $(\ell_p + 2\ell_q)$-bit long. We still have to prove that the scheme is tightly related to the computational Diffie-Hellman problem, which is done in the next section — but assuming this for the moment, we can see that, using the numerical values of [GJ03], our scheme leads to a gain of $\ell_r = 111$ bits per signature.

4.2 Security of the Proposed Scheme

In this section, we reduce the security of the proposed scheme to the security of the computational Diffie-Hellman problem. The proof consists in showing that the proposed scheme is a proof that $DL_h(z) = DL_g(y) = x$.

Theorem 2. *Let $\mathcal{A}$ be an adversary which can produce, with success probability ε, an existential forgery under a chosen-message attack within time τ, after q_h queries to the hash oracles and q_s queries to the signing oracle, in the random oracle model. Then the computational Diffie-Hellman problem can be solved with success probability ε' within time τ', with*

$$\varepsilon' \geq \varepsilon - 2q_s \left(\frac{q_s + q_h}{q} \right)$$

and

$$\tau' \lesssim \tau + (6q_s + q_h)\tau_0$$

where τ_0 is the time for an exponentiation in $G_{g,q}$.

Proof. We are given a group $G_{g,q}$ and a CDH challenge (g, g^x, g^a). We will use an attacker $\mathcal{A}$ against our signature scheme to solve this challenge, *i.e.*, to find g^{ax}. Our attacker $\mathcal{A}$, after $q_{\mathcal{H}}$ (resp. $q_{\mathcal{G}}$) hash queries to $\mathcal{H}$ (resp. $\mathcal{G}$) oracle and q_s signature queries, is able to produce a signature forgery with probability ε within time τ. We let $q_h = q_{\mathcal{H}} + q_{\mathcal{G}}$.

Attacker $\mathcal{A}$ is run with the following simulation:

Initialization: $\mathcal{A}$ is initialized with public key $y = g^x$ and public parameters $(g, q, G_{g,q})$.

Answering new $\mathcal{G}(m, g, h, y, z, u, v)$ query: The simulator returns a random number in $\mathbb{Z}_q$.

Answering new $\mathcal{H}(u)$ query: The simulator generates a random number $d \in \mathbb{Z}_q$, and returns $(g^a)\, g^d$. All queries u are stored in a list called U-List.

Answering signatures query on $m \in \mathcal{M}$: The simulator randomly generates $(\kappa, s, c) \in (\mathbb{Z}_q)^3$. Then, it computes $u = g^s\, y^{-c}$. If $\mathcal{H}(u)$ is already set, the simulator stops (Event 1). Else, the simulator sets $h = \mathcal{H}(u) = g^{\kappa}$ and computes $z = (g^x)^{\kappa}$ — remark that $DL_h(z) = DL_g(y)(= x)$. Finally, the simulator computes $v = h^s\, z^{-c}$. If $\mathcal{G}(m, g, h, y, z, u, v)$ is already set, the simulator stops and fails (Event 2). Else, the simulator sets $\mathcal{G}(m, g, h, y, z, u, v) = c$, and returns the valid signature (z, s, c). All u's computed during signature queries are stored in a list called Υ-List

As we can see, this simulator is valid and indistinguishable from an actual signer, except for some events:

- Event 1: As u is a random number in $G_{g,q}$, the probability that the $\mathcal{H}(u)$ is already set is less than $\frac{q_s+q_{\mathcal{H}}}{q}$, for one signature query. For q_s signature queries, the failure probability is thus upper bounded by $\frac{q_s\cdot(q_s+q_{\mathcal{H}})}{q}$.
- Event 2: From the simulation, the input tuples to the $\mathcal{G}$ oracle are of the form $(m, g, h, y, z, u, v) = (m, g, g^{\kappa}, y, y^{\kappa}, g^k, g^{\kappa k})$ for $k \in \mathbb{Z}_q$ and κ which is determined by the relation $h = \mathcal{H}(g^k) = g^{\kappa}$; but as Event 1 did not happened, h is absolutely unknown for the attacker, and so κ is a random integer of $\mathbb{Z}_q$. Then, the probability that $\mathcal{G}(m, g, h, y, z, u, v)$ is already set is less than $\frac{q_s+q_{\mathcal{G}}}{q^2}$. For q_s signature queries, the failure probability is thus upper bounded by $\frac{q_s\cdot(q_s+q_{\mathcal{G}})}{q^2} \leq \frac{q_s\cdot(q_s+q_{\mathcal{G}})}{q}$.

As a conclusion, except with a probability smaller than $\delta_{sim} = q_s\big(\frac{q_h+2q_s}{q}\big)$, the simulation is successful.

In other words, with a probability $\varepsilon_{sim} \geq \varepsilon - \delta_{sim}$, the attacker $\mathcal{A}$ is able to return a valid signature forgery $(\hat{z}, \hat{s}, \hat{c})$ on a message $\hat{m} \in \mathcal{M}$ that was never submitted to the signature oracle. The simulator deduces from this forgery the corresponding tuple $(\hat{u}, \hat{v}, \hat{h})$, by the following computations: $\hat{u} = g^{\hat{s}}\, y^{-\hat{c}}$, $\hat{h} = \mathcal{H}(\hat{u})$, and $\hat{v} = \hat{h}^{\hat{s}}\, \hat{z}^{-\hat{c}}$. Notably, if $\mathcal{H}(\hat{u})$ has not been queried to the $\mathcal{H}$ oracle by the attacker or set by the signature oracle, the simulator queries it to the $\mathcal{H}$ oracle itself. Hence, $\hat{u}$ is a member of U-List or a member of Υ-List.

Solving the CDH challenge (g, g^x, g^a). At this step, once the forgery is returned by the attacker, there are two cases, contrary to the proof of *EDL*.

In the first case, $\hat{u}$ is a member of U-List. This is the case that corresponds to the only case of the proof of *EDL*. As in *EDL*, we write $\hat{u} = g^k$, $\hat{v} = \hat{h}^{k'}$ and $\hat{z} = \hat{h}^{x'}$, and we get, as the signature is valid, $k = \hat{s} - \hat{c}x \bmod q$ and $k' = \hat{s} - \hat{c}x' \bmod q$. Then, if $x \neq x'$, we have $\hat{c} = \mathcal{G}(\hat{m}, g, \hat{h}, y, \hat{h}^{x'}, g^k, \hat{h}^{k'}) = \frac{k-k'}{x'-x} \bmod q$. As the message $\hat{m}$ is new, $\mathcal{G}(\hat{m}, g, \hat{h}, y, \hat{h}^{x'}, g^k, \hat{h}^{k'})$ was not set during a signature query, and so we know that $DL_{\hat{h}}(\hat{z}) = DL_g(y)(= x)$, except with a probability $\frac{q_{\mathcal{G}}}{q}$. Apart this error, the simulator receives from the attacker a signature with $\hat{z} = \hat{h}^x$, and it knows d such that $\hat{h} = \mathcal{H}(\hat{u}) = (g^a)\, g^d$. Then the simulator can return the solution to the CDH challenge, which is $\hat{z}\,(g^x)^{-d}$. In this first case, the forgery is successfully used to solve the CDH challenge, except with a probability smaller than $\delta_1 = \frac{q_h}{q}$.

