A Signature Scheme as Secure as the Diffie-Hellman Problem

Eu-Jin Goh and Stanisław Jarecki

Computer Science Department, Stanford University, Stanford CA 94305-9045
eujin@cs.stanford.edu, stasio@theory.stanford.edu

Abstract. We show a signature scheme whose security is *tightly* related to the Computational Diffie-Hellman (CDH) assumption in the Random Oracle Model. Existing discrete-log based signature schemes, such as ElGamal, DSS, and Schnorr signatures, either require non-standard assumptions, or their security is only loosely related to the discrete logarithm (DL) assumption using Pointcheval and Stern's "forking" lemma. Since the hardness of the CDH problem is widely believed to be closely related to the hardness of the DL problem, the signature scheme presented here offers better security guarantees than existing discrete-log based signature schemes. Furthermore, the new scheme has comparable efficiency to existing schemes.
The signature scheme was previously proposed in the cryptographic literature on at least two occasions. However, no security analysis was done, probably because the scheme was viewed as a slight modification of Schnorr signatures. In particular, the scheme's tight security reduction to CDH has remained unnoticed until now. Interestingly, this discrete-log based signature scheme is similar to the trapdoor permutation based PSS signatures proposed by Bellare and Rogaway, and has a tight reduction for a similar reason.

Keywords: Signature Schemes. Computational Diffie-Hellman. Discrete Logarithm. Exact Security. Tight Reductions. Random Oracle Model.

1 Introduction

From the computational complexity view of cryptography, a proof of security for a signature scheme follows if there exists a polynomial time *reduction* algorithm with the following ability: the reduction algorithm can use a poly-time algorithm that forges a signature to construct another poly-time algorithm that solves a hard computational problem. If no poly-time algorithm that solves the computational problem exist, then the existence of such a reduction implies that the signature scheme is also unbreakable in poly-time.

However, such security arguments are asymptotic. In the case of discrete-log signature schemes, forging signatures is infeasible in prime order groups where the prime is larger than some threshold length. In practice, we would like to know *exactly* how long a prime to use in order to impose a sufficiently infeasible computational bound on an adversary.

For this purpose, Bellare and Rogaway started an exact method of security analysis [BR96] where more attention is paid to the computational efficiency of the reduction algorithm. This method allows us to quantify the relation between the difficulty of forging signatures and the hardness of the underlying computational problem. As Micali and Reyzin [MR02] put it, if the reduction is efficient and hence the relative hardness of forging and that of breaking the underlying computational assumption is close, we call the reduction *tight*. If the reduction is less efficient, we call it *close*, and if it is significantly less efficient, we call it *loose*. These terms are imprecise and should be only used to compare different reduction algorithms to deduce the relative strength of different security claims.

State of research on signature schemes. The standard definition of the security of signature schemes, together with the first construction that satisfies it, was given by Goldwasser, Micali, and Rivest [GMR88]. The results of Naor, Yung, and Rompel [NY89,Rom90] imply a signature scheme based on the discrete-log assumption, but the construction is inefficient and not stateless. Recently, practical signatures schemes [GHR99,CS00] based on the strong RSA assumption were proposed.

There are practical signature schemes whose security is tightly related to weaker computational assumptions such as RSA and factoring, but their security is (so far) proven only in the Random Oracle model. The Random Oracle model was implicitly considered by Fiat and Shamir [FS86] and rigorously defined by Bellare and Rogaway [BR93]. In this model, we assume that the adversarial algorithms work equally well when a hash function like SHA-1 or MD5 is replaced by a true random function. This restriction on the adversarial capability appears arbitrary [CGH98], but in practice, given a carefully implemented hash function, there are no examples of adversarial algorithms which break cryptographic schemes by exploiting the properties of a hash function. An efficient signature scheme with a tight security reduction to trapdoor permutations was given in this model by Bellare and Rogaway [BR96]. Recently, Micali and Reyzin [MR02] showed a number of signature schemes whose security is tightly related to factoring in this model.

Loose security of discrete-log based signatures. Some discrete-log based signature schemes, such as ElGamal [ElG85] and DSS [NIS94], require non-standard assumptions. Other schemes, such as Schnorr signatures [Sch89], the "Modified ElGamal" signatures of Pointcheval and Stern [PS96], and some DSS variants [BPVY00], have security proofs that are only loosely related to the discrete-log problem. The latter schemes with loose reductions follow a Fiat-Shamir methodology [FS86], which is used to convert any *commit-challenge-respond* identification scheme into a signature scheme secure in the Random Oracle model. The only known reduction converting a forging algorithm for a Fiat-Shamir based signature scheme into an algorithm that breaks the underlying computational problem is the "forking" lemma [PS96]. However, this reduction is inefficient: to break the computational problem with a probability comparable to the success probability of the signature forger, the reduction algorithm needs to execute a full run of the forging algorithm q_H times, where q_H denotes the number of hash function queries made by the forger.

If we take 2^n hash function evaluations as our infeasible computational bound, the adversarial algorithm is allowed to make $q_H = 2^n$ hash function queries. The forking lemma then implies that if an adversary that breaks Schnorr signatures in 2^n steps exists, then there is an algorithm that computes discrete logarithms in 2^{2n} steps. Therefore, to be provably secure, all signature schemes based on the discrete-log problem with security parameter n must work in a group where the discrete-log problem is believed secure with security parameter $2n$. In a traditional discrete-log system of a prime field $\mathbb{Z}_p^*$, the index-calculus method of breaking the discrete-log problem works in time about $O(\exp(\sqrt[3]{|p|}))$. Hence, a factor of α increase in the security parameter implies a α^3 increase in the length of the modulus p. For example, if the discrete-log problem in a prime field is believed to be infeasible for 1000 bit primes, the "forking" lemma reduction tells us that Schnorr signatures are secure only in a field modulo a 8000 bit prime.

