Boneh-Boyen signatures and the Strong Diffie-Hellman problem

David Jao and Kayo Yoshida*

Department of Combinatorics and Optimization
University of Waterloo, Waterloo ON, N2L 3G1, Canada
{djao,k2yoshid}@ecc.math.uwaterloo.ca

Abstract. The Boneh-Boyen signature scheme is a pairing based short signature scheme which is provably secure in the standard model under the q-Strong Diffie-Hellman assumption. In this paper, we prove the converse of this statement, and show that forging Boneh-Boyen signatures is actually equivalent to solving the q-Strong Diffie-Hellman problem. Using this equivalence, we exhibit an algorithm which, on the vast majority of pairing-friendly curves, recovers Boneh-Boyen private keys in $O(p^{\frac{2}{5}+\varepsilon})$ time, using $O(p^{\frac{1}{5}+\varepsilon})$ signature queries. We present implementation results comparing the performance of our algorithm and traditional discrete logarithm algorithms such as Pollard's lambda algorithm and Pollard's rho algorithm. We also discuss some possible countermeasures and strategies for mitigating the impact of these findings.

1 Introduction

The q-SDH assumption was proposed by Boneh and Boyen [5, 6] as a tool to assist in the security analysis of the Boneh-Boyen short signature scheme. Versions of this assumption are also used in Mitsunari et al. [21], Dodis and Yampolskiy [13], and in the Boneh-Boyen IBE scheme [4]. The survey article of Boyen [7] lists the q-SDH assumption as one of the first in a family of new assumptions that have appeared in the context of pairing-based cryptography, and the first of these to be analyzed in the generic group model.

Prior to this work, no equivalence was known between the security of the q-SDH assumption and the security of the Boneh-Boyen signature scheme. Boneh and Boyen [5, 6] provide a security reduction with a running time of $\Theta(q^2)$, but it only goes in one direction: namely, if the q-SDH assumption holds, then Boneh-Boyen signatures are unforgeable. There are two reasons why one might desire to prove the converse result. One reason is practical: Brown and Gallant [9] and Cheon [11] have shown that, in groups of size p, the q-SDH problem can be solved in $O(\sqrt{p/d}+\sqrt{d})$ exponentiations, instead of the $O(\sqrt{p})$ operations required for discrete log, for any divisor $d \leq q$ of $p-1$ (a similar result holds for $p+1$). Knowing that q-SDH and Boneh-Boyen are equivalent thus allows one to forge Boneh-Boyen signatures in faster than square root time; in our case this is possible via a known or chosen message attack. Although the resulting algorithm remains exponential, a lower exponent is still interesting in the context of a short signature scheme, especially for extremely short signature lengths at the lower margins of security. A further motivation for proving equivalence is given by Koblitz and Menezes [16, 17]. They argue that an equivalence result is preferable from a philosophical standpoint, since researchers have more incentive to solve the underlying hard problem (that is, q-SDH) if such solutions lead immediately to cryptanalysis of a concrete scheme.

In this paper, we present an algorithm for performing existential forgeries of Boneh-Boyen signatures using a q-SDH oracle, whose running time is also $\Theta(q^2)$. This shows that the security of Boneh-Boyen **cannot** be proved under **any weaker assumption** than SDH; in other words, the security of the Boneh-Boyen scheme is **equivalent** to the intractability of the q-SDH problem. Our reduction holds for both the "basic" and "full" versions of the Boneh-Boyen scheme. Together with the algorithms of Brown and Gallant and Cheon, our result allows a total break (i.e. recovery of the private key) of the full (resp., basic) Boneh-Boyen scheme in time $O(p^{\frac{2}{5}+\varepsilon})$, under a chosen (resp., known) message attack, whenever $p \pm 1$ has a divisor of appropriate size (which in practice is almost always the case; see Section 6.3). This running time is slightly higher than the generic group bound of $\Omega(p^{\frac{1}{3}})$ given by Boneh and Boyen [5, 6], because

* The authors were partially supported by NSERC.

of the quadratic runtime of our reduction. Nevertheless, it represents a significant improvement over the $O(p^{\frac{1}{2}+\varepsilon})$ time required to calculate discrete logarithms.

The techniques we use are not entirely new, although we did discover them independently. A simplified version of Proposition 4.1 appears in Mitsunari et al. [21], a paper which is cited by Boneh and Boyen [5, 6] and Cheon [11]. However, we are quite confident that our overall result is new. For example, Cheon [11] applies his results to the cryptanalysis of several different cryptosystems, but omits the Boneh-Boyen scheme from such consideration, indicating that no such cryptanalysis was available. In addition, the survey article of Boyen [7] asserts that the MSDH assumption (which amounts to forging Boneh-Boyen signatures) is "an actually weaker statement" than q-SDH. This sentence implies that no equivalence between Boneh-Boyen and q-SDH was known at the time of that writing.

We note here that the abovementioned generic group analysis already yields a bound of $\Omega(p^{\frac{1}{3}})$ on the security of the q-SDH assumption for large q. Thus, it would be reasonable for a conservative adopter to view the Boneh-Boyen scheme as having cube root security under large scale chosen message attacks, even in the absence of any concrete algorithm that runs faster than discrete log. However, an explicit result showing that forging signatures reduces to the q-SDH problem is still useful, precisely because such a reduction yields concrete algorithms for forging signatures, and hence helps to validate the conservative viewpoint.

1.1 Organization of the paper

The rest of this paper is organized as follows. Section 2 contains background material such as security definitions, bilinear pairings, and the q-SDH and related problems. Section 3 presents the basic and full versions of the Boneh-Boyen short signature scheme [5, 6]. In Section 4, we give a security analysis of the signature scheme, and show how to forge Boneh-Boyen signatures using a q-SDH oracle. In Section 5 we review the algorithms of Brown and Gallant [9] and Cheon [11] for solving the q-SDH problem, and describe how their algorithms can be used to compute the private key in the Boneh-Boyen scheme. Section 6 contains theoretical and experimental runtime figures showing that a Boneh-Boyen private key can be computed in $O(p^{\frac{2}{5}+\varepsilon})$ time, given access to a signing oracle. We conclude with an analysis of the proportion of curves for which a divisor of the suitable form exists, together with a list of related open problems.

2 Preliminaries

2.1 Security definitions

We begin by reviewing the two security definitions used in the proof of security for the Boneh-Boyen signature scheme [5, 6].

Strong Existential Unforgeability. Strong existential unforgeability is defined via the following game between a challenger and an adversary $\mathcal{A}$.

1. The challenger generates a key pair $(\mathrm{PK}, \mathrm{SK})$ and gives PK to the adversary.
2. The adversary $\mathcal{A}$ can adaptively make up to q_S queries for signatures of messages $m_1, \ldots, m_{q_S}$ of its choice. The challenger must respond to the queries with valid signatures $\sigma_1, \ldots, \sigma_{q_S}$ of the messages $m_1, \ldots, m_{q_S}$.
3. Eventually, the adversary $\mathcal{A}$ outputs a pair (m_*, σ_*), and wins the game if $(m_*, \sigma_*) \neq (m_i, \sigma_i)$ for $i = 1, \ldots, q_S$ and $\mathrm{Verify}(m_*, \sigma_*, \mathrm{PK}) = \texttt{true}$.

The adversary $\mathcal{A}$'s advantage, denoted $\mathrm{Adv}\,\mathrm{Sig}(\mathcal{A})$ is defined as the probability that $\mathcal{A}$ wins the above game, where the probability is taken over the coin tosses made by $\mathcal{A}$ and the challenger.

Definition 2.1. *An adversary $\mathcal{A}$ is said to (t, q_S, ϵ)-break a signature scheme if $\mathcal{A}$ runs in time at most t, makes at most q_S signature queries, and $\mathrm{Adv}\,\mathrm{Sig}(\mathcal{A}) \geq \epsilon$. We say that a signature scheme is (t, q_S, ϵ)-existentially unforgeable under an adaptive chosen message attack if there is no adversary that (t, q_S, ϵ)-breaks it.*

Weak Existential Unforgeability. Weak existential unforgeability is defined via the following game between a challenger and an adversary $\mathcal{A}$.

1. The challenger generates a key pair $(\mathrm{PK}, \mathrm{SK})$.
2. The adversary $\mathcal{A}$ chooses up to q_S messages $m_1, \ldots, m_{q_S}$ and sends them to the challenger.
3. The challenger gives $\mathcal{A}$ the public key PK and valid signatures $\sigma_1, \ldots, \sigma_{q_S}$ for the messages $m_1, \ldots, m_{q_S}$.
4. Eventually, the adversary $\mathcal{A}$ outputs a pair (m_*, σ_*), and wins the game if $m_* \neq m_i$ for $i = 1, \ldots, q_S$ and $\mathrm{Verify}(m_*, \sigma_*, \mathrm{PK}) = \texttt{true}$.

The adversary $\mathcal{A}$'s advantage, denoted $\mathrm{Adv}\,\mathrm{Sig}\,\mathrm{W}(\mathcal{A})$, is defined as the probability that $\mathcal{A}$ wins the above game, where the probability is taken over the coin tosses made by $\mathcal{A}$ and the challenger.

Definition 2.2. *An adversary $\mathcal{A}$ is said to (t, q_S, ϵ)-weakly break a signature scheme if $\mathcal{A}$ runs in time at most t, makes at most q_S signature queries, and $\mathrm{Adv}\,\mathrm{Sig}\,\mathrm{W}(\mathcal{A}) \geq \epsilon$. We say that a signature scheme is (t, q_S, ϵ)-existentially unforgeable under a weak chosen message attack if there is no adversary that (t, q_S, ϵ)-weakly breaks it.*

2.2 Bilinear pairings

The Boneh-Boyen short signature scheme makes use of bilinear pairings. Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be cyclic groups of prime order $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T| = p$. The operations in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are written multiplicatively. Recall that a function $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is called a bilinear pairing if it satisfies the following conditions:

– **Bilinearity:** For any $u_1, u_2, u \in \mathbb{G}_1$ and $v_1, v_2, v \in \mathbb{G}_2$,

$$e(u_1 u_2,\ v) = e(u_1,\ v) \cdot e(u_2,\ v), \quad \text{and}$$
$$e(u,\ v_1 v_2) = e(u,\ v_1) \cdot e(u,\ v_2).$$

– **Non-degeneracy:** There exists $u \in \mathbb{G}_1$ and $v \in \mathbb{G}_2$ such that $e(u,\ v) \neq 1$.

We assume that the pairing function and the group operations are efficiently computable. The pair $(\mathbb{G}_1, \mathbb{G}_2)$ is called a bilinear group pair.

2.3 SDH and related problems

The q-SDH problem and its variants provide the underlying basis for security in several pairing-based protocols [4–7, 13, 21]. Throughout this section, let $(\mathbb{G}_1, \mathbb{G}_2)$ be a bilinear group pair of prime order p, and let g_1 and g_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.

q-SDH Problem. In the full version of the Boneh-Boyen paper [6], the q-Strong Diffie-Hellman (q-SDH) problem on the bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$ is defined as follows:

> Given a $(q+3)$-tuple $(g_1, g_1^x, \ldots, g_1^{x^q}, g_2, g_2^x) \in \mathbb{G}_1^{q+1} \times \mathbb{G}_2^2$ as input, output $(c, g_1^{\frac{1}{x+c}})$ for some $c \in \mathbb{Z}_p$ such that $x + c \not\equiv 0 \pmod{p}$.

