ANOTHER LOOK AT GENERIC GROUPS

NEAL KOBLITZ AND ALFRED MENEZES

Abstract. Starting with Shoup's seminal paper [24], the generic group model has been an important tool in reductionist security arguments. After an informal explanation of this model and Shoup's theorem, we discuss the danger of flaws in proofs. We next describe an ontological difference between the generic group assumption and the random oracle model for hash functions. We then examine some criticisms that have been leveled at the generic group model and raise some questions of our own.

1. Introduction

In many cryptographic settings, systems based on elliptic curves are more efficient than those based on either the ring of integers modulo an RSA-modulus N or the multiplicative group of a finite field. The main reason is that in the latter cases there are subexponential-time index calculus algorithms that solve the underlying hard problems, whereas no subexponential-time algorithms are known for the discrete logarithm problem on a (suitably chosen) elliptic curve. Moreover, for the most part[1] the best available algorithms are ones that have nothing to do with the specific structure of the elliptic curve group, but rather would work with essentially the same running time on any group. Such algorithms are said to be "generic." It was noticed that the two best generic algorithms — Shanks' "baby-step/giant-step" and Pollard's rho — each require time roughly $O(\sqrt{q})$, where q is the order of the largest prime-order subgroup.[2] The question naturally arose: Could there be faster generic algorithms?

In 1997, Shoup [24] answered this question in the negative. To do this he introduced the notion of the "generic group model" (a somewhat different

Date: July 4, 2006; revised on September 4, 2006.

Key words and phrases. Cryptography, Public Key, Generic Group.

[1] By grouping together a point with its negative on an elliptic curve, one can speed up generic algorithms by a factor of $\sqrt{2}$; and if the curve is defined over a small subextension, one can get a slightly greater speed-up by grouping together points that are conjugate over the subextension. In certain very special cases one has algorithms for the elliptic curve discrete logarithm that are substantially faster than generic algorithms. These cases include supersingular elliptic curves and elliptic curves for which an efficient Weil descent can be constructed.

[2] Normally the elliptic curves used in cryptography are chosen to have a subgroup of prime order of the same order of magnitude as the order of the whole group, and the cryptosystem uses that subgroup.

version of this concept was introduced earlier by Nechaev [19]). We shall give a sketch of his proof in §2. The idea of the generic group model is to give a precise definition of what it means to have an algorithm that does not make use of any special features of the group.

For simplicity suppose that the multiplicatively-written group G is cyclic of prime order q. Such a group is isomorphic to the additive group $\mathbb{Z}/q\mathbb{Z}$, and for any non-identity element $g \in G$ an explicit isomorphism from $\mathbb{Z}/q\mathbb{Z}$ to G can be efficiently constructed by sending i to g^i. Namely, whatever formulas we have for the group operation can be used in conjunction with some version of repeated-squaring in order to exponentiate in polynomial time. The *discrete logarithm problem* in G to the base g is the problem of inverting the exponentiation $i \mapsto g^i$.

In the generic group model one supposes that instead of formulas for the group operation we have an "oracle" that for any i will give us an "encoding" $\sigma(i)$. In addition, if we have two encodings $\sigma(i)$ and $\sigma(j)$ (but we do not necessarily know i or j), then the oracle will give us $\sigma(i \pm j) = \sigma(i)\sigma(j)^{\pm 1}$. By repeatedly querying the oracle, we can also efficiently determine $\sigma(ri+sj) = \sigma(i)^r\sigma(j)^s$ for any integers $0 \le r, s < q$. Without loss of generality we may suppose that we're allowed to ask for the value $\sigma(ri+sj)$ in a single query. Thus, the oracle will tell us either the encoding of an integer i, $0 \le i < q$, or else the element $\sigma(i)^r\sigma(j)^s$ for $0 \le r, s < q$ of our choice. However, the oracle reveals no other information. The way to ensure this is to stipulate that the oracle's encodings are randomly selected elements from some set of bitstrings. The only condition on the oracle's responses is that if the same group element is queried a second time, it must respond with the same encoding.

2. Shoup's Theorem

Here is a more precise (but informal) description of how a generic group oracle works. The input to the oracle is of one of two types:

Type 1: an integer i, $0 \le i < q$. In this case the oracle outputs an encoding $\sigma(i)$ and keeps a record of it. If the integer i has already been queried, then the same $\sigma(i)$ is returned as before; otherwise, the oracle chooses a random value for $\sigma(i)$ that is different from all the earlier values it chose.