In the second case, $\hat{u}$ is not a member of $\mathsf{U\text{-}List}$, and so is a member of $\Upsilon\text{-}\mathsf{List}$. This case can happen, contrary to the *EDL* signature scheme, as there is no message in the input of $\mathcal{H}$, and so we can imagine that the attacker reuse a u that corresponds to a u of a signature given by the signature oracle. Then, the simulator can recover from its log files all quantities that correspond to this $u = \hat{u}$, *i.e.*, h, v, z, s, c and m.

At this moment, we can see that we have $u = g^s y^{-c} = \hat{u} = g^{\hat{s}} y^{-\hat{c}}$. It is exactly the kind of hypothesis that is used by the forking lemma to prove a (loose) security. But here, this equality is not obtained by restarting the attacker (as it is done in the forking lemma), but just by construction. More precisely, we can recover easily the private key x, as far as $\hat{c} \neq c \bmod q$.

As the message $\hat{m}$ is new, $c \neq \hat{c}$ or a collision on $\mathcal{G}$ function happened, between a $\mathcal{G}$ returned the signature simulation and a $\mathcal{G}$ returned by a direct $\mathcal{G}$ query, which occurs with a probability smaller than $\frac{q_s \cdot q_{\mathcal{G}}}{q}$. Hence, except an error with a probability smaller than $\delta_2 = \frac{q_s \cdot q_{\mathcal{G}}}{q}$, we have $\hat{c} \neq c$, and so we can recover the private key x: equation $s - xc = \hat{s} - x\hat{c} \bmod q$ gives $x = \frac{s - \hat{s}}{c - \hat{c}} \bmod q$. We can see that this second case gives not only the solution to the CDH challenge, but also the solution to the discrete logarithm.

As a conclusion, we can see that in both cases, our simulator can transform the forgery given by the attacker into the solution to the CDH challenge.

Putting all together, the success probability ε' of our reduction satisfies $\varepsilon' \geq \varepsilon - \delta_{sim} - \max(\delta_1, \delta_2)$, which gives, using $q_{\mathcal{H}} + q_{\mathcal{G}} = q_h$,

$$\varepsilon' \geq \varepsilon - 2q_s \left(\frac{q_s + q_h}{q} \right)$$

and the running time τ' satisfies

$$\tau' \lesssim \tau + (6q_s + q_h)\tau_0 \ .$$

As we can see, our scheme is tight, as far as $\frac{q_s \cdot q_h}{q} \leq \frac{\varepsilon}{4}$. □

4.3 Our Proposed Scheme with Coupons

Interestingly, our scheme allows what we call a *cost-free* use of coupons. By this, we mean that the signer is free to choose to use coupons or not: this choice of the signer does not affect the verifier as the verification step remains unchanged.

This is done in a very natural way: the signature step (cf. Section 4.1) is simply split into two steps.

Off-line signature: To create a new coupon, one randomly chooses $k \in \mathbb{Z}_q$ and computes $u = g^k$, $h = \mathcal{H}(u)$, $z = h^x$ and $v = h^k$. The coupon is the tuple (u, v, h, z, k).

On-line signature: To sign a message $m \in \mathcal{M}$, one uses a fresh coupon (u, v, h, z, k) and just computes $c = \mathcal{G}(m, g, h, y, z, u, v)$ and $s = k + cx \bmod q$. The signature on m is $\sigma = (z, s, c)$.

The verification step remains the same. This property is very useful as it allows the signer to precompute coupons and to sign on-line very quickly, namely, by just performing one hash function evaluation followed by one modular multiplication.[7]

As previously described, our scheme features a coupon size of $(4\ell_p + \ell_q)$ bits. This size can be reduced to $(3\ell_p + \ell_q)$ bits by not storing the value of h, *i.e.,* a coupon is defined as (u, v, z, k). Then, $h = \mathcal{H}(u)$ is evaluated in the on-line step. This option turns out useful for memory constrained devices like smart cards.

An even more sophisticated solution that minimizes the size of the coupon is described in Appendix A.

4.4 Size of Parameters

In this section, we show how to set the values of ℓ_q and ℓ_p to attain a security level of 2^κ. Our analysis basically follows Goh and Jarecki's for *EDL*. Assuming we take the best $(q_h, q_s, \tau, \varepsilon)$-attacker against our scheme, he can find a forgery in an average time of $\frac{\tau}{\varepsilon}$. Letting $\tau = 2^n$ and $\varepsilon = 2^{-e}$, we get $\log_2(\frac{\tau}{\varepsilon}) = n + e = \kappa$, by definition of the security level of our scheme.

Furthermore, we can use this attacker, as shown in the proof of Section 4.2, to solve the CDH problem in a time of $\frac{\tau'}{\varepsilon'}$. We let $2^{\kappa'}$ denote the security level of the CDH in the subgroup $G_{g,q}$. By definition, we have $\kappa' \leq \log_2(\frac{\tau'}{\varepsilon'})$. Because of the $O(\sqrt{q})$ security for the discrete logarithm in $G_{g,q}$, we have $\ell_q \geq 2\kappa'$.

We use the cost of the evaluation of a hash function as the unit of time. Hence, $q_h \leq 2^n$. We suppose that τ_0 (the time for an exponentiation in $G_{g,q}$) is 100 times the time of a hash function evaluation. So, using $q_s \leq q_h$, we obtain that $\tau' \simeq 2^{n+7}$ and $\varepsilon' \gtrsim \varepsilon - \frac{4q_s \cdot q_h}{q}$. As long as $q_s \leq 2^{\ell_q - e - 3 - n} = 2^{\ell_q - \kappa - 3}$ (*e.g.,* $\kappa = 80$, $q_s \leq 2^{80}$, $q_h \leq 2^{80}$ and $\ell_q \geq 176$), we have $\varepsilon' \gtrsim 2^{-e-1}$. Then, $\log_2(\frac{\tau'}{\varepsilon'}) \lesssim n + 7 + e + 1 = \kappa + 8$. We finally obtain $\kappa \geq \kappa' - 8$.

For example, if the targeted security level is $\kappa = 80$, it is sufficient to use $\kappa' = 88$ (and hence $\ell_q \geq 176$). It proves that our scheme is very efficient in terms of signature size, as we can use the same subgroup $G_{g,q}$ as the one used by Goh and Jarecki for *EDL* and have the same security. One can remark that our scheme remains secure even if we limit q_s to 2^{80}, while in *EDL*, q_s was limited to 2^{30}, or the random number r was made appropriately longer.