The advantage $\mathrm{Adv}\, q\text{-SDH}(\mathcal{A})$ of an algorithm $\mathcal{A}$ in solving the q-SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$ is defined as

$$\mathrm{Adv}\, q\text{-SDH}(\mathcal{A}) = \Pr\left[\mathcal{A}(g_1, g_1^x, \ldots, g_1^{x^q}, g_2, g_2^x) = (c, g_1^{\frac{1}{x+c}})\right],$$

where the probability is taken over the random choices of generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, the random choice of $x \in \mathbb{Z}_p^*$, and the coin tosses made by $\mathcal{A}$.

Definition 2.3. *An algorithm $\mathcal{A}$ is said to (t, ϵ)-break the q-SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$ if $\mathcal{A}$ runs in time t and $\mathrm{Adv}\, q\text{-SDH}(\mathcal{A}) \geq \epsilon$. We say that the (q, t, ϵ)-SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$ if there is no algorithm that (t, ϵ)-breaks the q-SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$.*

The definition of the q-SDH problem given in the original version of the Boneh-Boyen paper [5] is slightly different. The original version uses a $(q+2)$-tuple $(g_1, g_2, g_2^x, \ldots, g_2^{x^q})$ as input rather than $(g_1, g_1^x, \ldots, g_1^{x^q}, g_2, g_2^x)$, and it also assumes an efficiently computable isomorphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$ is available. In this paper, we adopt the definition given in the full version of the Boneh-Boyen paper.

Related Problems. A notable variation of the q-SDH problem for our purposes is the MSDH problem [7, 8]. The Modified q-SDH or q-MSDH problem on a group $\mathbb{G}$ is the following computational problem: given $g, g^x \in \mathbb{G}$, and a $(q-1)$-tuple $(c_1, g^{\frac{1}{x+c_1}}), \ldots, (c_{q-1}, g^{\frac{1}{x+c_{q-1}}})$ where each $c_i \in \mathbb{Z}_p$, output $(c, g^{\frac{1}{x+c}})$ for some $c \in \mathbb{Z}_p \setminus \{c_1, \ldots, c_{q-1}\}$. Over a group equipped with a type 1 pairing [15], solving the q-MSDH problem is equivalent to existential forgery of the Boneh-Boyen basic signature scheme under a known message attack using q signature queries. Boyen remarks in [7] that the MSDH assumption is weaker than SDH. Our results, however, imply that in groups with a type 1 pairing the q-MSDH problem is equivalent to the q-SDH problem via a $\Theta(q^2)$ reduction.

3 Boneh-Boyen signature scheme

Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be cyclic groups of prime order p, and let $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear pairing. In [5, 6], Boneh and Boyen present two versions of their signature schemes, a basic scheme and a full scheme, with the former being used to prove the security of the latter. The protocols in the original version [5] of their paper are slightly different from those in the full version [6]. Here we use only the schemes from the full version of the paper [6].

The Basic Signature Scheme

- **Key generation:** KeyGen outputs random generators g_1 and g_2 of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and a random integer $x \in \mathbb{Z}_p^*$. Let $\zeta \leftarrow e(g_1, g_2) \in \mathbb{G}_T$. The public key is $\mathrm{PK} = (g_1, g_2, g_2^x, \zeta)$, and the private key is $\mathrm{SK} = (g_1, x)$.
- **Signing:** Given a message $m \in \mathbb{Z}_p$ and a private key SK, $\mathrm{Sign}(m, \mathrm{SK})$ outputs a signature $\sigma \leftarrow g_1^{\frac{1}{x+m}}$, where the exponent is calculated modulo p. In the unlikely event that $x + m \equiv 0 \pmod{p}$, $\mathrm{Sign}(m, \mathrm{SK})$ outputs $\sigma \leftarrow 1$.
- **Verification:** $\mathrm{Verify}(m, \sigma, \mathrm{PK}) = \texttt{true}$ if and only if $e(\sigma,\ g_2^x \cdot g_2^m) = \zeta$.

The Full Signature Scheme

- **Key generation:** KeyGen outputs random generators g_1 and g_2 of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and random integers $x, y \in \mathbb{Z}_p^*$. Let $\zeta \leftarrow e(g_1, g_2) \in \mathbb{G}_T$. The public key is $\mathrm{PK} = (g_1, g_2, g_2^x, g_2^y, \zeta)$, and the private key is $\mathrm{SK} = (g_1, x, y)$.
- **Signing:** Given a message $m \in \mathbb{Z}_p$ and a private key SK, $\mathrm{Sign}(m, \mathrm{SK})$ randomly picks $r \in \mathbb{Z}_p$ such that $x + m + yr \not\equiv 0 \pmod{p}$, and calculates $\sigma \leftarrow g_1^{\frac{1}{x+m+yr}}$. The signature is (σ, r).
- **Verification:** $\mathrm{Verify}(m, (\sigma, r), \mathrm{PK}) = \texttt{true}$ if and only if $e(\sigma,\ g_2^x \cdot g_2^m \cdot (g_2^y)^r) = \zeta$.

The element g_1 can be omitted from the public key with no loss of functionality. None of our proofs use g_1, except for the proof of Theorem 4.3, and even this theorem can be modified to hold without g_1 (see remarks at the end of the proof of Theorem 4.3).

4 Security analysis of the Boneh-Boyen signature scheme

We present our equivalence results in this section. We begin with a partial fraction decomposition which refines and generalizes a formula given in [21].

Proposition 4.1 *Let $\mathbb{F}$ be a field, and $x \in \mathbb{F}$. Let $d, k \in \mathbb{Z}$ be such that $d \geq 1$, $k \geq 0$. Let m_i for $i = 1, \ldots, d$ be distinct elements of $\mathbb{F}$ such that $x + m_i \neq 0$. Then,*

$$\frac{x^k}{\prod_{i=1}^{d}(x+m_i)} = \begin{cases} \displaystyle\sum_{i=1}^{d} \frac{(-m_i)^k}{(x+m_i)\prod_{j\neq i}(m_j - m_i)} & \text{for } 0 \leq k < d \\ \displaystyle 1 + \sum_{i=1}^{d} \frac{(-m_i)^d}{(x+m_i)\prod_{j\neq i}(m_j - m_i)} & \text{for } k = d \\ \displaystyle x + \sum_{i=1}^{d} \left[-m_i + \frac{(-m_i)^{d+1}}{(x+m_i)\prod_{j\neq i}(m_j - m_i)} \right] & \text{for } k = d+1 \end{cases}$$

Proof. By the principle of permanence of identity [1, p. 456], it suffices to prove that the equations hold when $\mathbb{F} = \mathbb{C}$, since they then form an algebraic identity. Thus, we let

$$f(x) = \frac{x^k}{(x+m_1)\cdots(x+m_d)},$$

and treat $f(x)$ as a complex function in x. We can write $f(x)$ as a partial fraction of the form

$$f(x) \;=\; a_k x + b_k + \frac{c_1}{x+m_1} + \frac{c_2}{x+m_2} + \cdots + \frac{c_d}{x+m_d}$$

where

$$a_k = \begin{cases} 1 & \text{if } k = d+1, \\ 0 & \text{otherwise,} \end{cases} \qquad b_k = \begin{cases} -\sum_{i=1}^{d} m_i & \text{if } k = d+1, \\ 1 & \text{if } k = d, \text{ and} \\ 0 & \text{otherwise,} \end{cases}$$

and each c_i is a constant. By symmetry, we only need to prove $c_1 = \frac{(-m_1)^k}{\prod_{j\neq 1}(m_j - m_1)}$.

Note that $f(x) - \frac{c_1}{x+m_1} = a_k x + b_k + \frac{c_2}{x+m_2} + \cdots + \frac{c_d}{x+m_d}$ has an analytic Taylor series expansion about $x = -m_1$. Thus c_1 is the residue of f at the simple pole $x = -m_1$. If we write $f(x) = \frac{\phi(x)}{x+m_1}$ where $\phi(x) = \frac{x^k}{(x+m_2)\cdots(x+m_d)}$, then $\phi(x)$ is analytic and nonzero at $x = -m_1$. A standard theorem in complex analysis (see [10, p. 234] or [2, p. 115]) gives

$$c_1 = \phi(-m_1) = \frac{(-m_1)^k}{\prod_{j\neq 1}(m_j - m_1)}$$

as desired.

Corollary 4.2 *Let $\mathbb{G}$ be a cyclic group of order p, let $g \in \mathbb{G}$ be a generator, and let $x \in \mathbb{Z}_p$. Let m_i for $i = 1, \ldots, d$ be distinct elements of $\mathbb{Z}_p$ such that $x + m_i \not\equiv 0 \pmod{p}$. Then,*

$$g^{\frac{x^k}{\prod_{i=1}^{d}(x+m_i)}} = \begin{cases} \displaystyle\prod_{i=1}^{d} g^{\frac{(-m_i)^k}{(x+m_i)\prod_{j\neq i}(m_j - m_i)}} & \text{for } \; 0 \leq k \leq d-1 \\ \displaystyle g \cdot \prod_{i=1}^{d} g^{\frac{(-m_i)^d}{(x+m_i)\prod_{j\neq i}(m_j - m_i)}} & \text{for } \; k = d \\ \displaystyle g^x \cdot g^{-\sum_{i=1}^{d} m_i} \cdot \prod_{i=1}^{d} g^{\frac{(-m_i)^{d+1}}{(x+m_i)\prod_{j\neq i}(m_j - m_i)}} & \text{for } \; k = d+1 \end{cases}$$

Assume that all the values m_i and $g^{\frac{1}{x+m_i}}$ are known. Furthermore, assume for $k = d$ and $k = d+1$ that g is known, and for $k = d+1$ that g^x is known. Then calculating $g^{\frac{x^k}{\prod_{i=1}^{d}(x+m_i)}}$ for a single value of k takes $\Theta(dT + d^2T_p)$ time, where T is the maximum time needed for a single exponentiation in $\mathbb{G}$, and T_p is the maximum time needed for one operation in $\mathbb{Z}_p$. Calculating all of $g^{\frac{1}{\prod_{i=1}^{d}(x+m_i)}}, g^{\frac{x}{\prod_{i=1}^{d}(x+m_i)}}, \ldots, g^{\frac{x^{d+1}}{\prod_{i=1}^{d}(x+m_i)}}$ takes $\Theta(d^2T)$ time.

Proof. The proof of this Corollary is straightforward from Proposition 4.1.

4.1 Security of the basic signature scheme

In this section, we analyze the security of the basic Boneh-Boyen signature scheme. We show that existential forgery of the basic scheme under a weak chosen message attack (indeed, under a known message attack) reduces to the q-SDH problem. This result is the converse of [6, Lemma 9], and it also illustrates the main idea behind the corresponding result for the full scheme (Theorem 4.4).

Theorem 4.3. *If there is an algorithm that (t', ϵ')-breaks the q-SDH problem, then we can (t, q_S, ϵ)-weakly break the Boneh-Boyen basic signature scheme provided that*

$$t \geq t' + \Theta(q^2 T), \quad q_S \geq q, \quad \text{and} \quad \epsilon \leq \frac{p-1-q}{p-1}\,\epsilon',$$

where T is the maximum time needed for one exponentiation in $\mathbb{G}_1$.