Our contribution: signatures as secure as Diffie-Hellman. In this paper, we present a signature scheme whose security is tightly related to the Computational Diffie-Hellman (CDH) assumption in the Random Oracle model. This signature scheme was previously considered by Chaum and Pedersen [CP92], and by Jakobsson and Schnorr [JS99]. However, the scheme appeared without a security analysis, probably because it was viewed as a slight modification of Schnorr signatures. In particular, the scheme's *tight security reduction to CDH* has remained unnoticed until now.

Since the hardness of the CDH problem is widely believed to be closely related to the hardness of the DL problem [Sho97,BL96,MW99], our signature scheme offers better security guarantees than well-known discrete-log based signature schemes. Moreover, by the results of Maurer and Wolf [MW99], we can relate the security of our signature scheme directly to the hardness of the discrete-log problem for a large class of groups in which the discrete-log problem is believed hard. The resulting security degradation is a factor of about 2^{43} for $\log_2(\log_2(q)) \approx 7$, where q is the group size.[1] This degradation is much better than the $q_H = 2^{80}$ security degradation carried by the forking lemma, which so far is the only known security argument for discrete-log based schemes in the Random Oracle model.

Related Work. The main technical trick allowing us to avoid the forking lemma in the security proof is the replacement of a zero-knowledge *proof of knowledge* with a zero-knowledge *proof* of equality of discrete logarithms. Using a zero knowledge proof of equality allows for a one-pass simulation in the security proof, and hence results in a tight security reduction. It is interesting to note that the same technical idea was used by Gennaro and Shoup to achieve a

[1] The size of q recommended by Lenstra [LV01] for the year 2003 is 129 bits long.

one-pass simulation in a proof of CCA security for a variant of an ElGamal encryption [SG98].

Recently, Micali and Reyzin showed a different way of avoiding the forking lemma in the security reduction of signature schemes built using the Fiat-Shamir method [MR02]. Their idea works for a class of signature schemes based on factoring, and results in tight security reductions for such signatures. However, their method does not apply to discrete-log based schemes.

Our scheme is similar to the PSS signature scheme [BR96] which has security that is tightly related to the security of a trap-door permutation (e.g. RSA or factoring). Our scheme is also similar to the undeniable signature scheme proposed by Okamoto and Pointcheval [OP01], and to the BLS short signature scheme [BLS01], both secure under the "Gap Diffie-Hellman" assumption.

We remark that in the Generic Group model introduced in the work of Shoup [Sho97], Schnorr signatures are as secure as the discrete-log problem. However, the security results in the generic group model imply security only against a restricted class of adversarial algorithms. Namely, this model considers only algorithms that are oblivious to the representation of the elements in the group. This is a significant restriction because there are known adversarial algorithms, such as the index-calculus method for finding discrete logarithms in $\mathbb{Z}_p^*$, which do not fall into this category.

2 Definitions

In this section, we present some definitions and notational conventions. We first recall the definition of a signature scheme.

Definition 1 (Signature Scheme).
A signature scheme ($\texttt{Gen}, \texttt{Sign}, \texttt{Ver}$*) is a triple of probabilistic algorithms:*

- *The key generation algorithm* `Gen` *that when given a security parameter* 1^n *as input, outputs a private key (sk) and a public key (pk).*
- *The signature algorithm* `Sig` *that when given sk and a message* M *as inputs, produces a signature* σ *as output.*
- *The (usually deterministic) verification algorithm* `Ver` *that when given input* (pk, M, σ)*, either accepts or rejects, such that if* $(sk, pk) \leftarrow \texttt{Gen}(1^n)$ *and* $\sigma \leftarrow \texttt{SIG}(sk, M)$*, then* $\texttt{Ver}(pk, M, \sigma) = \texttt{accept}$.

We consider signature schemes that are secure against existential forgery under an adaptive chosen message attack (CMA) in the Random Oracle model. This definition is an adaptation of the standard existential CMA security definition [GMR88] to the Random Oracle model. In the definition below, the "hash function oracle" is an ideal random function.

Definition 2 (Existential CMA Security of a Signature Scheme).
A probabilistic algorithm $\mathcal{F}$ *is said to* $(t, q_H, q_{sig}, \epsilon)$*-break a signature scheme if after running in at most* t *steps, making at most* q_H *adaptive queries to the*

hash function oracle, and requesting signatures on at most q_{sig} adaptively chosen messages, $\mathcal{F}$ outputs a valid forgery (M, σ) on some new message M (i.e. a message on which $\mathcal{F}$ has not requested a signature) with probability at least ϵ, where the probability is taken over the coins of $\mathcal{F}$, the `Gen` *and* `Sig` *algorithms, and the hash function oracle.*

We say that a signature scheme is $(t, q_H, q_{sig}, \epsilon)$-secure if no forger can $(t, q_H, q_{sig}, \epsilon)$-break it.

Discrete-Log Setting and Notation. We consider groups where the discrete-log problem is hard. For notational convenience, we only consider prime order subgroups of the multiplicative group $\mathbb{Z}_p^*$ with prime p. However, our results also carry over to other groups, such as those built on elliptic curves.