4.5 Detailed Comparison with *EDL*, the Katz-Wang Scheme and Other Schemes

In this paragraph, we sum up the advantages of our scheme. Compared to *EDL*, our scheme features

[7] This is comparable to the fastest off-line/on-line signature schemes of Schnorr, Girault-Poupard-Stern or Poupard-Stern [Sch91,Gir91,PS98,PS99]. One would remark that Girault-Poupard-Stern scheme does not require a reduction modulo the group order, but yields longer signatures: this elegant technique can also be used in our scheme, to get an even faster on-line signature scheme at the price of longer signatures.

1. faster signatures with a cost-free use of coupons: the on-line part only requires one hash function evaluation followed by one modular multiplication in $\mathbb{Z}_q$, while in *EDL*, this phase consists of two hash function evaluations and two modular exponentiations in $G_{g,q}$;
2. same verification step efficiency;
3. shorter signatures of $\ell_r \geq 111$ bits: in a subgroup of ${\mathbb{F}_p}^*$, taking $\ell_p = 1024$ and $\ell_q = 176$, this represents an improvement of 7%. In the elliptic curve setting, the gain is even more sensible, as z can be represented with a length around $\ell_q = 176$, resulting in an improvement of 17%.

Compared to the Katz-Wang scheme, our scheme features

1. faster signatures with a cost-free use of coupons: the on-line part only requires one hash function evaluation followed by one modular multiplication in $\mathbb{Z}_q$, while in Katz-Wang signature scheme, this phase consists of two hash function evaluations and two modular exponentiations in $G_{g,q}$;
2. same verification step efficiency;
3. less significantly, shorter signatures of 1 bit and a security parameter greater of 1 bit;
4. smaller key size, as computing b by a PRF or in another way require an additional key, that should better not be related to the private key x.

Furthermore, as noticed in [KW03], the computation of an hash $\mathcal{H} : G_{g,q} \rightarrow G_{g,q}$ can be very long, namely it costs an exponentiation of $(\ell_p - \ell_q)$ bits, which is much longer than the two exponentiations in $G_{g,q}$. In our scheme, this hash computation is done off-line, contrary to *EDL* and Katz-Wang schemes.

Compared to the off-line/on-line version of *EDL*, *EDL-CH*, the off-line/on-line version of our scheme presents

1. faster and unchanged verification step (remember that *EDL-CH* relies on chameleon hashes, which requires additional exponentiations);
2. shorter signatures, *i.e.*, $\ell_q \geq 176$ bits less than *EDL-CH*; again, in a subgroup of ${\mathbb{F}_p}^*$, taking $\ell_p = 1024$ and $\ell_q = 176$, this represents an improvement of 11% and of 25% in the elliptic curve setting.

Finally, owing to its security tightness, our scheme fulfills or even improves most of the advantages of *EDL* that were presented by Goh and Jarecki, by comparison with other discrete-logarithm schemes, such as Schnorr signature, with same security level.

On the one hand, using our scheme in $G_{g,q} \subseteq {\mathbb{F}_p}^*$, we can use a field 8 times smaller and a subgroup of order twice smaller than in other discrete-logarithm schemes (as in *EDL*). Notably, it means that public keys are smaller by a factor of 8, private keys are smaller by a factor of 2. In this case, our signatures are about twice as long as other discrete-logarithm schemes.

On the other hand, in the elliptic curve setting, our public and private keys are smaller by a factor of 2 and our signatures are 25% smaller than in previously known schemes.

This clearly shows the advantages of the proposed scheme.

5 Conclusion

At Eurocrypt '03, Goh and Jarecki gave a proof that the security of *EDL* is tightly related to the CDH problem, in the random oracle model. They also proposed to use the technique of Shamir and Tauman, based on chameleon hash functions, to get a version of *EDL* scheme with coupons: *EDL-CH*.

In this paper, we have proposed a new signature scheme which, similarly to *EDL*, features a *tight* security reduction relatively to the CDH problem but whose resulting signatures are smaller: if coupons are not used, we gain ℓ_r bits compared to *EDL* signatures; in the off-line/on-line version, we gain ℓ_q bits compared to *EDL-CH* signatures. Furthermore, contrary to *EDL*, no additional trick is needed to turn our signature scheme in an off-line/on-line version.

Our scheme represents to date the most efficient scheme of any signature scheme with a tight security reduction in the discrete-log setting.

Acknowledgements

The author would like to thank his careful PhD advisor, David Pointcheval, for teaching him so much about provable security. Many thanks also go to Marc Joye for his attention and fruitful support in our research. The author thanks Jean-François Dhem, Philippe Proust and David Naccache, as well as Dan Boneh and Jonathan Katz for their comments. Finally, anonymous referees of Crypto '05 are also thanked for their precious remarks, and notably for corrections on our previous proofs.

References

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.

[BR96] M. Bellare and P. Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In U. Maurer, editor, *Advances in Cryptology – EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer-Verlag, 1996.

[BLS04] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. *Journal of Cryptology*, 17(4):297–319, 2004.

[BSS99] I. Blake, G. Seroussi, and N.P. Smart. Elliptic Curves in Cryptography. Cambridge University Press, 1999.

[Che05] B. Chevallier-Mames. An Efficient CDH-based Signature Scheme With a Tight Security Reduction. In V. Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, to appear in *Lecture Notes in Computer Science*, Springer-Verlag, 2005.

[Cor02] J.-S. Coron. Optimal security proofs for PSS and other signature schemes. In L.R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 272–287. Springer-Verlag, 2002.

[CP92] D. Chaum and T.P. Pedersen. Wallet databases with observers. In E. Brickell, editor, *Advances in Cryptology – CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer-Verlag, 1992.

[CS00] R. Cramer and V. Shoup. Signature scheme based on the strong RSA assumption. *ACM Transactions on Information and System Security*, 3(3):161–185, 2000.

[DH76] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[ElG85] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, 1985.

[FS87] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A.M. Odlyzko, editor, *Advances in Cryptology – CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer-Verlag, 1987.

[GHR99] R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In M. Bellare, editor, *Advances in Cryptology – EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 123–139. Springer-Verlag, 1999.

[Gir91] M. Girault. An identity-based identification scheme based on discrete logarithms modulo a composite number. In I.B. Damgård, editor, *Advances in Cryptology – EUROCRYPT '90*, volume 473 of *Lecture Notes in Computer Science*, pages 481–486. Springer-Verlag, 1991.

[GJ03] E.-J. Goh and S. Jarecki. A signature scheme as secure as the Diffie-Hellman problem. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, Lecture Notes in Computer Science, pages 401–415. Springer-Verlag, 2003.

[GMR84] S. Goldwasser, S. Micali, and R. Rivest. A "paradoxical" solution to the signature problem. In *Proceedings of the 25th FOCS*, pages 441–448. IEEE, 1984.

[GMR88] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen message attacks. *SIAM Journal of Computing*, 17(2):281–308, 1988.

[JS99] M. Jakobsson and C.P. Schnorr. Efficient oblivious proofs of correct exponentiation. In B. Preneel, editor, *Communications and Multimedia Security – CMS '99*, volume 152 of *IFIP Conference Proceedings*, pages 71–86. Kluver, 1999.