Proof. Let $\mathcal{A}$ be an algorithm that (t', ϵ')-breaks the q-SDH problem. We show that an adversary $\mathcal{B}$ can perform existential forgeries of the basic signature scheme under a weak chosen message attack. In fact, it turns out that a list of valid message-signature pairs suffices. Accordingly, the adversary $\mathcal{B}$ receives a public key (g_1, g_2, g_2^x, ζ) and a list of distinct messages $m_1, \ldots, m_{q_S}$ together with their valid signatures $(\sigma_1, \ldots, \sigma_{q_S}) = (g_1^{1/(x+m_1)}, \ldots, g_1^{1/(x+m_{q_S})})$, where $q_S \geq q$.

Let $h_k \leftarrow g_1^{\frac{x^k}{(x+m_1)\cdots(x+m_q)}}$ for each $k = 0, \ldots, q$. The adversary $\mathcal{B}$ calculates $(h_0, h_1, \ldots, h_q)$ using Corollary 4.2, and runs the algorithm $\mathcal{A}$ on the input $(h_0, h_1, \ldots, h_q, g_2, g_2^x)$. With probability ϵ', $\mathcal{A}$ returns $(m_*, g_1^{\frac{1}{(x+m_1)\cdots(x+m_q)(x+m_*)}})$ for some $m_* \in \mathbb{Z}_p$.

We claim that m_* is not equal to any of the m_i except with negligible probability. To show this, observe that g_1 is not disclosed to $\mathcal{A}$ and that $g_1 = h_k^{\frac{(x+m_1)\cdots(x+m_q)}{x^k}}$ for all $k = 0, \ldots, q$. Thus, from the point of view of $\mathcal{A}$, any combination of $m_1, \ldots, m_q$ is equally likely to give rise to a fixed input $(h_0, h_1, \ldots, h_q)$. That is, $\mathcal{A}$ has no better than random chance of choosing an m_* which coincides with one of $m_1, \ldots, m_q$. Therefore, $m_* \neq m_i$ for all $i = 1, \ldots, q$ with probability at least $\frac{p-1-q}{p-1}$. If $m_* = m_i$ for some $1 \leq i \leq q$, then $\mathcal{B}$ aborts. Otherwise, by Proposition 4.1,

$$\frac{1}{(x+m_1)\cdots(x+m_q)(x+m_*)} = \frac{1}{(x+m_*)\prod_{j=1}^{q}(m_j - m_*)} + \sum_{i=1}^{q} \frac{1}{(x+m_i)\prod_{j\neq i}(m_j - m_i)}.$$

Using this equation, $\mathcal{B}$ can calculate $\sigma_* = g_1^{\frac{1}{x+m_*}}$ as follows:

$$\sigma_* \leftarrow \left[g_1^{\frac{1}{(x+m_1)\cdots(x+m_q)(x+m_*)}} \Big/ \prod_{i=1}^{q} (\sigma_i)^{\prod_{j\neq i} \frac{1}{m_j - m_i}} \right]^{\prod_{j=1}^{q}(m_j - m_*)} = g_1^{\frac{1}{x+m_*}}.$$

In this way $\mathcal{B}$ outputs (m_*, σ_*) which is a forgery for the basic signature scheme.

The bounds for ϵ and q_S are obvious from the above construction. The running time is bounded by the calculation of $g_1^{\frac{1}{(x+m_1)\cdots(x+m_q)}}$, $g_1^{\frac{x}{(x+m_1)\cdots(x+m_q)}}$, $\ldots$, $g_1^{\frac{x^q}{(x+m_1)\cdots(x+m_q)}}$, which takes $\Theta(q^2T)$ time by Corollary 4.2, and the query of $\mathcal{A}$, which takes time t'.

The above proof requires knowledge of the element g_1. If g_1 is not part of the public key, Theorem 4.3 remains valid, provided that q is replaced by $q+1$ in the inequalities. In this case $\mathcal{B}$ uses $q+1$ signature queries, and calculates $h'_k \leftarrow g_1^{\frac{x^k}{(x+m_1)\cdots(x+m_{q+1})}}$ for $k = 0, \ldots, q$, in place of $h_0, \ldots, h_q$.

4.2 Security of the full signature scheme

We now show that strong existential forgery of the full Boneh-Boyen signature scheme under chosen message attack reduces to the q-SDH problem. This result is the converse of [6, Theorem 8].

Theorem 4.4. *If there is an algorithm that (t', ϵ')-breaks the q-SDH problem, then we can (t, q_S, ϵ)-break the Boneh-Boyen full signature scheme provided that*

$$t \geq t' + \Theta(q_S^2 T), \quad q_S \geq q+1, \qquad \textit{and} \qquad \epsilon \leq \frac{(p-2-q)\left(p-1-\left(q^2+q\right)/2\right)}{(p-1)^2}\,\epsilon',$$

where T is the maximum time needed for one exponentiation in $\mathbb{G}_1$.

Proof. Let $\mathcal{A}$ be an algorithm that (t', ϵ')-breaks the q-SDH problem. Using $\mathcal{A}$, we show that an adversary $\mathcal{B}$ can perform existential forgeries for the full signature scheme under a chosen message attack.

First, $\mathcal{B}$ receives the public key $(g_1, g_2, g_2^x, g_2^y, \zeta)$ from the challenger. Next, $\mathcal{B}$ randomly selects a message $m_* \in \mathbb{Z}_p$, and queries the challenger for q_S different signatures of m_*. Each time the challenger receives m_*, it sends back a valid signature $(\sigma_i, r_i) = (g_1^{1/(x+m_*+yr_i)}, r_i)$ to $\mathcal{B}$, where r_i is chosen at random so that $x + m_* + yr_i \not\equiv 0 \bmod p$. In this way, $\mathcal{B}$ obtains q_S valid (and hopefully distinct) signatures $(\sigma_1, r_1), \ldots, (\sigma_{q_S}, r_{q_S})$ of the message m_*. If $\{r_1, \ldots, r_{q_S}\}$ does not contain $q+1$ distinct elements of $\mathbb{Z}_p$, then $\mathcal{B}$ aborts. Otherwise, let $h \leftarrow g_1^{1/y}$ and $z \leftarrow \frac{x+m_*}{y}$. Without loss of generality (reindexing if necessary), assume $r_1, r_2, \ldots, r_{q+1}$ are distinct. Then, for each $i = 1, \ldots, q+1$, we have

$$\sigma_i = g_1^{\frac{1}{x+m_*+yr_i}} = \left(g_1^{\frac{1}{y}}\right)^{\frac{1}{\frac{x+m_*}{y}+r_i}} = h^{\frac{1}{z+r_i}}.$$

Hence, for each $k = 1, \ldots, q$, the adversary $\mathcal{B}$ can calculate

$$h^{\frac{z^k}{(z+r_1)\cdots(z+r_{q+1})}} = \prod_{i=1}^{q+1} \sigma_i^{\frac{(-r_i)^k}{\prod_{j\neq i}(r_j-r_i)}}$$

using Corollary 4.2, since $\mathcal{B}$ knows each σ_i and each r_i. Also note that if we let $g_2' \leftarrow g_2^y$, then $g_2^x g_2^{m_*} = {g_2'}^z$. When $\mathcal{B}$ runs the algorithm $\mathcal{A}$ on the input $(h^{\frac{1}{(z+r_1)\cdots(z+r_{q+1})}}, h^{\frac{z}{(z+r_1)\cdots(z+r_{q+1})}}, \ldots, h^{\frac{z^q}{(z+r_1)\cdots(z+r_{q+1})}}, g_2', {g_2'}^z)$, the algorithm $\mathcal{A}$ returns $(r_*, h^{\frac{1}{(z+r_1)\cdots(z+r_{q+1})(z+r_*)}})$ for some $r_* \in \mathbb{Z}_p$ with probability ϵ'. If $r_* = r_i$ for some $1 \leq i \leq q+1$, then $\mathcal{B}$ aborts, but this event occurs with only negligible probability, by the same argument as in Theorem 4.3. Otherwise, by Proposition 4.1,

$$\frac{1}{(z+r_1)\cdots(z+r_{q+1})(z+r_*)} = \frac{1}{(z+r_*)\prod_{j=1}^{q+1}(r_j-r_*)} + \sum_{i=1}^{q+1} \frac{1}{(z+r_i)\prod_{j\neq i}(r_j-r_i)}$$

and thus $\mathcal{B}$ can calculate

$$\sigma_* \leftarrow \left[h^{\frac{1}{(z+r_1)\cdots(z+r_{q+1})(z+r_*)}} \Big/ \prod_{i=1}^{q+1} (\sigma_i)^{\prod_{j\neq i}\frac{1}{r_j-r_i}}\right]^{\prod_{j=1}^{q+1}(r_j-r_*)} = h^{\frac{1}{z+r_*}} = g_1^{\frac{1}{x+m_*+yr_*}}$$

In this way $\mathcal{B}$ outputs $(m_*, (\sigma_*, r_*))$ which, as indicated below, is with high probability an existential forgery for the full signature scheme.

The bound for q_S is obvious from the above construction. The running time is determined by the time needed to calculate $h^{\frac{1}{(z+r_1)\cdots(z+r_{q+1})}}$, $h^{\frac{z}{(z+r_1)\cdots(z+r_{q+1})}}$, $\ldots$, $h^{\frac{z^q}{(z+r_1)\cdots(z+r_{q+1})}}$, which is $\Theta(q^2T)$ by Corollary 4.2, and the query of $\mathcal{A}$, which takes time t'.

The probability that $\mathcal{B}$ succeeds is $P_1P_2\epsilon'$ where P_1 is the probability that the sequence of random elements $\{r_1, \ldots, r_{qS}\}$ chosen by the signing oracle comprises at least $q+1$ distinct elements, and P_2 is the probability that the r_* returned by $\mathcal{A}$ differs from the $q+1$ values r_i used by $\mathcal{B}$. We know that $P_2 \geq \frac{p-2-q}{p-1}$ using the argument from the proof of Theorem 4.3. Moreover, $P_1 \geq 1 - Q$ where Q is the probability that among the original $r_1, \ldots, r_{q+1}$ there exist $1 \leq i < j \leq q+1$ such that $r_i = r_j$. We have

$$Q \leq \sum_{j=2}^{q+1} \Pr\left(\exists i < j \text{ such that } r_i = r_j\right) \leq \sum_{j=2}^{q+1} \frac{j-1}{p-1} = \frac{q(q+1)}{2(p-1)}$$

so $P_1 \geq 1 - Q \geq \frac{p-1-q(q+1)/2}{p-1}$, which yields the bound for ϵ.