Let p and q be large primes. Also let g be a generator of subgroup $G_{g,p} = \{g^0, \ldots, g^{q-1}\}$ with order q in the multiplicative group $\mathbb{Z}_p^*$. We use the triple (g, p, q) to describe the group $G_{g,p}$. All the algorithms discussed in this paper implicitly take the triple (g, p, q) as input.

Group operations in $G_{g,p}$ are always modulo p and operations on secret values are always modulo q. For notational convenience, we will omit the "$(\bmod\, p)$" and "$(\bmod\ q)$" markers. We denote the bit length of q by $|q| = n_q$. The notation $a \stackrel{R}{\leftarrow} S$ means that a is picked uniformly at random from set S.

The security of our signature scheme relies on the hardness of the Computational Diffie-Hellman (CDH) problem [DH76]:

Definition 3 (CDH assumption).
A probabilistic algorithm $\mathcal{A}$ is said to (t, ϵ)-break CDH in a group $G_{g,p}$ if on input (g, p, q) and (g^a, g^b) and after running in at most t steps, $\mathcal{A}$ computes the Diffie-Hellman function, $DH_{g,p}(g^a, g^b) = g^{ab}$, with probability at least ϵ, where the probability is over the coins of $\mathcal{A}$ and (a, b) chosen uniformly from $\mathbb{Z}_q \times \mathbb{Z}_q$.

We say that group $G_{g,p}$ is a (t, ϵ)-CDH group if no algorithm (t, ϵ)-breaks CDH in $G_{g,p}$.

As previously mentioned, there is strong evidence that the CDH problem is closely related to the hardness of computing discrete logarithms. In particular, we know an efficient reduction from the discrete-log problem to the CDH problem for many easily constructible groups $G_{g,p}$ [MW99].

3 The *EDL* Signature Scheme

We present the *EDL* signature scheme and prove that its security is tightly related to the CDH problem. In the introduction, we mentioned that the *EDL* signature scheme was previously proposed in the literature [CP92,JS99]. However, the security properties of this scheme have not been examined.

The scheme proceeds as follows: the private key is $x \in \mathbb{Z}_q$ and the public key is $y = g^x$. To sign a message m, the signer hashes the message m with a random string r of size $n_r = 111$ to obtain $H(m, r) = h$, computes $z = h^x$, and outputs as a signature (z, r) together with a non-interactive zero-knowledge proof (ZKP) that $DL_g(y) = DL_h(z)$.[2]

We now describe the *EDL* scheme in full detail: let (p, q, g) be a discrete-log triple defining a group $G_{g,p}$. Let $n_r = 111$. Let H and H' be (ideal) hash functions where $H : \{0,1\}^* \to G_{g,p}$ and $H' : (G_{g,p})^6 \to \mathbb{Z}_q$.

- Key Generation Algorithm (`Gen`)
  Pick a random $x \xleftarrow{R} \mathbb{Z}_q$ as the private key. The corresponding public key is $y \leftarrow g^x$.
- Signing Algorithm (`Sign`)
  The inputs are a secret key $x \in \mathbb{Z}_q$ and a message $M \in \{0,1\}^*$.
  First pick a random $r \xleftarrow{R} \{0,1\}^{n_r}$. Compute $h \leftarrow H(m, r)$ and $z \leftarrow h^x$.
  Next, prepare a non-interactive ZKP that $DL_h(z) = DL_g(y)$: pick a random $k \xleftarrow{R} \mathbb{Z}_q$. Compute $u \leftarrow g^k$, $v \leftarrow h^k$, $c \leftarrow H'(g, h, y, z, u, v) \in \mathbb{Z}_q$, and $s \leftarrow k + cx$.
  The signature is $\sigma \leftarrow (z, r, s, c)$.
- Verification Algorithm (`Ver`)
  The inputs are a public key y, a message M, and a signature $\sigma = (z, r, s, c)$.
  First compute $h \leftarrow H(m, r)$, $u \leftarrow g^s y^{-c}$, and $v \leftarrow h^s z^{-c}$. Then compute $c' = H'(g, h, y, z, u, v)$. If $c = c'$, output `valid`. Otherwise, output `invalid`.

Similarity to PSS. The *EDL* signature scheme is similar to the PSS signature scheme [BR96] (and the RSA identification protocol): the verifier sends a random value h, and the prover responds with $z = h^x$. Recall that in the RSA identification protocol, the inverse of the secret key x in the exponent group $\mathbb{Z}_q$ is the public key, and verification is done by checking if $z^{(1/x)} = h$. However, in the *EDL* signature scheme, the public key is g^x and verification is achieved by a zero-knowledge proof that $DL_g(y) = DL_h(z)$.

Because *EDL* and PSS have a similar structure, they both have tight security reductions to the underlying hard problem of "exponentiating to the secret x committed in the public key". In the case of PSS, the hard problem is inverting the RSA trap-door permutation. And in the case of *EDL*, the hard problem is computing the CDH function. Note that a successful forgery in the *EDL* scheme requires computing h^x on a random public key $y = g^x$, and a random h returned by the hash function. On input g^x, g^w, a simulator easily embeds g^w into the h values returned by the hash oracle, and then recovers the CDH value $(g^w)^x$ from the successful forgery. This simulator only needs one pass of the simulated CMA attack to provide a successful forgery for recovering g^{wx}. Hence, the reduction from the CDH problem to security of the *EDL* scheme is tight.