[Kob87] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, vol. 48, pp. 203-209, 1987.

[KM04] N. Koblitz and A. Menezes. Another look at "provable security". Cryptology ePrint Archive, Report 2004/152, 2004. http://eprint.iacr.org/.

[KR00] H. Krawczyk and T. Rabin. Chameleon signatures. In *Symposium on Network and Distributed System Security – NDSS 2000*, pages 143–154. Internet Society, 2000.

[KW03] J. Katz and N. Wang. Efficiency improvements for signature schemes with tight security reductions. In *ACM Conference on Computer and Communications Security*, pages 155–164. ACM Press, 2003.

[MR02] S. Micali and L. Reyzin. Improving the exact security of digital signatre schemes. *Journal of Cryptology*, 15(1):1–18, 2002.

[Mil85] V. Miller. Use of elliptic curves in cryptography. In H. C. Williams, editor, *Advances in Cryptology – CRYPTO '85*, Lecture Notes in Computer Science, pages 417–426. Springer-Verlag, 1986.

[P1363] IEEE P1363. IEEE Standard Specifications for Public-Key Cryptography. IEEE Computer Society, August 2000.

[PS96] D. Pointcheval and J. Stern. Security proofs for signature schemes. In U. Maurer, editor, *Advances in Cryptology – EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. Springer-Verlag, 1996.

[PS98] G. Poupard and J. Stern. Security analysis of a practical "on the fly" authentication and signature generation. In K. Nyberg, editor, *Advances in Cryptology – EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 422–436. Springer-Verlag, 1998.

[PS99] G. Poupard and J. Stern. On the fly signatures based on factoring. In *ACM Conference on Computer and Communications Security*, pages 37–45. ACM Press, 1999.

[Rab79] M.O. Rabin. Digital signatures and public-key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, MIT Laboratory for Computer Science, January 1979.

[RSA78] R.L. Rivest, A. Shamir, and L.M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[Sch91] C.-P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[SPM+02] J. Stern, D. Pointcheval, J. Malone-Lee, and N. Smart. Flaws in applying proof methodologies to signature schemes. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 93–110. Springer-Verlag, 2002.

[ST01] A. Shamir and Y. Tauman. Improved online/offline signature schemes. In J. Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, Lecture Notes in Computer Science, pages 355–367. Springer-Verlag, 2001.

A An Efficient CDH-based Signature Scheme with Smaller Coupons

In this appendix, we propose some modifications to the proposed scheme (Section 4) in order to obtain smaller coupons with the same security.

A.1 An Efficient Coupon-based Variant

We modify our scheme in the following way:

Global set-up: Let ℓ_p, ℓ_q and ℓ_t denote security parameters. Let also a cyclic group $G_{g,q}$ of order q, generated by g, where q is a ℓ_q-bit prime and the representation of the elements of $G_{g,q}$ is included in $\{0,1\}^{\ell_p}$. Finally, let three hash functions, $\mathcal{H} : G_{g,q} \to G_{g,q}$, $\mathcal{G} : \mathcal{M} \times \{0,1\}^{\ell_t} \to \mathbb{Z}_q$ and $\mathcal{I} : (G_{g,q})^6 \to \{0,1\}^{\ell_t}$.

Key generation: The private key is a random number $x \in \mathbb{Z}_q$. The corresponding public key is $y = g^x$.

Off-line signature: To create a new coupon, one randomly chooses $k \in \mathbb{Z}_q$ and computes $u = g^k$, $h = \mathcal{H}(u)$, $z = h^x$ and $v = h^k$. Finally, one computes $t = \mathcal{I}(g, h, y, z, u, v)$. The coupon is the tuple (k, z, t).

On-line signature: To sign a message $m \in \mathcal{M}$, one uses a fresh coupon (k, z, t) and just computes $c = \mathcal{G}(m, t)$ and $s = k + cx \bmod q$. The signature on m is $\sigma = (z, s, c)$.

Verification: To verify a signature $\sigma = (z, s, c) \in G_{g,q} \times (\mathbb{Z}_q)^2$ on a message $m \in \mathcal{M}$, one computes $u' = g^s y^{-c}$, $h' = \mathcal{H}(u')$, $v' = h'^s z^{-c}$, and $t' = \mathcal{I}(g, h', y, z, u', v')$. The signature σ is accepted iff $c = \mathcal{G}(m, t')$.

This version of our scheme allows small coupons (*i.e.*, $\ell_p + \ell_q + \ell_t$ bits instead of $3\ell_p + \ell_q$), which allows, even in a constrained device like a smart card, to precompute and store a large number of coupons beforehand.

Remarkably, this version keeps the efficiency in the on-line phase. Moreover, this coupon technique has no cost for the verifier: contrary to *EDL-CH*, the verifier needs not to compute any chameleon hashes. Last but not least, the resulting signatures are still smaller than the *EDL* or *EDL-CH*'s ones: only $(\ell_p + 2\ell_q)$-bit long.

We show that our variant is still tightly related to the computational Diffie-Hellman problem in the next section.

A.2 Security of this Variant of our Scheme

About the security of our variant with small coupons, the following theorem stands:

Theorem 3. *Let $\mathcal{A}$ be an adversary which can produce, with success probability ε, an existential forgery under a chosen-message attack within time τ, after q_h queries to the hash oracles and q_s queries to the signing oracle, in the random oracle model. Then the computational Diffie-Hellman problem can be solved with success probability ε' within time τ', with*

$$\varepsilon' \geq \varepsilon - q_s\left(\frac{2q_s + 2q_h}{q} + \frac{q_s + q_h}{2^{\ell_t}}\right)$$

and

$$\tau' \lesssim \tau + (6q_s + q_h)\tau_0$$

where τ_0 is the time for an exponentiation in $G_{g,q}$.

Proof. We are given a group $G_{g,q}$ and a CDH challenge (g, g^x, g^a). We will use an attacker $\mathcal{A}$ against this variant of our signature scheme to solve this challenge, *i.e.,* to find g^{ax}. Our attacker $\mathcal{A}$, after $q_{\mathcal{H}}$ (resp. $q_{\mathcal{G}}, q_{\mathcal{I}}$) hash queries to the $\mathcal{H}$ (resp. $\mathcal{G}, \mathcal{I}$) oracle and q_s signature queries, is able to produce a signature forgery with probability ε within time τ. We let $q_h = q_{\mathcal{H}} + q_{\mathcal{G}} + q_{\mathcal{I}}$.

Attacker $\mathcal{A}$ is run with the following simulation:

Initialization: $\mathcal{A}$ is initialized with public key $y = g^x$ and public parameters $(g, q, G_{g,q})$.

Answering new $\mathcal{G}(m, t)$ query: The simulator returns a random number in $\mathbb{Z}_q$.