5 Cheon's algorithms

In [11], Cheon presents an algorithm which in certain cases computes the secret exponent x from the input of an instance of the q-SDH problem. Portions of this algorithm were also independently discovered by Brown and Gallant [9] in the context of a different problem. In what follows, we refer to this algorithm as Cheon's algorithm. Specifically, Cheon proves the following:

Theorem 5.1. *Let $\mathbb{G}$ be a cyclic group of prime order p and let g be a generator. Let T denote the maximum time needed for one exponentiation in $\mathbb{G}$.*

1. *Let d divide $p-1$. Given the group elements g, g^x, and g^{x^d}, the value of x can be recovered in time $O((\sqrt{p/d} + \sqrt{d})T)$.*
2. *Let d divide $p+1$. Given the group elements $g, g^x, g^{x^2}, \ldots, g^{x^{2d}}$, the value of x can be recovered in time $O((\sqrt{p/d} + d)T)$.*

Note that, if $q \geq d$ in the first case or $q \geq 2d$ in the second, then the algorithm in the theorem can solve the q-SDH problem; in fact, such an algorithm will reveal the secret exponent x. We show in this section that the algorithm can be applied to find the secret exponent in the Boneh-Boyen signature scheme over a bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$.

Theorem 5.2. *(Basic scheme) Let T and T_p denote the maximum time needed to perform one group exponentiation in $\mathbb{G}_1$ and one modular multiplication mod p, respectively.*

1. *Let d divide $p-1$. Given $d+1$ valid message-signature pairs, the private key x in the basic Boneh-Boyen signature scheme can be computed in time $O((\sqrt{p/d} + d)T + d^2T_p)$.*
2. *Let d divide $p+1$. Given $2d+1$ valid message-signature pairs, the private key x in the basic Boneh-Boyen signature scheme can be computed in time $O((\sqrt{p/d} + d^2)T)$.*

If g_1 is included in the public key, then d and $2d$ message-signature pairs are sufficient for the above two parts respectively.

Theorem 5.3. *(Full scheme) Let T and T_p be as in Theorem 5.2.*

1. *Let d divide $p-1$. Then the private key pair (x, y) of the full Boneh-Boyen signature scheme can be computed under a chosen message attack, using $2d+2$ signature queries, in time $O((\sqrt{p/d}+d)T+d^2T_p)$, with probability at least $\left(\frac{p-1-d(d+1)/2}{p-1}\right)^2$.*

2. *Let d divide $p+1$. Then the private key pair (x,y) of the full Boneh-Boyen signature scheme can be computed under a chosen message attack, using $4d+2$ signature queries, in time $O((\sqrt{p/d}+d^2)T)$, with probability at least $\left(\frac{p-1-d(2d+1)}{p-1}\right)^2$.*

Proof. The proofs of these two theorems are similar. We will give the proof for Theorem 5.3.

(1) Let d be a positive divisor of $p-1$. We will construct an algorithm $\mathcal{A}$ which recovers the private key of the signature scheme under a chosen message attack, using Cheon's algorithm. Suppose $\mathcal{A}$ is given the public key $(g_1, g_2, g_2^x, g_2^y, \zeta)$. The algorithm $\mathcal{A}$ randomly selects a message $m_a \in \mathbb{Z}_p$, and queries for signatures of this same message $d+1$ times. As a result, $\mathcal{A}$ obtains $d+1$ valid (and hopefully distinct) signatures $(\sigma_1, r_1), \ldots, (\sigma_{d+1}, r_{d+1})$, where $\sigma_i = g_1^{\frac{1}{x+m_a+yr_i}}$ for each $i = 1, \ldots, d+1$. Let $h \leftarrow g_1^{1/y}$ and $z_a \leftarrow \frac{x+m_a}{y}$. Then, we have

$$\sigma_i = \left(g_1^{\frac{1}{y}}\right)^{\frac{1}{\frac{x+m_a}{y}+r_i}} = h^{\frac{1}{z_a+r_i}}$$

for each $i = 1, \ldots, d+1$. If the set $\{r_1, \ldots, r_{d+1}\}$ does not consist of distinct elements, then $\mathcal{A}$ aborts. Otherwise, assume $r_1, \ldots, r_{d+1}$ are distinct. Using Corollary 4.2, the algorithm $\mathcal{A}$ calculates

$$h^{\frac{1}{(z_a+r_1)\cdots(z_a+r_{d+1})}}, h^{\frac{z_a}{(z_a+r_1)\cdots(z_a+r_{d+1})}}, \text{ and } h^{\frac{z_a^d}{(z_a+r_1)\cdots(z_a+r_{d+1})}}.$$

Then, it runs Cheon's algorithm in $\mathbb{G}_1$ with these inputs, and obtains $z_a = \frac{x+m_a}{y}$ as output.

Next, $\mathcal{A}$ repeats the above process with a different random message m_b, and obtains $z_b = \frac{x+m_b}{y}$. Since $\mathcal{A}$ knows z_a, z_b, m_a, and m_b, it can solve a linear system of equations to obtain the private exponents x and y.

Since calculating $h^{\frac{1}{(z+r_1)\cdots(z+r_{d+1})}}$, $h^{\frac{z}{(z+r_1)\cdots(z+r_{d+1})}}$, and $h^{\frac{z^d}{(z+r_1)\cdots(z+r_{d+1})}}$ for $z = z_a$ and z_b takes time $\Theta(dT + d^2T_p)$ and Cheon's algorithm has a running time of $\Theta((\sqrt{p/d}+\sqrt{d})T)$, the overall runtime is $\Theta((\sqrt{p/d}+d)T + d^2T_p)$. The attack succeeds if the set $\{r_1, \ldots, r_{d+1}\}$ for m_a consists of distinct elements (and likewise for m_b). Using an argument analogous to the one used in Theorem 4.4, we see that a lower bound for this probability is $\left(\frac{p-1-d(d+1)/2}{p-1}\right)^2$.

(2) We now suppose d is a divisor of $p+1$. The proof here is similar, except that $\mathcal{A}$ needs to calculate

$$h^{\frac{1}{(z+r_1)\cdots(z+r_{2d+1})}}, h^{\frac{z}{(z+r_1)\cdots(z+r_{2d+1})}}, \ldots, h^{\frac{z^{2d}}{(z+r_1)\cdots(z+r_{2d+1})}}.$$

from the signatures $h^{\frac{1}{z+r_1}}, \ldots, h^{\frac{1}{z+r_{2d+1}}}$, for each of $z = z_a$ and z_b. This takes $\Theta(d^2T)$ time, and Cheon's algorithm takes $\Theta((\sqrt{p/d}+d)T)$ time, for a total runtime of $\Theta((\sqrt{p/d}+d^2)T)$. The attack succeeds if the set $\{r_1, \ldots, r_{2d+1}\}$ for each of z_a and z_b consists of distinct elements, and the probability of this is at least $\left(\frac{p-1-d(2d+1)}{p-1}\right)^2$.

6 Runtime analysis

In this section we calculate, both theoretically and experimentally, the complexity of recovering a Boneh-Boyen private key using the algorithms of Theorems 5.2 and 5.3, for various values of d. We also determine, both theoretically and experimentally, the optimal values of d for a given p. To simplify the analysis, we only consider divisors d of $p-1$. In what follows, we refer to this algorithm as the "SDH algorithm" and consider only the case of the basic scheme, where $d+1$ valid signatures are required (assuming that g_1 is not included in the public key). The running time and signature requirements for breaking the full scheme are almost exactly twice as large as for the basic scheme.

6.1 Experimental analysis

Using a 2.4 GHz Core 2 duo, we implemented the SDH algorithm on a collection of 14 different Barreto-Naehrig curves [3] ranging in size from 32 bits to 60 bits, and compared its running time to that of Pollard's lambda and Pollard's rho algorithms for discrete logarithms[1]. We chose Barreto-Naehrig curves because they are highly suitable for pairing-based short signature schemes. For Cheon's algorithm, we chose the Pollard's lambda variant of Cheon's algorithm instead of the baby-step-giant-step variant or variants such as Kozaki et al. [18]; the use of the lambda variant saves memory and is also necessary in order to benefit from parallelization.

Implementing the SDH algorithm is straightforward. We wrote a small program based on the `PBC` library [19] to compute the products listed in Corollary 4.2. Our program is multithreaded and makes use of multiple processor cores, with parallelization being achieved by dividing the main product into subproducts and computing each subproduct separately. For Cheon's algorithm, we used the existing `sdhkangaroo` program [22], which is also based on `PBC`. The original `sdhkangaroo` program maintains a list of distinguished points, defined as those for which the MD5 hash of the point ends in a sufficiently long string of zeros. To improve performance, we modified this program to change the distinguished points to those for which the x-coordinate itself ends in a long string of zeros. For comparison purposes, we also conducted trials of Pollard's lambda and Pollard's rho algorithms for discrete logarithms. Our implementation of Pollard's lambda algorithm was obtained by modifying the `sdhkangaroo` program, and for Pollard's rho algorithm we used the optimized implementation included in the MAGMA Computer Algebra System [20], based on Teske's work [23]. All programs, except for the MAGMA implementation of Pollard's rho algorithm, supported multithreading and made use of both processor cores.

For each curve, we performed a number of trials of the SDH algorithm (at least 50 for each curve), from which we determined empirically the optimal value of the divisor d to use in Cheon's algorithm. In general, this optimal value does not correspond to an actual divisor of $p-1$, but using nearby divisors we were able to estimate the hypothetical performance of the SDH algorithm at the optimal choice of d. (Note that, even when the optimal value of d does not divide $p-1$, near-optimal divisors almost always exist, c.f. Section 6.3.) Figure 1 compares the measured performance of Pollard's lambda and Pollard's rho algorithms against the empirically determined optimal runtime of the SDH algorithm for each curve. Based on the best fit curves, we estimate that the SDH algorithm with the optimal d outperforms Pollard's lambda (resp., Pollard's rho) algorithm for curve sizes greater than 32.5 bits (resp., 50.8 bits).

6.2 Theoretical analysis

We now calculate the theoretical cost of computing Boneh-Boyen private keys using the SDH algorithm. Using an appropriately sized collection of distinguished points, the most optimized version of Pollard's lambda algorithm requires $\approx 2\sqrt{p}$ random walk steps [12]; our implementation, however, averaged $7.9\sqrt{p}$ steps. Each step represents an elliptic curve scalar multiplication, and hence requires $1.5 \log p$ elliptic curve operations if naive methods are used. Over a prime field, each elliptic curve operation takes roughly 15 field multiplications [12]. Hence, our running time for Cheon's algorithm is roughly $7.9(\sqrt{d} + \sqrt{p/d})(1.5 \log p) \cdot 15T_p$ where T_p represents the cost of a field multiplication. In addition, we also need to compute a triplet of the form g, g^x, and g^{x^d}. This requires three applications of Corollary 4.2, at a cost of $\approx 3d^2T_p$; however, since almost all of the multiplications in each computation are identical, the true cost is only $\approx d^2T_p$. (Note also that this step parallelizes linearly, since one can compute subproducts of the outer product on different processors.) Thus the total cost t of the SDH algorithm is

$$t = (7.9(\sqrt{d} + \sqrt{p/d})(1.5 \log p) \cdot 15 + d^2)T_p. \tag{1}$$

This cost is minimized by taking $d = \Theta(p^{\frac{1}{5}}(\log p)^{\frac{2}{5}})$, yielding a corresponding overall running time of $\Theta(p^{\frac{2}{5}}(\log p)^{\frac{4}{5}}T_p)$ for the SDH algorithm. In Figure 2 we compare the optimal values of d predicted by Equa-

[1] All comparisons took place over the base field, i.e., the group $\mathbb{G}_1$ in the pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Such a comparison is valid even though the public key in the Boneh-Boyen scheme lies in $\mathbb{G}_2$, because given a single valid message-signature pair one can recover the secret key of the basic scheme using a discrete log in $\mathbb{G}_1$.