Avoiding the Proof of Knowledge. All known signature schemes whose security is reducible to the discrete-log assumption [PS96,OO98,BPVY00] are based on zero-knowledge *proofs of knowledge*. Replacing the zero-knowledge proof of knowledge with a zero-knowledge *proof* (of discrete-log equality) allows us to

[2] Section 4 explains the derivation of the randomness size n_r.

avoid using the "forking lemma" [PS96] in the security proof. Therefore, *EDL* does not suffer from the $1/q_H$ degradation in security during the reduction (where q_H is the number of hash queries made by the forger).[3]

The public-coin proof underlying the *EDL* scheme is the proof of equality of two discrete logarithms. This proof proceeds as follows: the prover picks $k \in \mathbb{Z}_q$, sends $u = g^k, v = h^k$ to the verifier. The verifier picks a public random challenge $c \in \mathbb{Z}_q$, and the prover responds with $s = k + cx$. The verifier checks that $g^s = uy^c$ and $h^s = vz^c$. This proof system is public-coin zero-knowledge because a simulator given inputs g, h, y, z can pick c and s at random in $\mathbb{Z}_q$ and compute u and v from them. This proof system is also a proof of knowledge of x such that $g^x = y$ and $h^c = z$, but we do not rely on this property in our security analysis. In our analysis, it is sufficient that this proof system is an interactive proof where the prover's probability of cheating is at most $1/q$. This condition is sufficient because if $x = DL_g(y)$ is not equal to $x' = DL_h(z)$, then the prover can pass only if the public coin c is $(k - k')/(x' - x)$, where $k = DL_g(u)$ and $k' = DL_h(v)$.

This proof system was first proposed [CEvdG87] in a slightly different variant of zero-knowledge against any verifier and only 1/2 soundness.[4] This proof system can also be viewed as an extension of Schnorr's public-coin proof of knowledge of discrete logarithm [Sch89], and it is indeed also a proof of knowledge. However, we do not use the proof-of-knowledge property of this proof system in our scheme.

Efficiency Considerations. The signature size and the signing and verification costs of our scheme are larger than for traditional discrete-log based signature schemes like DSS or Schnorr, but only if one compares these costs for the two schemes working *in the same group* $G_{g,p}$. Our construction offers better security guarantees than the traditional discrete-log based schemes, and can therefore be used in much smaller groups. Section 4.1 has more details about the complexity/size vs. security bound trade-offs implied by our work.

The signature size is $|p| + 2|q| + n_r$. Signing takes three exponentiations and verification takes two multi-exponentiations [BGMW92]. The cost of a single (two-element) multi-exponentiation is about 20% more than the cost of a single exponentiation, assuming Montgomery squaring is used. In Section 4, we

[3] Intuitively, the security of signature schemes based on zero-knowledge proofs of knowledge relies on the simulator's ability to re-run the forger several times. If the simulator gets a successful forgery *on the same message* on any two runs, the simulator can *extract* the secret key x from the forger by the proof of knowledge property of the proof system. Hence, the simulator can solve the discrete logarithm problem of finding $DL_g(y)$. The forking lemma [PS96] shows that the probability of the simulator obtaining two such forgeries is at least $1/q_H$ of the forger's probability of success (because the simulator has to guess the specific message among the q_H messages submitted to the hash oracle by the forger that will be used by the forger to output a signature).

[4] See also [CS97] for a modern presentation of this protocol and for techniques for constructing discrete-log proof systems.

will compare *EDL* with other signature schemes, taking into account both the efficiency *and* the security parameters.

We point out two possible efficiency improvements. Firstly, if the signature includes element v (with the signature size increasing to $2|p| + 2|q| + n_r$), the verification time can be reduced to a single (five element) multi-exponentiation. The cost of this five element multi-exponentiation is about 70% more than the cost of a single exponentiation. Hence, we obtain a 30% reduction in verification cost. Verification is done using a simple trick similar to those used to batch signature verification [BGR98]: the verifier picks a random $\alpha \in \mathbb{Z}_q$, computes u as $g^s y^{-c} h^{s\alpha} z^{-c\alpha} v^{-\alpha}$, and then carries out the normal verification procedure with u and v. This trick negligibly increases the probability of accepting an invalid signature by $1/q$.

Secondly, unlike the Schnorr signature scheme, *EDL* signatures are not efficient on-line. However, using the trick of signing random commitments [ST01], the *EDL* signature scheme can be as efficient on-line as the Schnorr signature. The off-line cost increases to five exponentiations and the verification cost increases to about two exponentiations.

3.1 Security Proof

The following theorem proves a tight security reduction from the hardness of the CDH problem to the CMA security of the *EDL* scheme in the Random Oracle model. We denote the cost of a long exponentiation in $G_{g,p}$ by $\mathrm{C}_{\exp}(G_{g,p})$. Since the exponentiation cost is the primary factor in the cost of reduction, we ignore the costs of other operations in the theorem below.