Answering new $\mathcal{H}(u)$ query: The simulator generates a random number $d \in \mathbb{Z}_q$, and returns $(g^a)\, g^d$. All queries u are stored in a list called U-List.

Answering new $\mathcal{I}(g, h, y, z, u, v)$ query: The simulator returns a random number of ℓ_t bits.

Answering signatures query on $m \in \mathcal{M}$: The simulator randomly generates $(\kappa, s, c) \in (\mathbb{Z}_q)^3$. Then, it computes $u = g^s\, y^{-c}$. If $\mathcal{H}(u)$ is already set, the simulator stops (Event 1). Else, the simulator sets $h = \mathcal{H}(u) = g^\kappa$ and computes $z = (g^a)^\kappa$ —remark that $DL_h(z) = DL_g(y)(= x)$. Finally, the simulator computes $v = h^s\, z^{-c}$. If $\mathcal{I}(g, h, y, z, u, v)$ is already set, the simulator stops (Event 2). Else, the simulator takes a random number t of ℓ_t bits, and sets $\mathcal{I}(g, h, y, z, u, v) = t$. If $\mathcal{G}(m, t)$ is already set, the simulator stops and fails (Event 3). Else, the simulator sets $\mathcal{G}(m, t) = c$, and returns the valid signature (z, s, c). All u's computed during signature queries are stored in a list called Υ-List

As we can see, this simulator is valid and indistinguishable from an actual signer, except for some events:

- Event 1: As u is a random number in $G_{g,q}$, the probability that the $\mathcal{H}(u)$ is already set is less than $\frac{q_{\mathcal{H}}+q_s}{q}$, for one signature query. For q_s signature queries, the failure probability is thus upper bounded by $\frac{q_s \cdot (q_s + q_{\mathcal{H}})}{q}$.

- Event 2: From the simulation, the input tuples to the $\mathcal{I}$ oracle are of the form $(g, h, y, z, u, v) = (g, g^\kappa, y, y^\kappa, g^k, g^{\kappa k})$ for $k \in \mathbb{Z}_q$ and κ which is determined by relation $\mathcal{H}(g^k) = g^\kappa$. Then, the probability that $\mathcal{I}(g, h, y, z, u, v)$ is already set is less than $\frac{q_s+q_\mathcal{I}}{q}$. For q_s signature queries, the failure probability is thus upper bounded by $\frac{q_s \cdot (q_s+q_\mathcal{I})}{q}$.
- Event 3: As t is a random number, the probability that $\mathcal{G}(m, t)$ is already set is less than $\frac{q_s+q_\mathcal{G}}{2^{\ell_t}}$. For q_s signature queries, the failure probability is thus upper bounded by $\frac{q_s \cdot (q_s+q_\mathcal{G})}{2^{\ell_t}}$.

As a conclusion, except with a probability of $\delta_{sim} \le q_s\big(\frac{q_\mathcal{H}+q_\mathcal{I}+2q_s}{q} + \frac{q_s+q_\mathcal{G}}{2^{\ell_t}}\big)$, the simulation is successful.

In other words, with a probability $\varepsilon_{sim} \ge \varepsilon - \delta_{sim}$, the attacker $\mathcal{A}$ is able to return a valid signature $(\hat{z}, \hat{s}, \hat{c})$ on a message $\hat{m} \in \mathcal{M}$ that was never submitted to the signature oracle. The simulator deduces from this forgery the corresponding tuple $(\hat{u}, \hat{v}, \hat{h}, \hat{t})$, by the following computations: $\hat{u} = g^{\hat{s}}\, y^{-\hat{c}}$, $\hat{h} = \mathcal{H}(\hat{u})$, $\hat{v} = \hat{h}^{\hat{s}}\, \hat{z}^{-\hat{c}}$ and $\hat{t} = \mathcal{I}(g, \hat{h}, y, \hat{z}, \hat{u}, \hat{v})$. Notably, if $\mathcal{H}(\hat{u})$ has not been queried to the $\mathcal{H}$ oracle by the attacker or set by the signature oracle, the simulator queries it to the $\mathcal{H}$ oracle itself. Hence, $\hat{u}$ is a member of U-List or a member of Υ-List.

Solving the CDH challenge (g, g^x, g^a). At this step, once the forgery is returned by the attacker, there are two cases, contrary to the proof of *EDL*.

In the first case, $\hat{u}$ is member of U-List. This is the case that corresponds to the only case of the proof of the *EDL* scheme. As in *EDL*, we write $\hat{u} = g^k$, $\hat{v} = \hat{h}^{k'}$ and $\hat{z} = \hat{h}^{x'}$, and we get, as the signature is valid, $k = \hat{s} - \hat{c}x \bmod q$ and $k' = \hat{s} - \hat{c}x' \bmod q$. Then, if $x \neq x'$, we have $\hat{t} = \mathcal{I}(g, \hat{h}, y, \hat{h}^{x'}, g^k, \hat{h}^{k'})$, and $\hat{c} = \mathcal{G}(\hat{m}, \hat{t}) = \frac{k-k'}{x'-x} \bmod q$. As the forgery is a forgery on a *new* message, which means that $\mathcal{G}(\hat{m}, \hat{t})$ was not set during a signature query, this shows that $DL_{\hat{h}}(\hat{z}) = DL_g(y)(= x)$, except with a probability $\frac{q_\mathcal{G}}{q}$.

Apart this error, the simulator receives from the attacker a signature with $\hat{z} = \hat{h}^x$, and it knows d such that $\hat{h} = \mathcal{H}(\hat{u}) = (g^a)\, g^d$. Then, the simulator can return the solution to the CDH challenge, which is $\hat{z}\, (g^x)^{-d}$. In this first case, the forgery is successfully used to solve the CDH challenge, except with a probability smaller than $\delta_1 = \frac{q_h}{q}$.

In the second case, $\hat{u}$ is not a member of U-List, and so is a member of Υ-List. This case can happen, contrary to the *EDL* signature scheme, as there is no message in the input of $\mathcal{H}$, and so we can imagine that the attacker reuse a u that corresponds to a u of a signature given by the signature oracle. Then, the simulator can recover from its log files all quantities that correspond to this $u = \hat{u}$, and notably s, t, c and m.

At this moment, we can see that we have $u = g^s\, y^{-c} = \hat{u} = g^{\hat{s}}\, y^{-\hat{c}}$. It is exactly the kind of hypothesis that is used by the forking lemma to prove a (loose) security. But here, this equality is not obtained by restarting the attacker (as it is done in the forking lemma), but just by construction. More precisely, we can recover easily the private key x, as far as $\hat{c} \neq c$.