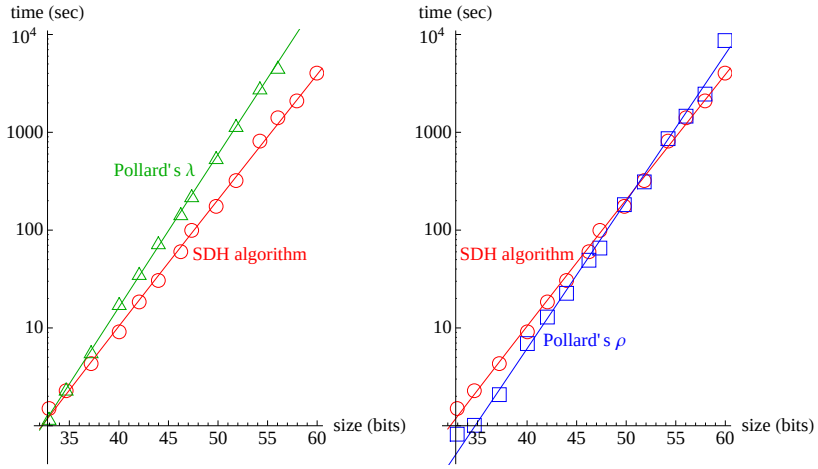


Fig. 1: Log-log plots comparing the optimal running time of the SDH algorithm to Pollard's lambda (left) and Pollard's rho (right) algorithms for discrete log, for Barreto-Naehrig curves of various bit sizes.

curve size (bits)	optimal d (predicted)	optimal d (observed)
32.95	1527	1173
34.68	1985	1545
37.20	2900	2351
40.03	4428	3773
42.05	5977	5676
43.98	7956	7599
46.24	11112	10722
47.34	13066	14508
49.81	18781	19873
51.82	25202	26564
54.23	35828	43795
56.04	46668	56469
57.95	61669	71572
59.97	82715	98733

Fig. 2: Table comparing the optimal values of d predicted in Section 6.2 vs. those observed in Section 6.1.

tion (1) to those observed in Section 6.1. We remark that the asymptotic running time of $\Theta(p^{\frac{2}{5}}(\log p)^{\frac{4}{5}}T_p)$ for optimal d is independent of the precise assumptions used in deriving Equation (1).

6.3 Existence of suitable divisors

Other than increasing the key length, the most obvious defense against the above attack is to use a curve of order p for which $p-1$ and $p+1$ admit no divisors of suitable size. We can estimate the prevalence of such curves using Equation (1). Examining the graph of this equation reveals that the curve is fairly flat for a wide range of values surrounding the optimal value of d. Hence, most sufficiently large pairing-friendly curves admit a divisor d of $p-1$ for which the SDH algorithm runs in nearly optimal time. As an experiment, we enumerated for each of $2^{80}, 2^{90}, \ldots, 2^{160}$ the 100 smallest Barreto-Naehrig curves having at least that many points. Out of these 900 curves, all curves except one (the curve with 146150164166205498805908872805620773627897540432 9 points) admit a divisor for which the runtime predicted by Equation (1) is within a factor of 4 of the optimal time. In addition, Ford [14] has shown asymptotically that a large proportion of primes p admit a divisor d within the interval required for our algorithm. These results indicate that pairing-friendly curves are unlikely to resist the SDH algorithm unless specifically chosen with this property in mind.

7 Conclusion

In this paper, we show that the existential forgery of signatures for both the basic and full versions of the Boneh-Boyen signature scheme can be reduced to the q-SDH problem via an algorithm which is quadratic in q. This result establishes the equivalence of the q-SDH assumption and the security of Boneh-Boyen signatures, thus resolving an open problem posed in [7, 17]. Together with Cheon's solution to q-SDH, the reduction algorithm allows us to recover Boneh-Boyen private keys in time $O(p^{\frac{2}{5}+\varepsilon})$ for groups of order p whenever $p \pm 1$ satisfies certain divisibility properties.

It would be worthwhile to design a new short signature scheme whose security can be proved in the standard model under a weaker assumption than q-SDH. Our proofs of equivalence rely on the fact that the denominator in the exponent of $g^{\frac{1}{x+m+yr}}$ is linear in both m and r. One natural starting point would be to look for signature schemes with nonlinear denominators. One examples of such a scheme is given

in [24], and another example is the scheme $\sigma \leftarrow (g_1^{\frac{1}{x+mr+yr^2}}, r)$. Alternatively, one might hash the message as in [6, §5] or [25], and try to give a security proof without random oracles. We emphasize that we have not studied the security proofs for these schemes, nor have we made any systematic effort to examine the security assumptions underlying them.

8 Acknowledgments

We thank Paulo S. L. M. Barreto, Daniel Brown, Steven Galbraith, Alfred Menezes, and Igor Shparlinski for their helpful comments and suggestions.

References

1. Michael Artin. *Algebra*. Prentice Hall, United States edition, 1991.
2. Joseph Bak and Donald J. Newman. *Complex Analysis*. Springer, 2nd edition, 1996.
3. Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected areas in cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer, Berlin, 2006.
4. Dan Boneh and Xavier Boyen. Efficient selective-ID identity-based encryption without random oracles. In *Advances in Cryptology—EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2004.
5. Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *Advances in Cryptology—EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer, 2004.
6. Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, 2008.
7. Xavier Boyen. The uber-assumption family – a unified complexity framework for bilinear groups. In *2nd International Conference on Pairing-based Cryptography—PAIRING 2008*, volume 5209 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 2008. Available at `http://www.cs.stanford.edu/~xb/pairing08/`.
8. Xavier Boyen and Brent Waters. Full-domain subgroup hiding and constant-size group signatures. In *Proceedings of PKC 2007*, volume 4450 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2007.
9. Daniel R. L. Brown and Robert P. Gallant. The static diffie-hellman problem. Cryptology ePrint Archive: Report 2004/306, 2004. `http://eprint.iacr.org/2004/306`.
10. James Ward Brown and Ruel V. Churchill. *Complex Variables and Applications*. McGraw-Hill, seventh edition, 2004.
11. Jung Hee Cheon. Security analysis of the Strong Diffie-Hellman problem. In *Advances in Cryptology—EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2006.
12. Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, and Frederik Vercauteren, editors. *Handbook of elliptic and hyperelliptic curve cryptography*. Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton, FL, 2006.
13. Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In *Proceedings of PKC 2005*, volume 3386 of *Lecture Notes in Computer Science*, pages 416–431. Springer, 2005.
14. Kevin Ford. The distribution of integers with a divisor in a given interval. *Ann. of Math. (2)*, 168(2):367–433, 2008.
15. Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.
16. Neal Koblitz and Alfred Menezes. Another look at generic groups. *Advances in Mathematics of Communications*, 1(1):13–28, 2007.
17. Neal Koblitz and Alfred Menezes. Another look at non-standard discrete log and Diffie-Hellman problems. *Journal of Mathematical Cryptology*, 2(4):311–326, 2008.
18. Shunji Kozaki, Taketeru Kutsuma, and Kazuto Matsuo. Remarks on Cheon's algorithms for pairing-related problems. In *Pairing 2007*, volume 4575 of *Lecture Notes in Computer Science*, pages 302–316. Springer, 2007.
19. Ben Lynn. The Pairing-Based Cryptography Library, version 0.4.18, 2008. `http://crypto.stanford.edu/pbc/`.
20. MAGMA Computational Algebra System. `http://magma.maths.usyd.edu.au/magma/`.
21. Shigeo Mitsunari, Ryuichi Sakai, and Masao Kasahara. A new traitor tracing. *IEICE Trans. Fundamentals*, E85-A(2):481–84, 2002.

22. Joel Reardon. sdhkangaroo: A kangaroo attack against the strong Diffie Hellman problem, 2007. http://www.cs.uwaterloo.ca/~jreardon/programs.html.
23. Edlyn Teske. On random walks for Pollard's rho method. *Math. Comp.*, 70(234):809–825, 2001.
24. Victor K. Wei and Tsz Hon Yuen. More short signatures without random oracles. Cryptology ePrint Archive, Report 2005/463, 2005. http://eprint.iacr.org/2005/463.
25. Fangguo Zhang, Reihaneh Safavi-Naini, and Willy Susilo. An efficient signature scheme from bilinear pairings and its applications. In *Public Key Cryptography—PKC 2004*, Lecture Notes in Computer Science, pages 277–290, 2004.

A Appendix: Implementation results

We implemented the SDH algorithm for recovering Boneh-Boyen private keys over a collection of Barreto-Naehrig curves [3] ranging in size from 32 to 60 bits. In this appendix we present the details of our experimental setup and the results of our trials.

The SDH algorithm consists of two parts. The first part consists of calculating a triplet of values of the form (g, g^x, g^{x^d}) given a Boneh-Boyen public key and $d+1$ message-signature pairs, via Corollary 4.2. Here x is the corresponding Boneh-Boyen private key, and d is a divisor of $p-1$, where p is the order of the groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ in the pairing. The second part consists of using Cheon's $p-1$ algorithm (Theorem 5.1) to compute x given the triplet (g, g^x, g^{x^d}). In what follows, we refer to the first part as the "reduction" phase and the second part as the "Cheon's algorithm" phase of the algorithm. In our context, the reduction phase requires $\Theta(d^2)$ modular multiplications and Cheon's algorithm requires $\Theta(\sqrt{p/d})$ elliptic curve scalar multiplications, with all other contributions to the running times being negligible. We used Joel Reardon's `sdhkangaroo` program [22], based on the `PBC` library [19], to implement Cheon's algorithm and Pollard's lambda algorithm. For Pollard's rho algorithm, a highly optimized implementation (based on Teske's work [23]) is included as part of the MAGMA Computer Algebra System [20]; we chose to use this implementation rather than re-implement the algorithm ourselves. Although this implementation is derived from a different code base, the running time is comparable to what we would expect based on the outcome of our measurements of Pollard's lambda algorithm, and we believe that the comparison between these implementations remains valid and interesting.

All trials were performed on a 2.4 GHz Intel Core 2 Duo E4600 with 2GB RAM running Fedora 9. Multithreading was used in the reduction phase, the Cheon's algorithm phase, and in Pollard's lambda algorithm for discrete logarithms to distribute the workload equally among the two cores. For Pollard's rho algorithm, multithreading was not used, since the MAGMA program does not support it. We have not attempted to correct for this discrepancy, since we prefer to report the actual numbers, and leave the reader to perform such adjustments if desired (for example by dividing the numbers by two).

Recall that a Barreto-Naehrig curve is an elliptic curve over a prime field of the form $y^2 = x^3 + b$, with cardinality equal to

$$36u^4 + 36u^3 + 18u^2 + 6u + 1$$

for some u. In general, we chose the smallest possible value of b; this constraint along with the value of u suffices to uniquely specify the curve. Figure 3 lists, for each curve appearing in our experiments, the value of u used to generate the curve, the size of the curve in bits, the list of divisors d of $p-1$ that we used, and for each divisor the running times of the two phases of the algorithm. For the running time of Cheon's algorithm, we report the average and standard deviation measured from ten trials of the `sdhkangaroo` algorithm for each divisor. For the reduction phase, there is very little variation in the running time, and hence we report for each divisor the running time measured for a single trial. We also report the estimated optimal values of d and the total running time at the optimal value, using best fit curves of the form $c_1\sqrt{p/d}$ and $c_2d^2 + c_3d$ for the Cheon's algorithm and reduction phases, respectively. Figure 4 contains the corresponding table for Pollard's lambda and Pollard's rho algorithms for discrete logarithm, along with the observed values of d for which the SDH algorithm outperforms Pollard's lambda and Pollard's rho algorithm respectively.