Theorem 1. *If G is a (t', ϵ')-CDH group then the* EDL *signature scheme is $(t, q_H, q_{sig}, \epsilon)$-secure against existential forgery on adaptive chosen message attack in the Random Oracle model, where*

$$t \le t' - q_H \cdot \mathrm{C}_{\exp}(G_{g,p})$$
$$\epsilon \ge \epsilon' + q_{sig} \cdot q_H \cdot 2^{-n_r} + q_H \cdot 2^{-n_q}$$

Proof. Let $\mathcal{F}$ be a forger that $(t, q_H, q_{sig}, \epsilon)$-breaks *EDL*. We construct a "simulator" algorithm $\mathcal{S}$ which takes (p, q, g) and (g^a, g^b) as inputs. Algorithm $\mathcal{S}$ uses the $\mathcal{F}$ algorithm to compute the $DH_{g,p}(g^a, g^b)$ function in t' steps and ϵ' probability where

$$t' \approx t + (q_H + 4.4 \cdot q_{sig}) \cdot \mathrm{C}_{\exp}(G_{g,p}) \tag{1}$$
$$\epsilon' = \epsilon - \left(q_{sig} \cdot q_H \cdot 2^{-n_r} + q_{sig} \cdot (q_H + q_{sig}) \cdot 2^{-2 \cdot n_q} + 2^{-n_q} + q_H \cdot 2^{-n_q} \right) \tag{2}$$

and the probability goes over (a, b) in $\mathbb{Z}_q \times \mathbb{Z}_q$ and the randomness used by $\mathcal{S}$ and $\mathcal{F}$. Since $n_r < n_q$, it follows that $2^{-n_r} \gg 2^{-2n_q}$ for sufficiently large n_q. Assuming that $q_H \gg q_{sig} \gg 1$, the theorem follows.

Algorithm $\mathcal{S}$ simulates a run of a signature scheme *EDL* to the forger $\mathcal{F}$. Algorithm $\mathcal{S}$ answers $\mathcal{F}$'s hash function queries, signature oracle queries, and it tries to translate $\mathcal{F}$'s possible forgery (m, σ) into an answer to the $DH_{g,p}(g^a, g^b)$ function. Algorithm $\mathcal{S}$ starts the simulation by providing (p, q, g) and the public key $y = g^a$ as input to $\mathcal{F}$. Then algorithm $\mathcal{S}$ answers $\mathcal{F}$'s queries as follows.

Answering H-oracle Queries. If the forger $\mathcal{F}$ provides a new query (m, r) as input to the H-oracle, algorithm $\mathcal{S}$ embeds g^b into its answer by picking d at random in $\mathbb{Z}_q$, and outputting $H(m, r)$ as $h = (g^b)^d$.

Answering H′-oracle Queries. The simulator $\mathcal{S}$ answers all new answers queries to the H' oracle completely at random.

Answering Signature Queries. Suppose the forger asks for a signature on message m. Algorithm $\mathcal{S}$ has to create a valid signature tuple without knowing the secret key. In the process, algorithm $\mathcal{S}$ defines some values of the two hash functions H and H'. The simulator proceeds as follows:

1. $\mathcal{S}$ picks a random $r \xleftarrow{R} \{0,1\}^{n_r}$. If H has been queried on inputs (m, r), it aborts.
2. Otherwise, $\mathcal{S}$ picks a random $\kappa \xleftarrow{R} \mathbb{Z}_q$, sets $z = y^\kappa$ and $h = g^\kappa$, and defines $H(m, r) \triangleq h$. Note that $DL_h(z) = DL_g(y)$ where $h = H(m, r)$.
3. $\mathcal{S}$ simulates the non-interactive proof of discrete logarithm equality in a standard way: $\mathcal{S}$ picks random $c \xleftarrow{R} \mathbb{Z}_q$, $s \xleftarrow{R} \mathbb{Z}_q$, sets $u = g^s y^{-c}$ and $v = h^s z^{-c}$.
4. If H' has been queried on inputs (g, h, y, z, u, v) before, $\mathcal{S}$ aborts. Otherwise, it sets $H'(g, h, y, z, u, v) \triangleq c$ and returns the tuple (z, r, s, c) as the signature of m.

Solving the CDH Problem. If the forger $\mathcal{F}$ returns a valid message and signature pair (m, σ) (where $\sigma = (z, r, s, c)$) for some previously unsigned m, then algorithm $\mathcal{S}$ tries to translate this forgery into computing g^{ab} as follows: If $\mathcal{F}$ has not queried the H oracle on (m, r), $\mathcal{S}$ aborts. Otherwise $h = H(m, r) = g^{bd}$ for some d known to simulator $\mathcal{S}$, and the simulator $\mathcal{S}$ outputs $z^{1/d}$ and stops. If it holds that $DL_g(y) = DL_h(z)$, then $z = h^a$ and hence $z = g^{abd}$, in which case algorithm $\mathcal{S}$'s output is equal to g^{ab}.

Let $\epsilon_{\mathtt{abort}}$ be the probability that algorithm $\mathcal{S}$ aborts the simulation and let ϵ_{DL} be the probability that $\mathcal{F}$ produces a valid forgery but $DL_g(y) \neq DL_h(z)$. Observe that the computational view shown to the forger by the simulator has the same distribution as the forger's conversation with an actual signature scheme and random hash functions except for the probability $\epsilon_{\mathtt{abort}}$. Hence the probability that $\mathcal{S}$ outputs a correct solution to the CDH challenge $DH_{g,p}(g^a, g^b)$ is at least $\epsilon - (\epsilon_{\mathtt{abort}} + \epsilon_{DL})$. We now upper bound the $\epsilon_{\mathtt{abort}} + \epsilon_{DL}$ term.