As $m \neq \hat{m}$ (the forgery is a forgery on a *new* message), $c \neq \hat{c}$, or a collision collision on $\mathcal{G}$ function happened, between a $\mathcal{G}$ returned the signature simulation and a $\mathcal{G}$ returned by a direct $\mathcal{G}$ query, which occurs with a probability smaller than $\frac{q_s \cdot q_\mathcal{G}}{q}$. Hence, except an error with a probability smaller than $\delta_2 = \frac{q_s \cdot q_\mathcal{G}}{q}$, we have $\hat{c} \neq c$, and so we can recover the private key x: equation $s - xc = \hat{s} - x\hat{c} \bmod q$ gives $x = \frac{s-\hat{s}}{c-\hat{c}} \bmod q$. One can see that this second case gives not only the solution to CDH challenge, but also the solution to the discrete logarithm.

As a conclusion, we can see that in both cases, our simulator can transform a forgery given by the attacker into the solution to the CDH challenge.

Putting all together, the success probability ε' of our reduction satisfies $\varepsilon' \geq \varepsilon - \delta_{sim} - \max(\delta_1, \delta_2)$, which gives, using $q_\mathcal{H} + q_\mathcal{G} + q_\mathcal{I} = q_h$,

$$\varepsilon' \geq \varepsilon - q_s \left(\frac{2q_s + q_h}{q} + \frac{q_s + q_h}{2^{\ell_t}} \right) - \frac{q_s \cdot q_h}{q}$$

i.e., supposing that $\ell_t \ll \ell_q$,

$$\varepsilon' \gtrsim \varepsilon - \frac{q_s \cdot q_h}{2^{\ell_t}} - \frac{2q_s \cdot q_h}{q}$$

Furthermore, the running time τ' of this simulation is such that

$$\tau' \lesssim \tau + (6q_s + q_h)\tau_0.$$

As we can see, our scheme is tight, as far as $\frac{2q_s \cdot q_h}{2^{\ell_t}} + \frac{q_s \cdot q_h}{q} \leq \frac{\varepsilon}{2}$ □

B First Step of Our Idea: Smaller Signatures Tightly Based on CDH

In a pedagogical purpose, we propose hereafter the first improvement that we thought about, in order to reduce the size of *EDL*'s signature. Anyway, we remind that there is no objective reason to prefer this version to our scheme that we described in Section 4.1.

B.1 Our Construction

The resulting scheme proceeds as follows (the global set-up and key generation are unchanged; cf. Section 3.1):

Signature: To sign a message $m \in \mathcal{M}$, one first randomly chooses $k \in \mathbb{Z}_q$, and computes $u = g^k$, $h = \mathcal{H}(m, u)$, $z = h^x$, $v = h^k$, $c = \mathcal{G}(g, h, y, z, u, v)$ and $s = k + cx \bmod q$. The signature on m is $\sigma = (z, s, c)$.

Verification: To verify a signature $\sigma = (z, s, c) \in G_{g,q} \times (\mathbb{Z}_q)^2$ on a message $m \in \mathcal{M}$, one computes $u' = g^s\, y^{-c}$, $h' = \mathcal{H}(m, u')$ and $v' = h'^s\, z^{-c}$. The signature is accepted iff $c = \mathcal{G}(g, h', y, z, u', v')$.

This modification to *EDL* gives a better bandwidth (signatures are ℓ_r bits smaller than regular *EDL* signatures). The security reduction is similar to the one of Section 3.2 and is given in the following.

B.2 Security of This Construction

About the security of this scheme, the following theorem stands:

Theorem 4. *Let $\mathcal{A}$ be an adversary which can produce, with success probability ε, an existential forgery under a chosen-message attack within time τ, after q_h queries to the hash oracles and q_s queries to the signing oracle, in the random oracle model. Then the computational Diffie-Hellman problem can be solved with success probability ε' within time τ', with*

$$\varepsilon' \geq \varepsilon - q_s \left(\frac{2q_s + q_h}{q} \right) - \frac{q_h}{q}$$

and

$$\tau' \lesssim \tau + (6q_s + q_h)\tau_0$$

where τ_0 the time for an exponentiation in $G_{g,q}$.

Proof. We are given a group $G_{g,q}$ and a CDH challenge (g, g^x, g^a). We will use an attacker $\mathcal{A}$ against our variant of the *EDL* signature scheme to solve this challenge, *i.e.,* to find g^{ax}. Our attacker $\mathcal{A}$, after $q_{\mathcal{H}}$ (resp. $q_{\mathcal{G}}$) hash queries to $\mathcal{H}$ (resp. $\mathcal{G}$) oracle and q_s signature queries, is able to produce a signature forgery with probability ε within time τ. We let $q_h = q_{\mathcal{H}} + q_{\mathcal{G}}$.

Attacker $\mathcal{A}$ is run with the following simulation:

Initialization: $\mathcal{A}$ is initialized with public key $y = g^x$ and public parameters $(g, q, G_{g,q})$.

Answering new $\mathcal{G}(g, h, y, z, u, v)$ query: The simulator returns a random number in $\mathbb{Z}_q$.

Answering new $\mathcal{H}(m, u)$ query: The simulator generates a random number $d \in \mathbb{Z}_q$, and returns $(g^a)\, g^d$.

Answering signatures query of $m \in \mathcal{M}$: The simulator generates random $(\kappa, s, c) \in (\mathbb{Z}_q)^3$. It computes $u = g^s\, y^{-c}$. If $\mathcal{H}(m, u)$ is already set, the simulator stops and fails (Event 1). Else, the simulator sets $h = \mathcal{H}(m, u) = g^\kappa$ and computes $z = (g^x)^\kappa$ —remark that $DL_h(z) = DL_g(y)(= x)$. Finally, the simulator computes $v = h^s\, z^{-c}$. If $\mathcal{G}(g, h, y, z, u, v)$ is already set, the simulator fails and stops (Event 2). Else, the simulator sets $\mathcal{G}(g, h, y, z, u, v) = c$ and returns the valid signature (z, s, c).

As we can see, the simulation is valid and indistinguishable from an actual signer, except for some events:

– Event 1: As (s, c) are random in $\mathbb{Z}_q \times \mathbb{Z}_q$, and as $u = g^s\, y^{-c}$, u is a random number in $G_{g,q}$ and so the probability that $\mathcal{H}(m, u)$ is already set is less than $\frac{q_{\mathcal{H}}+q_s}{q}$, for one signature query. For q_s signature queries, the failure probability is thus upper bounded by $\frac{q_s \cdot (q_{\mathcal{H}}+q_s)}{q}$.

- Event 2: From the simulation, the input tuples to the $\mathcal{G}$ oracle are of the form $(g, h, y, z, u, v) = (g, g^{\kappa}, y, y^{\kappa}, g^k, g^{\kappa k})$ for $k \in \mathbb{Z}_q$ and κ which is determined by relation $\mathcal{H}(g^k) = g^{\kappa}$. Then, the probability that $\mathcal{G}(g, h, y, z, u, v)$ is already set is less than $\frac{q_s + q_{\mathcal{G}}}{q}$. For q_s signature queries, the failure probability is thus upper bounded by $\frac{q_s \cdot (q_s + q_{\mathcal{G}})}{q}$.