Figures 5 and 6 present the information from Figures 3 and 4 in graphical form. Figure 5 contains quartile plots of the running times for Cheon's algorithm on each curve, showing the average, min, max, and interquartile range, together with the best fit curve. Figure 6 compares the running time of the SDH algorithm with those of Pollard's lambda and Pollard's rho algorithms, using the best fit curves given in Figure 5.

curve size (in bits)	u	d	average time for Cheon's algorithm (in seconds)	standard deviation	running time of reduction phase (seconds)	total time	optimal d and running time t
32.95	123	1004	1.57	0.65	0.38	1.95	$d = 1173$ $t = 1.50$
		1319	0.69	0.36	0.52	1.21	
		1394	0.58	0.26	0.55	1.13	
		1476	0.85	0.22	0.59	1.44	
		1506	0.59	0.18	0.60	1.19	
		2091	0.73	0.36	0.92	1.65	
		2259	0.60	0.27	1.02	1.62	
		2638	0.41	0.20	1.26	1.67	
		2788	0.81	0.57	1.40	2.21	
		3012	0.94	0.49	1.52	2.46	
		3957	0.38	0.17	2.37	2.75	
		4182	0.39	0.10	2.48	2.87	
		4267	0.36	0.14	2.56	2.92	
		4518	0.39	0.13	2.95	3.34	
		5276	0.48	0.20	3.58	4.06	
		6273	0.31	0.09	5.13	5.44	
		7914	0.28	0.14	6.96	7.24	
		8364	0.41	0.22	8.16	8.57	
		8534	0.33	0.17	7.91	8.24	
		9036	0.55	0.23	9.37	9.92	
		10291	0.33	0.13	10.87	11.20	
		11871	0.23	0.08	15.57	15.80	
		12546	0.26	0.08	15.71	15.97	
		12801	0.24	0.08	17.59	17.83	
		15828	0.35	0.14	26.64	26.99	
		17068	0.30	0.13	30.22	30.52	
34.68	166	996	2.24	1.01	0.39	2.63	$d = 1545$ $t = 2.28$
		1446	1.02	0.58	0.59	1.61	
		2892	1.91	0.95	1.44	3.35	
		20003	0.37	0.14	36.73	37.10	
		40006	0.28	0.12	135.01	135.29	
37.20	257	1028	4.47	2.12	0.42	4.89	$d = 2351$ $t = 4.32$
		1424	5.00	0.76	0.61	5.61	
		1691	2.45	1.73	0.76	3.21	
		2136	2.68	1.75	1.02	3.70	
		2848	6.11	2.32	1.52	7.63	
		3382	2.01	1.01	1.91	3.92	
		3779	2.47	1.15	2.21	4.68	
		4272	1.80	0.65	2.69	4.49	
		4883	1.62	0.78	3.36	4.98	
		5073	1.99	0.67	3.62	5.61	
		6168	2.28	0.78	4.63	6.91	
		6764	1.70	0.33	5.35	7.05	
		7558	1.58	0.88	6.53	8.11	
		8544	1.97	1.18	8.42	10.39	
		9766	1.87	0.79	10.07	11.94	
		10146	1.34	0.70	11.47	12.81	
		11337	1.47	0.46	12.85	14.32	
		12336	1.56	0.65	16.62	18.18	
		13528	1.23	0.62	17.60	18.83	
		15116	1.13	0.64	21.48	22.61	

Fig. 3: Running times for the SDH algorithm.

curve size (in bits)	u	d	average time for Cheon's algorithm (in seconds)	standard deviation	running time of reduction phase (seconds)	total time	optimal d and running time t
40.03	420	2051	9.25	5.29	1.01	10.26	$d = 3773$ $t = 9.12$
		2520	11.31	6.80	1.35	12.66	
		2930	5.82	3.19	1.65	7.47	
		3516	5.29	3.19	2.07	7.36	
		4102	4.76	2.03	2.61	7.37	
		5274	6.79	2.56	3.77	10.56	
		5860	6.13	3.13	4.46	10.59	
		6153	5.89	2.89	5.04	10.93	
		7032	5.40	3.95	5.90	11.30	
		8204	4.28	2.39	8.26	12.54	
		8790	4.41	1.95	8.56	12.97	
		10255	3.76	1.88	11.07	14.83	
42.05	596	1192	30.47	19.57	0.55	31.02	$d = 5676$ $t = 18.44$
		1788	18.64	11.79	0.89	19.53	
		1822	24.08	8.49	0.92	25.00	
		2733	21.28	4.97	1.52	22.80	
		3576	25.08	9.02	2.20	27.28	
		3644	14.81	4.65	2.29	17.10	
		5466	16.48	6.48	4.14	20.62	
		7288	19.63	5.81	6.77	26.40	
		10932	8.64	3.08	12.48	21.12	
		21864	11.36	6.21	48.25	59.61	
43.98	833	2352	32.25	15.61	1.31	33.56	$d = 7599$ $t = 30.56$
		2856	36.53	12.35	1.70	38.23	
		3332	32.52	14.03	2.11	34.63	
		3808	43.07	21.12	2.55	45.62	
		4704	47.02	15.60	3.45	50.47	
		5433	32.82	13.64	4.19	37.01	
		6664	22.26	12.54	5.96	28.22	
		7244	18.36	10.99	6.76	25.12	
		9996	16.68	9.30	10.95	27.63	
		10866	15.51	8.02	12.78	28.29	
		11424	22.49	14.02	13.75	36.24	
		12677	15.98	4.36	18.00	33.98	
		13328	14.17	6.48	19.30	33.47	
		14488	14.75	6.31	20.70	35.45	
46.24	1233	2192	140.71	44.18	1.25	141.96	$d = 10722$ $t = 60.14$
		2466	77.69	44.33	1.44	79.13	
		3288	68.83	33.46	2.11	70.94	
		3699	67.36	26.55	2.52	69.88	
		4384	68.28	15.23	3.17	71.45	
		4932	49.64	18.93	3.79	53.43	
		6576	63.70	21.68	6.04	69.74	
		7398	69.22	35.11	7.28	76.50	
		8768	92.30	45.57	9.08	101.38	
		9864	47.84	20.33	10.98	58.82	
		13152	41.90	15.41	17.80	59.70	
		14796	39.10	19.70	23.70	62.80	
		19728	24.62	13.09	40.90	65.52	

Fig. 3: Running times for the SDH algorithm (continued).

curve size (in bits)	u	d	average time for Cheon's algorithm (in seconds)	standard deviation	running time of reduction phase (seconds)	total time	optimal d and running time t
47.34	1492	2984	181.09	104.45	1.89	182.98	$d = 14508$ $t = 99.38$
		4476	140.84	63.75	3.30	144.14	
		7087	88.54	28.38	6.84	95.38	
		8952	129.02	57.08	10.09	139.11	
		14714	72.46	33.93	21.99	94.45	
		21261	39.20	15.06	48.07	87.27	
		28348	44.80	20.82	72.58	117.38	
		42522	36.16	10.02	177.33	213.49	
		56696	67.22	31.42	267.78	335.00	
		85044	26.60	12.71	648.44	675.04	
49.81	2289	2088	502.15	179.30	1.26	503.41	$d = 19873$ $t = 174.95$
		2436	312.33	131.14	1.57	313.90	
		2994	295.97	152.87	1.97	297.94	
		3488	186.09	101.16	2.46	188.55	
		4032	240.02	97.29	2.97	242.99	
		5232	320.04	186.83	4.24	324.28	
		6496	237.38	125.92	5.90	243.28	
		7424	185.56	126.12	7.65	193.21	
		8352	232.65	89.76	9.27	241.92	
		9156	228.17	126.65	10.62	238.79	
		9744	267.98	113.66	11.10	279.08	
		10752	395.97	162.02	14.01	409.98	
		11976	177.22	71.87	15.72	192.94	
		12992	198.68	91.44	19.32	218.00	
		13952	164.80	102.68	23.63	188.43	
		14848	243.71	86.97	24.70	268.41	
		15968	142.30	72.06	27.94	170.24	
		17964	145.50	71.33	34.65	180.15	
		19433	179.55	61.06	36.46	216.01	
		22127	145.72	93.93	46.63	192.35	
		25288	184.08	82.86	65.08	249.16	
		28449	112.86	62.01	73.54	186.40	
		29232	106.86	49.87	92.76	199.62	
		32256	174.39	59.88	96.47	270.86	
		35928	80.81	48.72	113.13	193.94	
		37932	49.74	39.59	143.12	192.86	
		41856	98.42	35.73	150.84	249.26	
		44254	88.36	29.05	168.37	256.73	
		47904	70.61	28.55	196.79	267.40	
		50576	77.61	30.72	214.62	292.23	
		54391	69.85	29.28	247.10	316.95	
		57884	62.65	35.56	279.29	341.94	
		62784	69.64	23.07	323.61	393.25	
		66381	66.31	18.20	424.16	490.47	
		71856	61.42	19.85	421.50	482.92	
		75864	95.42	35.52	470.53	565.95	
		77952	75.55	37.68	498.27	573.82	
		83832	76.85	35.06	671.00	747.85	
		88508	69.20	34.94	635.71	704.91	
		95808	66.65	28.57	743.79	810.44	
		97664	62.52	31.60	904.66	967.18	

Fig. 3: Running times for the SDH algorithm (continued).