1. The simulator $\mathcal{S}$ might abort at Step 1 of the signature oracle simulation. This event occurs if $\mathcal{S}$ chooses a r that was previously given as input to the H-oracle. Since there are at most q_H such r's, the probability of aborting is at most $q_H \cdot 2^{-n_r}$. Therefore, the probability that the simulator aborts at Step 1 for any of the q_{sig} signature queries is less than $q_{sig} \cdot q_H \cdot 2^{-n_r}$.

2. Similarly, simulator $\mathcal{S}$ only aborts at Step 4 if it has run into an input string (g, h, y, z, u, v) on which H' has been already queried. Note that this input string can be represented as (g, g^k, y, y^k, u, u^k), where u, k are chosen at random in $G_{g,p} \times \mathbb{Z}_q$. Since there are at most $q_H + q_{sig}$ such strings on which H' was previously queried on, the probability of collision is at most $(q_H + q_{sig}) \cdot 2^{-2 \cdot n_q}$. Thus, the probability of $\mathcal{S}$ aborting at any time in the simulation at this step is at most approximately $q_{sig} \cdot (q_H + q_{sig}) \cdot 2^{-2 \cdot n_q}$.
3. Let NH be the event that the forger $\mathcal{F}$ does not query the H-oracle on the (m, r) which it outputs as part of the forgery. Let NQ be the event that $DL_g(y) \neq DL_h(z)$. We want to compute an upper bound for the probability $\Pr[NH \vee NQ]$. Together with the probability of aborting in the signature simulation phase, this allows us to derive an upper bound for $\epsilon_{\texttt{abort}} + \epsilon_{DL}$. Observe that $\Pr[NH \vee NQ] = \Pr[NH \wedge \neg NQ] + \Pr[NQ]$. We first calculate $\Pr[NH \wedge \neg NQ]$. $\Pr[NH \wedge \neg NQ]$ is given by the probability that the forger $\mathcal{F}$ outputs a z such that $z^{-x} = h = H(M, r)$ on a successful forgery. This probability is 2^{-n_q}.
   We now calculate $\Pr[NQ]$. Let $u = g^k, v = h^{k'}, y = g^x$, and $z = h^{x'} \neq h^x$. Since the signature is valid, we must have $u = g^s y^{-c}$ and $v = h^s z^{-c}$ for $c = H'(g, h, y, z, u, v)$. Considering only exponents, we have $k = s - xc$ and $k' = s - x'c$. Hence, $H'(g, h, g^x, h^{x'}, g^k, h^{k'}) = c = (k - k')/(x' - x)$. Since H' is a random oracle, the probability that this equation holds for an input to H' is $1/|\mathbb{Z}_q| = 2^{-n_q}$. Therefore, the probability that $\mathcal{F}$ finds such a c in q_H oracle queries is at most $q_H \cdot 2^{-n_q}$.

Summing the probabilities, we see that algorithm $\mathcal{S}$ solves the CDH problem with probability (approximately) at least $\epsilon - q_{sig} \cdot q_H \cdot 2^{-n_r} + q_{sig} \cdot (q_H + q_{sig}) \cdot 2^{-2 \cdot n_q} - 2^{-n_q} + q_H \cdot 2^{-n_q}$ which gives us equation (2).

Running Time of $\mathcal{S}$. The running time of algorithm $\mathcal{S}$ is that of running the forger $\mathcal{F}$ and a number of modular exponentiations with exponents belonging to $\mathbb{Z}_q$. Each query to the H-oracle requires one exponentiation. Each query to the signature oracle requires two exponentiations and two multi-exponentiations. Adding these values gives the running time in equation (1).

4 Security and Efficiency Analysis

Minimum required randomness size. Assume the *EDL* signature scheme is $(t, q_H, q_{sig}, \epsilon)$-broken by some algorithm $\mathcal{A}$. This adversary $\mathcal{A}$ can be run repeatedly, and the expected time to produce a forgery is t/ϵ. Let $n = \log t$ and $e = \log(1/\epsilon)$. Thus, the security parameter of the *EDL* scheme is $n_{ss} \leq \log(t/\epsilon) = n + e$.

A customary bound on the number of signatures that an instance of a signature scheme can generate is $q_{sig} \leq 2^{30}$. If we assume that evaluating a hash function is a unit operation, there can be at most $q_H \leq t = 2^n$ hash oracle queries. We also assume that an exponentiation in $G_{g,p}$ takes about 100 times longer than one hash query. Using these assumptions in Theorem 1, we obtain

$$t' \approx t + q_H \cdot \mathrm{C}_{\exp}(G_{g,p}) \approx 2^{n+7}$$

and

$$\epsilon' \approx \epsilon - (q_{sig} \cdot q_H \cdot 2^{-n_r} + q_H \cdot 2^{-n_q}) = 2^{-e} - (2^{n+30-n_r} + 2^{n-n_q}).$$

Because of the $O(\sqrt{q})$ security of discrete-log in $G_{g,p}$, we have $n_q > 2n$, and $2^{n-n_q} < 2^{-n_q/2}$ is negligible. Therefore, for ϵ' to be at least $\epsilon/2$, we need $-e > (n + 30 - n_r)$, and hence also require that $n_r > n_{ss} + 30$. Setting $n_r = n_{ss} + 31$ and assuming that computations that take time greater than 2^{80} are infeasible, we get $n_r = 111$.