Solving the CDH challenge (g, g^x, g^a): Except when these two rare events occur, attacker $\mathcal{A}$ returns, with probability ε, a valid signature forgery $\sigma = (z, s, c)$ on a message m that was not submitted to the signature oracle, with $h = \mathcal{H}(m, u) = (g^a)\, g^d$ for some d known to the simulator. Then, provided that $DL_h(z) = DL_g(y) = x$, the solution to the CDH challenge is $z\,(g^x)^{-d}$.

As for *EDL* signature scheme, $DL_h(z) = DL_g(y) = x$, except with a probability $\frac{q_{\mathcal{G}}}{q}$.

We get hence the conclusion that our variant of *EDL* signature scheme is tightly as secure as the Diffie-Hellman problem. The success probability ε' of our reduction satisfies

$$\varepsilon' \geq \varepsilon - q_s \left(\frac{2q_s + q_h}{q} \right) - \frac{q_h}{q}$$

Furthermore, the running time τ' of this simulation satisfies

$$t' \lesssim \tau + (6q_s + q_h)\tau_0$$

where τ_0 is the time required for an exponentiation. □

C Our Scheme With Even Shorter Signatures

(Section that was added Monday, January 23, 2006)

Classically (*e.g.,* as in Schnorr signature scheme), we show in this section that we can use in fact a shorter hash output. Namely, our scheme is unchanged, except that now, we define $\mathcal{G}$ output to be $\{0,1\}^{\ell_c}$, and no more in $\mathbb{Z}_q$. Our scheme becomes:

Global set-up: Let ℓ_p, ℓ_q and $\ell_c < \ell_q$ denote security parameters. Let also a cyclic group $G_{g,q}$, generated by g, for a ℓ_p-bit prime p and a ℓ_q-bit prime divisor q of $(p-1)$. Finally, let two hash functions, $\mathcal{H} : G_{g,q} \to G_{g,q}$ and $\mathcal{G} : \mathcal{M} \times (G_{g,q})^6 \to \{0,1\}^{\ell_c}$.

Key generation: The private key is a random number $x \in \mathbb{Z}_q$. The corresponding public key is $y = g^x$.

Signature: To sign a message $m \in \mathcal{M}$, one first randomly chooses $k \in \mathbb{Z}_q$, and computes $u = g^k$, $h = \mathcal{H}(u)$, $z = h^x$ and $v = h^k$. Next, one computes $c = \mathcal{G}(m, g, h, y, z, u, v)$ and $s = k + cx \bmod q$. The signature on m is $\sigma = (z, s, c)$.

Verification: To verify a signature $\sigma = (z, s, c) \in G_{g,q} \times \mathbb{Z}_q) \times \{0,1\}^{\ell_c}$ on a message $m \in \mathcal{M}$, one computes $u' = g^s\, y^{-c}$, $h' = \mathcal{H}(u')$, and $v' = h'^s\, z^{-c}$. The signature σ is accepted iff $c = \mathcal{G}(m, g, h', y, z, u', v')$.

As an advantage, signatures are shorter: they are $\ell_p + \ell_c + \ell_q$ bit long, instead of $\ell_p + 2\ell_q$.

We now prove that the scheme is still as secure, and then, we will see the gain that this little transformation can give.

Theorem 5. *Let $\mathcal{A}$ be an adversary against our modified signature scheme, which can produce, with success probability ε, an existential forgery under a chosen-message attack within time τ, after q_h queries to the hash oracles and q_s queries to the signing oracle, in the random oracle model. Then the Computational Diffie-Hellman problem can be solved with success probability ε' within time τ', with*

$$\varepsilon' \geq \varepsilon - 2q_s\left(\frac{q_s + q_h}{q}\right) - \frac{q_h}{2^{\ell_c}}$$

and

$$\tau' \lesssim \tau + (6q_s + q_h)\tau_0$$

where τ_0 is the time for an exponentiation in $G_{g,q}$.

We use most of the proof of Section 4.2, and just point out when the size of c is crucial.

Proof. We are given a group $G_{g,q}$ and a CDH challenge (g, g^x, g^a). We will use an attacker $\mathcal{A}$ against our modified signature scheme to solve this challenge, *i.e.*, to find g^{ax}. Our attacker $\mathcal{A}$, after $q_{\mathcal{H}}$ (resp. $q_{\mathcal{G}}$) hash queries to $\mathcal{H}$ (resp. $\mathcal{G}$) oracle and q_s signature queries, is able to produce a signature forgery with probability ε within time τ. We let $q_h = q_{\mathcal{H}} + q_{\mathcal{G}}$ and $y = g^x$.

Attacker $\mathcal{A}$ is run with the following simulation:

Initialisation: $\mathcal{A}$ is initialised with public key $y = g^x$ and public parameters $(g, q, G_{g,q})$.

Answering new $\mathcal{G}(m, g, h, y, z, u, v)$ query: The simulator returns a random number in $\{0,1\}^{\ell_c}$.

Answering new $\mathcal{H}(u)$ query: The simulator generates a random number $d \in \mathbb{Z}_q$, and returns $(g^a)\, g^d$. All queries u are stored in a list called U-List.

Answering signatures query on $m \in \mathcal{M}$: The simulator generates randomly $(\kappa, s, c) \in (\mathbb{Z}_q)^2 \times \{0,1\}^{\ell_c}$. Then, it computes $u = g^s\, y^{-c}$. If $\mathcal{H}(u)$ is already set, the simulator stops (Event 1). Else, the simulator sets $h = \mathcal{H}(u) = g^\kappa$ and computes $z = (g^x)^\kappa$ —remark that $DL_h(z) = DL_g(y) (= x)$. Finally, the simulator computes $v = h^s\, z^{-c}$. If $\mathcal{G}(m, g, h, y, z, u, v)$ is already set, the simulator stops and fails (Event 2). Else, the simulator sets $\mathcal{G}(m, g, h, y, z, u, v) = c$, and returns the valid signature (z, s, c). All u's computed during signature queries are stored in a list called Υ-List

As we can see, this simulator is valid and indistinguishable from an actual signer, except for some events:

- Event 1: As u is a random number in $G_{g,q}$, the probability that the $\mathcal{H}(u)$ is already set is less than $\frac{q_s+q_\mathcal{H}}{q}$, for one signature query. For q_s signature queries, the failure probability is thus upper bounded by $\frac{q_s\cdot(q_s+q_\mathcal{H})}{q}$.
- Event 2: From the simulation, the input tuples to the $\mathcal{G}$ oracle are of the form $(m, g, h, y, z, u, v) = (m, g, g^\kappa, y, y^\kappa, g^k, g^{\kappa k})$ for $k \in \mathbb{Z}_q$ and κ which is unknown before the signature query, as $h = \mathcal{H}(u)$ was not known by the signer (in fact, Event 2 happens only if Event 1 does not). Then, the probability that $\mathcal{G}(m, g, h, y, z, u, v)$ is already set is less than $\frac{q_s+q_\mathcal{G}}{q^2}$. For q_s signature queries, the failure probability is thus upper bounded by $\frac{q_s\cdot(q_s+q_\mathcal{G})}{q^2} \leq \frac{q_s\cdot(q_s+q_\mathcal{G})}{q}$.