curve size (in bits)	u	d	average time for Cheon's algorithm (in seconds)	standard deviation	running time of reduction phase (seconds)	total time	optimal d and running time t
51.82	3241	20461	215.30	108.79	40.79	256.09	$d = 26564$ $t = 320.94$
		23384	303.41	185.10	52.52	355.93	
		25928	261.15	158.33	62.26	323.41	
		34262	228.84	88.91	117.28	346.12	
		38892	176.64	77.73	132.31	308.95	
		46768	262.05	138.55	186.08	448.13	
		51856	251.92	143.32	251.03	502.95	
		61383	154.63	64.46	312.11	466.74	
		68524	165.26	47.44	453.10	618.36	
		73154	120.97	77.30	437.14	558.11	
		77784	174.68	80.36	577.40	752.08	
		81844	159.18	68.87	603.36	762.54	
		102786	127.19	64.03	842.97	970.16	
54.23	4918	14754	972.69	202.82	25.28	997.97	$d = 43795$ $t = 812.99$
		20654	660.99	128.14	41.60	702.59	
		29508	1001.36	596.09	81.68	1083.04	
		30981	745.61	284.50	87.43	833.04	
		41308	1090.91	590.50	150.64	1241.55	
		56557	468.01	193.67	311.91	779.92	
		61962	416.84	203.31	320.34	737.18	
		113114	498.24	188.56	1033.22	1531.46	
56.04	6732	8874	1929.99	1159.99	10.72	1940.71	$d = 56469$ $t = 1407.48$
		12771	1505.63	680.92	18.23	1523.86	
		16082	1017.31	719.57	27.06	1044.37	
		20196	1106.45	535.75	44.70	1151.15	
		24123	894.96	497.72	63.08	958.04	
		27434	997.73	718.27	70.46	1068.19	
		32164	1300.73	586.14	94.25	1394.98	
		35496	1527.55	917.02	113.11	1640.66	
		40392	1898.90	714.81	147.81	2046.71	
		48807	992.79	365.96	204.12	1196.91	
		54868	755.08	342.15	254.70	1009.78	
		64328	1103.86	654.88	345.04	1448.90	
		68904	926.47	588.88	436.13	1362.60	
		78948	493.77	129.63	593.43	1087.20	
		84796	589.86	183.23	590.22	1180.08	
		96492	381.06	217.74	754.08	1135.14	
		102168	871.60	498.97	991.40	1863.00	
57.95	9379	9379	4262.57	2535.77	11.90	4274.47	$d = 71572$ $t = 2094.10$
		18758	2152.00	1043.85	35.89	2187.89	
		28137	2688.96	1530.03	84.10	2773.06	
		37516	4421.34	2711.74	138.67	4560.01	
		51377	1904.58	817.78	226.17	2130.75	
		56274	1865.67	409.99	310.73	2176.40	
		69947	1387.84	628.00	406.34	1794.18	
		102754	1382.66	916.93	861.20	2243.86	
		112548	1765.93	1326.17	1058.57	2824.50	
		139894	1008.50	557.78	1577.60	2586.10	

Fig. 3: Running times for the SDH algorithm (continued).

curve size (in bits)	u	d	average time for Cheon's algorithm (in seconds)	standard deviation	running time of reduction phase (seconds)	total time	optimal d and running time t
59.97	13308	27144	7351.51	3611.18	78.12	7429.63	
		39924	4047.05	1647.40	142.74	4189.79	
		57668	3473.59	1783.98	325.06	3798.65	
		64322	2807.24	1023.31	386.01	3193.25	
		79848	3777.75	1719.38	583.59	4361.34	$d = 98733$
		86502	2524.81	989.76	721.42	3246.23	$t = 4029.20$
		96483	2097.68	914.98	886.83	2984.51	
		115336	6044.95	2567.84	1190.53	7235.48	
		129753	3017.17	1694.44	1496.18	4513.35	
		173004	1822.12	1194.14	2414.70	4236.82	

Fig. 3: Running times for the SDH algorithm (continued).

curve size (bits)	Pollard's λ avg. time (seconds)	Pollard's λ std. dev. (runtime)	Pollard's λ avg. # of steps	Pollard's λ std. dev. (# of steps)	Pollard's ρ avg. time (seconds)	Pollard's ρ std. dev. (runtime)	$d^{\lambda}_{\min}$	$d^{\lambda}_{\max}$	$d^{\rho}_{\min}$	$d^{\rho}_{\max}$
32.95	1.19	0.78	598140.5	398893.1	0.82	0.47	-	-	-	-
34.68	2.37	0.91	1191051.6	457466.8	1.01	0.40	1113	2073	-	-
37.20	5.71	3.22	2817514.4	1595634.5	2.08	0.91	821	5034	-	-
40.03	17.77	5.20	8680161.8	2552190.8	6.92	3.83	562	11645	-	-
42.05	36.07	17.16	17541406.6	8394859.9	12.92	8.50	901	16279	-	-
43.98	74.33	41.59	35980247.8	20099792.6	22.60	11.85	757	26022	-	-
46.24	147.28	74.38	69746834.0	35363891.5	49.23	40.50	1108	35190	-	-
47.34	224.91	79.84	105809907.7	37207371.2	61.53	51.45	1743	44667	-	-
49.81	551.45	327.06	253471890.7	150399968.9	182.12	84.54	1253	74399	14195	26525
51.82	1174.83	565.79	536513434.4	259357762.7	311.32	117.00	1286	112618	-	-
54.23	2843.83	2063.27	1289321991.2	938829405.4	861.28	540.66	2192	179523	29530	61046
56.04	4622.31	2772.68	2070262591.5	1244353765.0	1462.00	1318.38	3397	210101	44877	67577
57.95	n/a	n/a	n/a	n/a	2458.46	1651.48	n/a	n/a	35906	120406
59.97	n/a	n/a	n/a	n/a	8671.82	5883.56	n/a	n/a	13113	297745

Fig. 4: Observed running times of Pollard's lambda and Pollard's rho algorithms for discrete logarithms. The right side of the table lists the observed minimum and maximum values of d for which the SDH algorithm outperforms Pollard's lambda and Pollard's rho algorithm respectively. (Note: Information on the number of steps performed in the random walk was not available for the MAGMA implementation of Pollard's rho algorithm.)

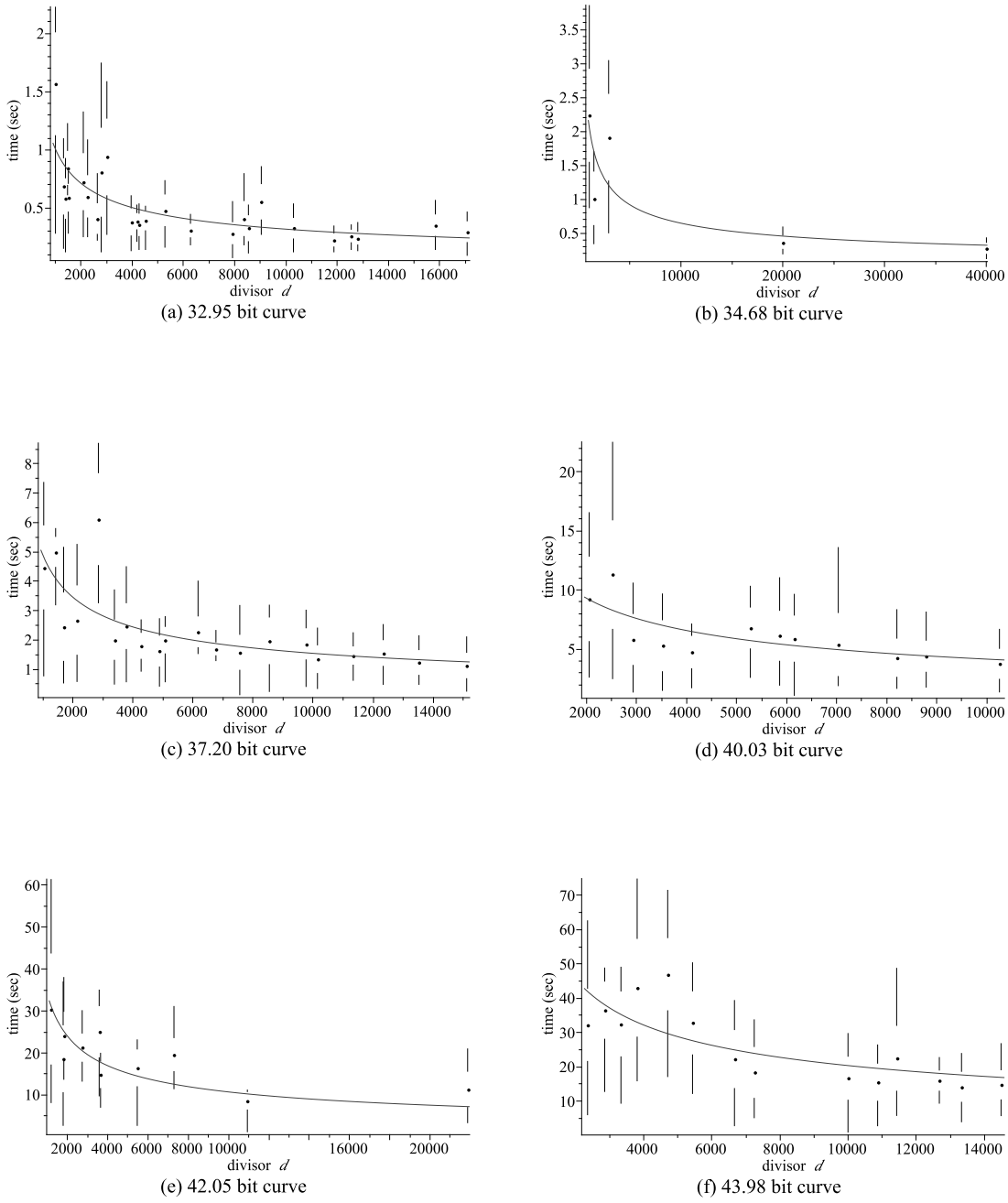
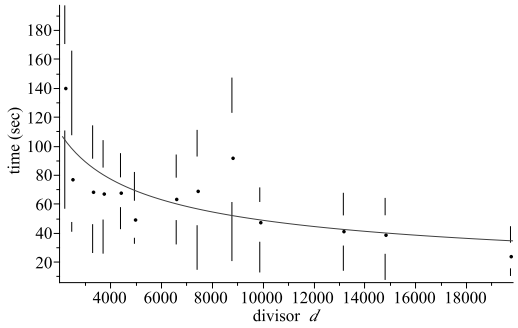
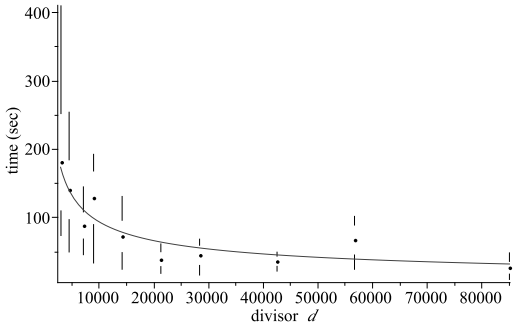


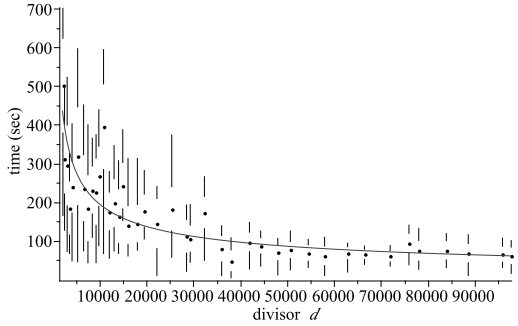
Fig. 5: Quartile plots of the running times for Cheon's algorithm.