Relative "tightness" and its implications. Note that the bound on the reduction time is $t' \approx t \cdot 2^7$ and $\epsilon' \approx \epsilon/2$, which means that our reduction carries a small factor of 2^8 decrease in security. Therefore, to get a provable 2^{80} hardness bound on our signature scheme in the random oracle model, we need a group $G_{g,p}$ with at least 2^{88} security of the CDH problem. Recall that other discrete-log based signatures encounter a $q_H = 2^{80}$ security degradation via the "forking lemma" reduction. Hence, to achieve the same 2^{80} hardness bound, they require a group $G_{g,p}$ with a 2^{160} security of the discrete-log problem. If we assume a widely held belief that the CDH and DL problems have comparable security, our scheme is as secure as the standard discrete-log based schemes over groups with half the security parameter for the DL or CDH problem.

Moreover, by the results of Maurer and Wolf [MW99], our scheme achieves better provable security *under the DL assumption alone* than traditional discrete-log based signatures (in a large class of groups). Maurer and Wolf show that for a large class of groups, there exist an efficient reduction from DL to CDH. Their reduction algorithm solves the DL problem by invoking a *perfect* CDH oracle $O((\log q)^5)$ times, where q is the (prime) size of the group. Since a forger against our signature scheme implements a perfect oracle, taking $\log q \approx 128$, the Maurer and Wolf reduction encounters a $(128)^5 = 2^{35}$ decrease in the security parameter. Combining the two reductions show that our scheme provably achieves (in the Random Oracle model) the 2^{80} bound in a group $G_{g,p}$ with a $2^{80+35+8} = 2^{123}$ security of the DL problem, provided that the DL to CDH reduction of Maurer and Wolf holds for this group.

In summary, we call our scheme *tightly* related to the CDH problem because the 2^8 security degradation is small, and *closely* related to the DL problem because the $2^{35+8} = 2^{43}$ degradation is medium. At the same time, we call other discrete-log based schemes *loosely* related to the DL problem because the 2^{80} degradation is large. Indeed, the terms *tight*, *close*, and *loose* are used only comparatively.

4.1 Security vs. Efficiency Trade-offs

Signature size increases or stays constant. Signatures produced by our scheme are tuples in $(G_q \times \{0,1\}^{111} \times \mathbb{Z}_q \times \mathbb{Z}_q)$. In other discrete-log schemes, like DSS or Schnorr, the signature is a pair of elements in $\mathbb{Z}_q$. In a traditional discrete-log setting over a field $\mathbb{Z}_p^*$, taking $|p| \approx 1000$ and $|q| \approx 160$, our signatures are 4 to 5 times longer, but with much better security guarantees. If we instead require security guarantees for traditional signature schemes that are comparable to EDL, then assuming CDH $\approx$ DL, traditional discrete-log signatures need $|q| = |G_{g,p}|$ values that are two times larger than *EDL* signatures. In this case, our signatures are 2 to 2.5 times longer.

The signature size comparison is more favorable for our scheme in elliptic curve systems where elements in $G_{g,p}$ have a representation of similar size to elements in $\mathbb{Z}_q$. Assuming that the two schemes work over the same curve, our signature is about twice the size of traditional discrete-log signatures. But if we require similar security guarantees based on the CDH $\approx$ DL assumption, our signatures become slightly smaller than the traditional discrete-log signatures implemented over elliptic curves.

Signature generation and verification time decrease. The computational costs in our scheme are three exponentiations for the signer and about two exponentiations for the verifier. Traditional signatures require one exponentiation for signing and verification. However, if we compare the signature schemes based on similar levels of security (assuming CDH $\approx$ DL), traditional signatures need to use larger groups. Recall that the cost of exponentiation in a field of q elements is proportional to $|q|$ times the multiplication cost. The best multiplication algorithms have a cost of at least $O(|p|^{1.6})$. Therefore, a factor of α increase in $|q|$ and a factor of β increase in $|p|$ results in a $\alpha \cdot \beta^{1.6}$ increase in the exponentiation cost.

In the traditional discrete-log setting of a prime field $\mathbb{Z}_p^*$, the index-calculus method of breaking the discrete-log problem works in time about $O(\exp(\sqrt[3]{|p|}))$. Hence, a factor of 2 increase in the security parameter implies $\beta = 2^3$ increase in the length of the modulus p. The size of the group $|q|$ increases by factor $\alpha = 2$ because of baby-step/giant-step algorithm which works in time $O(\exp(|q|/2))$. Therefore, under the CDH $\approx$ DL assumption, the exponentiation cost in a traditional signature scheme with security guarantees matching our scheme is $2 \cdot 8^{1.6} \approx 56$ times than in our scheme. Our signature operation is thus about $56/3 \approx 18$ times faster and the verification is $56/2 = 27$ times faster. For elliptic curve groups, to match the security guarantees of the new scheme, we need a $\alpha = \beta = 2$ factor increase in the size of the representation of the group. Hence, the exponentiation cost for traditional schemes on elliptic curves matching the security bounds of our scheme is $2 \cdot 2^{1.6} \approx 6$ times larger, which still makes our signer two times faster and the verifier three times faster.

5 Open Problems

We see several interesting open problems: (1) One question is to find a signature scheme that has a tight security reduction to CDH or DL, but whose signature size and signing and verification times are the same as those in the traditional signature schemes, even when the two schemes work in the same group. (2) Another question is to find a signature scheme that improves on the "forking lemma" reduction to the DL assumption for a larger class of groups than those of Maurer and Wolf [MW99]. (3) Another interesting question is to find a blind signature scheme with a tight security reduction to the CDH problem.