As a conclusion, except with a probability smaller than $\delta_{sim} = q_s\big(\frac{q_h+2q_s}{q}\big)$, the simulation is successful.

In other words, with a probability $\varepsilon_{sim} \geq \varepsilon - \delta_{sim}$, the attacker $\mathcal{A}$ is able to return a valid signature forgery $(\hat{z}, \hat{s}, \hat{c})$ on a message $\hat{m} \in \mathcal{M}$ that was never submitted to the signature oracle. The simulator deduces from this forgery the corresponding tuple $(\hat{u}, \hat{v}, \hat{h})$, by the following computations: $\hat{u} = g^{\hat{s}}\, y^{-\hat{c}}$, $\hat{h} = \mathcal{H}(\hat{u})$, and $\hat{v} = \hat{h}^{\hat{s}}\, \hat{z}^{-\hat{c}}$. Notably, if $\mathcal{H}(\hat{u})$ has not been queried to $\mathcal{H}$ oracle by the attacker or set by the signature oracle, the simulator queries it to $\mathcal{H}$ oracle itself. Hence, $\hat{u}$ is a member of U-List or a member of Υ-List.

Solving the CDH challenge (g, g^x, g^a). At this step, once the forgery is returned by the attacker, there are two cases.

In the first case, $\hat{u}$ is a member of U-List. We then write $\hat{u} = g^k$, $\hat{v} = \hat{h}^{k'}$ and $\hat{z} = \hat{h}^{x'}$, and we get, as the signature is valid, $k = \hat{s} - \hat{c}x \bmod q$ and $k' = \hat{s} - \hat{c}x' \bmod q$. Then, if $x \neq x'$, we have $\hat{c} = \mathcal{G}(\hat{m}, g, \hat{h}, y, \hat{h}^{x'}, g^k, \hat{h}^{k'}) = \frac{k-k'}{x'-x} \bmod q$.

Here, we can see that the size of $\mathcal{G}$ output is important: Indeed, a forger could try to find $(k, k', \hat{m}, \hat{h})$ and $x' \neq x$, so that $\mathcal{G}(\hat{m}, g, \hat{h}, y, \hat{h}^{x'}, g^k, \hat{h}^{k'}) = \frac{k-k'}{x'-x} \bmod q$. Fortunately, as the message $\hat{m}$ is new, $\mathcal{G}(\hat{m}, g, \hat{h}, y, \hat{h}^{x'}, g^k, \hat{h}^{k'})$ was not set during a signature query, and so this can happen with probability smaller than $\frac{q_\mathcal{G}}{2^{\ell_c}}$. Else, we know that $DL_{\hat{h}}(\hat{z}) = DL_g(y)(= x)$.

Apart this error, the simulator receives from the attacker a signature with $\hat{z} = \hat{h}^x$, and it knows d such that $\hat{h} = \mathcal{H}(\hat{u}) = (g^a)\, g^d$. The simulator can return the solution to the CDH challenge, which is $\hat{z}\,(g^x)^{-d}$. As a partial conclusion, in this first case, the forgery is successfully used to solve the CDH challenge, except with a probability smaller than $\delta_1 = \frac{q_h}{2^{\ell_c}}$.

In the second case, $\hat{u}$ is not a member of U-List, and so is a member of Υ-List. This case can happen as there is no message in the input of $\mathcal{H}$, and so we can imagine that the attacker reuse a u that corresponds to a u of a signature given by the signature oracle. Then, the simulator can recover from its log files all quantities that correspond to this $u = \hat{u}$, *i.e.*, h, v, z, s, c and m.

At this moment, we can see that we have $u = g^s y^{-c} = \hat{u} = g^{\hat{s}} y^{-\hat{c}}$. It is exactly the kind of hypothesis that is used by the forking lemma to prove a (loose) security. However, here, this equality is not obtained by restarting the attacker (as it is done in the forking lemma), but just by construction. More precisely, we can recover easily the private key x, as far as $\hat{c} \neq c \bmod q$.

As $u = \hat{u}$, it follows that $h = \hat{h}$, $v = \hat{v}$. Furthermore, except if $DL_{\hat{h}}(\hat{z}) \neq DL_g(y)(= x)$, which happens also in this case with probability smaller than $\delta_1 = \frac{q_h}{2^{\ell_c}}$, we have $z = \hat{z}$.[8] Then, to have an event $\hat{c} = c \bmod q$, one needs that $\mathcal{G}(m, g, h, y, z, u, v) = \mathcal{G}(\hat{m}, g, h, y, \hat{z}, u, v) \bmod q$. As the message $\hat{m}$ is new, and so different from m, it is very improbable: it occurs with a probability smaller than $\frac{q_{\mathcal{G}}}{\min(2^{\ell_c}, q)}$. Hence, except an error with a probability smaller than $\delta_2 = \frac{q_h}{\min(2^{\ell_c}, q)}$, we have $\hat{c} \neq c \bmod q$, and so we can recover the private key x: equation $s - xc = \hat{s} - x\hat{c} \bmod q$ gives $x = \frac{s - \hat{s}}{c - \hat{c}} \bmod q$. We can see that this second case gives not only the solution to the CDH challenge, but also the solution to the discrete logarithm.

As a conclusion, we can see that in both cases, our simulator can transform the forgery given by the attacker into the solution to the CDH challenge.

Putting all together, the success probability ε' of our reduction satisfies $\varepsilon' \geq \varepsilon - \delta_{sim} - \max(\delta_1, \delta_2)$, which gives, using $q_{\mathcal{H}} + q_{\mathcal{G}} = q_h$ and $\ell_c \leq \ell_q$,

$$\varepsilon' \geq \varepsilon - q_s \left(\frac{2q_s + q_h}{q} \right) - \frac{q_h}{2^{\ell_c}}$$

and the running time τ' satisfies

$$\tau' \lesssim \tau + (6q_s + q_h)\tau_0 \ .$$

□

Then, one can adapt the discussion of Section 4.4. If the targeted security level is $\kappa = 80$, it is sufficient to use $\kappa' = 88$ (and hence $\ell_q \geq 176$) and $\ell_c = 82$. Hence, the size of this modified version of our signature scheme, in a subgroup of $\mathbb{F}_p{}^*$ (taking $\ell_p = 1024$, $\ell_q = 176$ and $\ell_c = 82$), is $1024 + 176 + 82 = 1282$ bits. In the elliptic curve setting, the gain is even more sensible, as z can be represented with a length around $\ell_q = 176$: signature sizes is then $176 * 2 + 82 = 434$ bits. This is very efficient for a tight signature on CDH.

[8] But even if $DL_{\hat{h}}(\hat{z}) \neq DL_g(y)(= x)$, it does not matter.