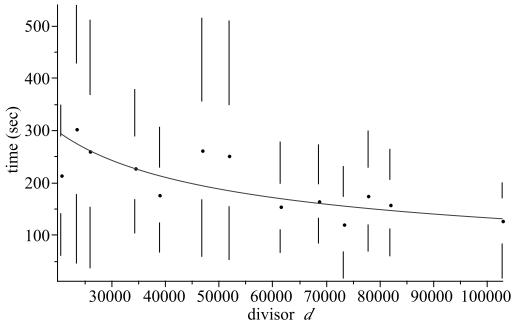
(g) 46.24 bit curve



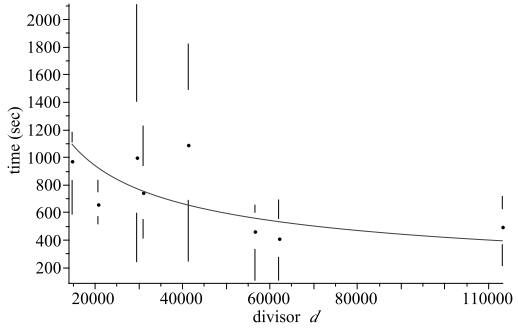
(h) 47.34 bit curve



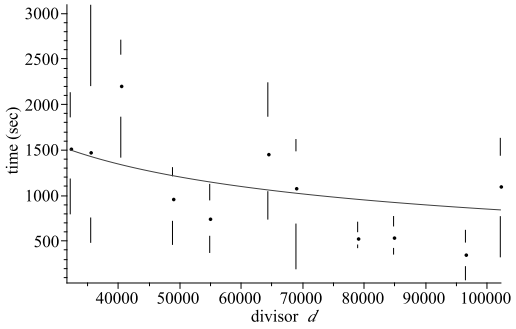
(i) 49.81 bit curve



(j) 51.82 bit curve

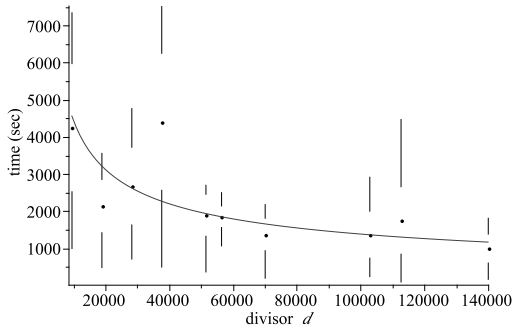


(k) 54.23 bit curve

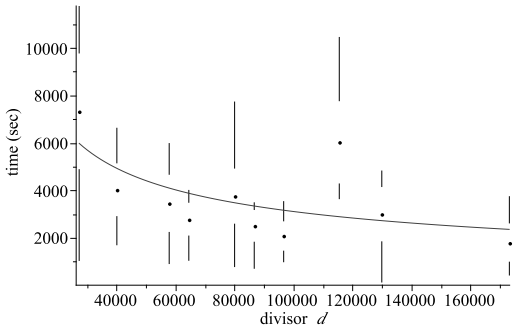


(l) 56.04 bit curve

Fig. 5: Quartile plots of the running times for the Cheon's algorithm (continued).

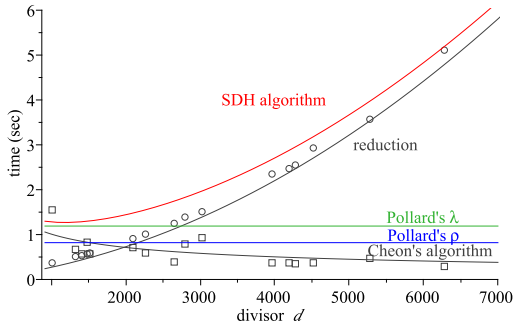


(m) 57.95 bit curve

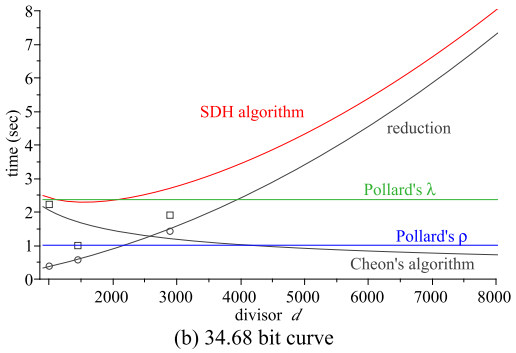


(n) 59.97 bit curve

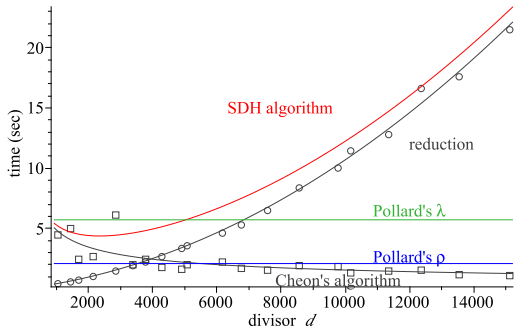
Fig. 5: Quartile plots of the running times for the Cheon's algorithm (continued).



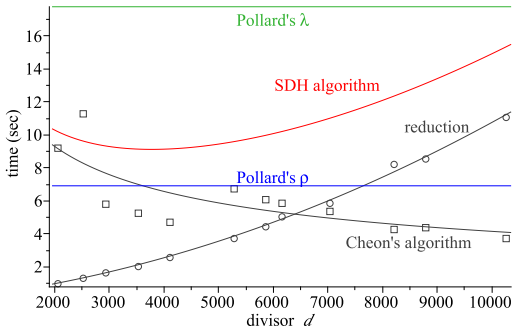
(a) 32.95 bit curve



(b) 34.68 bit curve

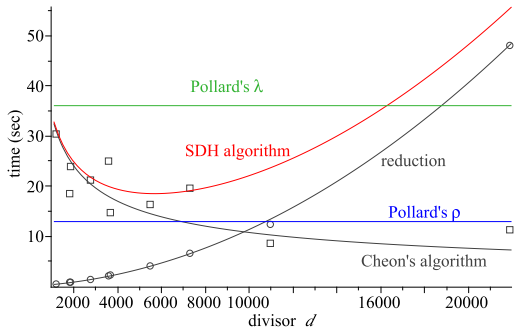


(c) 37.20 bit curve

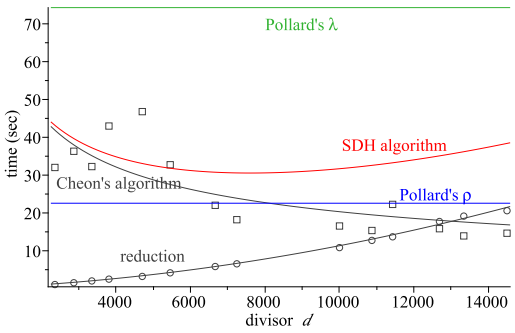


(d) 40.03 bit curve

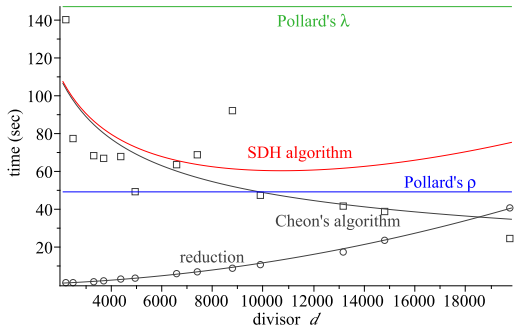
Fig. 6: Comparison of running times for the SDH algorithm, Pollard's lambda algorithm, and Pollard's rho algorithm.



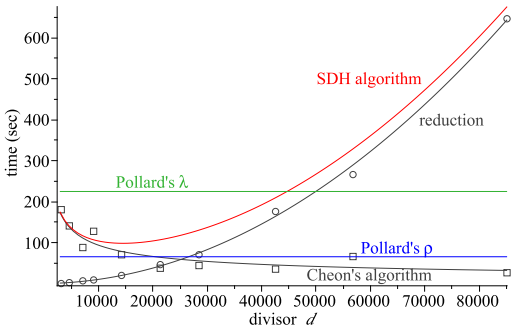
(e) 42.05 bit curve



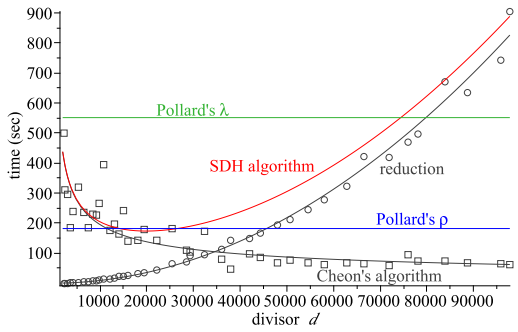
(f) 43.98 bit curve



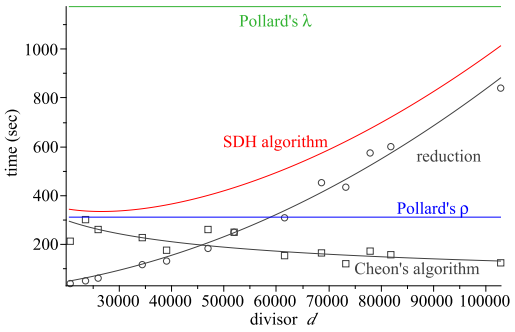
(g) 46.24 bit curve



(h) 47.34 bit curve

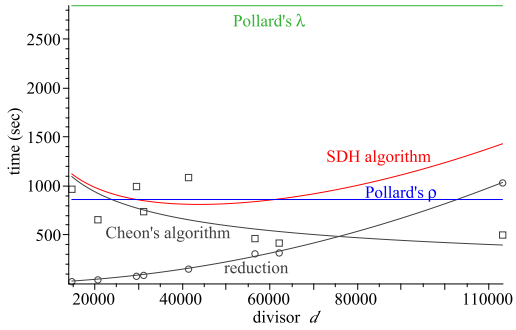


(i) 49.81 bit curve

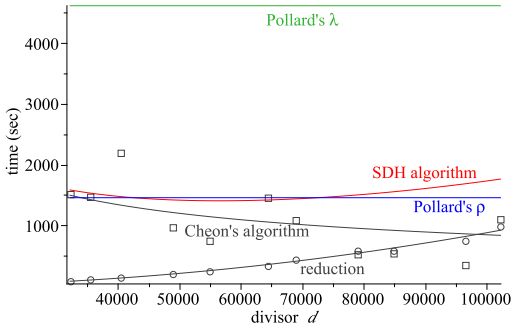


(j) 51.82 bit curve

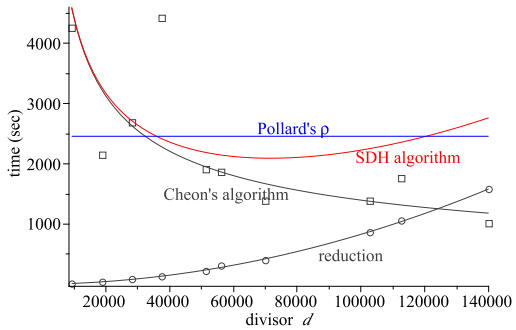
Fig. 6: Comparison of running times for the SDH algorithm, Pollard's lambda algorithm, and Pollard's rho algorithm (continued).



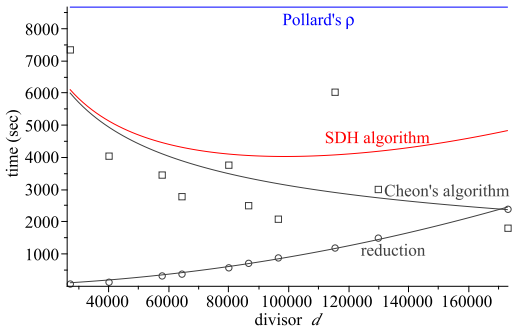
(k) 54.23 bit curve



(l) 56.04 bit curve



(m) 57.95 bit curve



(n) 59.97 bit curve

Fig. 6: Comparison of running times for the SDH algorithm, Pollard's lambda algorithm, and Pollard's rho algorithm (continued).