Acknowledgments

The authors thank Dan Boneh and the anonymous referees for their helpful comments. The authors were supported by a Packard Foundation Fellowship and NSF grant CCR-0205733.

References

[BGMW92] Ernest Brickell, Daniel Gordon, Kevin McCurley, and David Wilson. Fast exponentiation with precomputation. In R.A. Rueppel, editor, *Proceedings of Eurocrypt 1992*, volume 0658 of *LNCS*, pages 200–207. Springer-Verlag, May 1992.

[BGR98] Mihir Bellare, Juan Garay, and Tal Rabin. Fast batch verification for modular exponentiation and digital signatures. In K. Nyberg, editor, *Proceedings of Eurocrypt 1998*, volume 1403 of *LNCS*, pages 236–250. Springer-Verlag, May 1998.

[BL96] Dan Boneh and Richard Lipton. Algorithms for black-box fields and their application to cryptography. In Neal Koblitz, editor, *Proceedings of Crypto 1996*, volume 1109 of *LNCS*, pages 283–297. Springer-Verlag, May 1996.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In C. Boyd, editor, *Proceedings of Asiacrypt 2001*, volume 2248 of *LNCS*, pages 514–532. Springer-Verlag, December 2001.

[BPVY00] Ernest Brickell, David Pointcheval, Serge Vaudenay, and Moti Yung. Design validations for discrete logarithm based signature schemes. In Hideki Imai and Yuliang Zheng, editors, *Proceedings of PKC 2000*, volume 1751 of *LNCS*, pages 276–292. Springer-Verlag, January 2000.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures - How to sign with RSA and Rabin. In Ueli Maurer, editor, *Proceedings of Eurocrypt 1996*, volume 1070 of *LNCS*, pages 399–416. Springer-Verlag, May 1996.

[CEvdG87] David Chaum, Jan-Hendrik Evertse, and Jeroen van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In David Chaum and Wyn Price, editors, *Proceedings of Eurocrypt 1987*, volume 0304 of *LNCS*, pages 127–142. Springer-Verlag, May 1987.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. In *Proceedings of the 30th annual ACM symposium on Theory of Computing*, pages 209–218. ACM Press, 1998.

[CP92] David Chaum and Torben Pryds Pedersen. Wallet databases with observers. In Ernest Brickell, editor, *Proceedings of Crypto 1992*, volume 0740 of *LNCS*, pages 89–105. Springer-Verlag, August 1992.

[CS97] Jan Camenisch and Markus Stadler. Proof systems for general statements about discrete logarithms. Technical Report 260, Institute for Theoretical Computer Science, ETH Zürich, March 1997.

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

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976.

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

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew Odlyzko, editor, *Proceedings of Crypto 1986*, volume 0263 of *LNCS*, pages 186–194. Springer-Verlag, August 1986.

[GHR99] Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In Jacques Stern, editor, *Proceedings of Eurocrypt 1999*, volume 1592 of *LNCS*, pages 123–139. Springer-Verlag, May 1999.

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

[JS99] Markus Jakobsson and Claus-Peter Schnorr. Efficient oblivious proofs of correct exponentiation. In Bart Preneel, editor, *Proceedings of the IFIP Conference on Communications and Multimedia Security 1999*, volume 152, pages 71–86. Kluwer, September 1999.

[LV01] Arjen Lenstra and Eric Verheul. Selecting cryptographic key sizes. *Journal of Cryptology*, 14(4):255–293, 2001.

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

[MW99] Ueli Maurer and Stefan Wolf. The relationship between breaking the Diffie-Hellman protocol and computing discrete logarithms. *SIAM Journal on Computing*, 28(5):1689–1721, 1999.

[NIS94] NIST. Digital Signature Standard (DSS). Publication 196, Federal Information Processing Standards, November 1994.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the 21st annual ACM symposium on Theory of Computing*, pages 33–43. ACM Press, 1989.

[OO98] Kazuo Ohta and Tatsuaki Okamoto. On concrete security treatment of signatures derived from identification. In Hugo Krawczyk, editor, *Proceedings of Crypto 1998*, volume 1462 of *LNCS*, pages 354–369. Springer-Verlag, August 1998.

[OP01] Tatsuaki Okamoto and David Pointcheval. The Gap-Problems: A new class of problems for the security of cryptographic schemes. In Kwangjo Kim, editor, *Proceedings of PKC 2001*, volume 1992 of *LNCS*, pages 104–118. Springer-Verlag, February 2001.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli Maurer, editor, *Proceedings of Eurocrypt 1996*, volume 1070 of *LNCS*, pages 387–398. Springer-Verlag, May 1996.

[Rom90] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd annual ACM symposium on Theory of Computing*, pages 387–394. ACM Press, 1990.

[Sch89] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In G. Brassard, editor, *Proceedings of Crypto 1989*, volume 0435 of *LNCS*, pages 239–252. Springer-Verlag, August 1989.

[SG98] Victor Shoup and Rosario Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In Kaisa Nyberg, editor, *Proceedings of Eurocrypt 1998*, volume 1403 of *LNCS*, pages 1–16. Springer-Verlag, May 1998.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Proceedings of Eurocrypt 1997*, volume 1233 of *LNCS*, pages 256–266. Springer-Verlag, May 1997.

[ST01] Adi Shamir and Yael Tauman. Improved online/offline signature schemes. In Joe Killian, editor, *Proceedings of Crypto 2001*, volume 2139 of *LNCS*, pages 355–367. Springer-Verlag, August 2001.