Type 2: a 4-tuple r, s, $\sigma(i)$, $\sigma(j)$, where $0 \le r, s < q$ and $\sigma(i)$ and $\sigma(j)$ are outputs from earlier queries.[3] In this case the oracle must give the encoding of the linear combination $ri + sj \bmod q$. (The oracle can of course determine i and j from its records, which list the pairs $(i, \sigma(i))$ from all the previous queries.) If the oracle already returned a value for $\sigma(ri+sj \bmod q)$ earlier, then the same value is returned again; otherwise, the oracle chooses a random value for $\sigma(ri+sj \bmod q)$ that is different from the earlier random values and keeps a record of it.

[3] The person who queries the oracle may or may not know i or j. The earlier queries may have been made by other parties.

Following the exposition in [28], we now give a sketch of the proof of Shoup's theorem, which says that the discrete logarithm problem (DLP) in a generic group of order q cannot be solved in expected time less than $O(\sqrt{q})$.

The input to the DLP is a generator σ_1, which without loss of generality we may suppose to be $\sigma(1)$, and a second element $\sigma_2 = \sigma(x)$. We have to determine x. The only information that we can obtain is the oracle's responses to queries of the above two types. The k-th such response will be a value of the form $\sigma(r_k + s_k x)$, where we know r_k and s_k, but not x. Since these values are random, they will have no relation to one another until we query a linear combination $r_k + s_k x$ that happens to be equal modulo q to an earlier $r_\ell + s_\ell x$. As soon as such a collision occurs, we can immediately solve for x (unless $s_k \equiv s_\ell \pmod{q}$, in which case the pairs (r_k, s_k) and (r_ℓ, s_ℓ) are identical mod q). Namely, if $\sigma(r_k + s_k x) = \sigma(r_\ell + s_\ell x)$, it follows that $r_k + s_k x \equiv r_\ell + s_\ell x \pmod{q}$, and so

$$x \equiv -\frac{r_k - r_\ell}{s_k - s_\ell} \pmod{q}.$$

Conversely, if two queries with pairs (r_k, s_k) and (r_ℓ, s_ℓ) that are distinct mod q lead to a solution, then x must have the value given above. This means that after t queries there are at most $\binom{t}{2}$ possible values of x that would be found. Thus, there is insignificant probability of success until $\binom{t}{2}$ is of order q, in other words, until t is approximately $\sqrt{q}$.

3. The Danger of Flaws in Proofs

Ever since Shoup [25] discovered a subtle but crucial flaw in the Bellare-Rogaway proof [3] of security of their Optimal Asymmetric Encryption Padding (OAEP), researchers have been painfully aware of the danger of flaws in reductionist security arguments. It is possible for a "proof of security" to be accepted by the research community and influence cryptographic practice for many years before it is found to be fallacious. Stern, Pointcheval, Malone-Lee, and Smart [27] comment:

> Methods from *provable security*, developed over the last twenty years, have been recently extensively used to support emerging standards. However, the fact that proofs also need time to be validated through public discussion was somehow overlooked. This became clear when Shoup found that there was a gap in the widely believed security proof of OAEP against adaptive chosen-ciphertext attacks.... the use of provable security is more subtle than it appears, and flaws in security proofs themselves might have a devastating effect on the trustworthiness of cryptography.

We should not be surprised if proofs using the generic group model turn out to be as susceptible to flaws as other types of reductionist security arguments. In this section we discuss one such case.

3.1. **RSA-S1.** We start by describing RSA-S1 [16], which is a method for an untrusted server to assist an RSA signer with the exponentiation $y = x^d \bmod N$. Here d is the secret exponent, and (in the simplest version of RSA signatures) x is the hash-value of the message and y is the signature. In RSA-S1 the signer once and for all chooses a vector $\mathbf{d} = (d_1, \ldots, d_m)$ of positive integers and a vector $\mathbf{f} = (f_1, \ldots, f_m)$ of 0's and 1's, such that $\mathbf{f}$ has Hamming weight k and $\sum_{f_i=1} d_i \equiv d \pmod{\varphi(N)}$. Here $\mathbf{d}$ is public and $\mathbf{f}$ is secret (but for simplicity we may as well suppose that k is public). When the signer wants to compute $x^d \bmod N$, the server sends her the m values $x^{d_i} \bmod N$, after which the signer computes $x^d = \prod_{f_i=1} x^{d_i} \bmod N$.

We consider passive attacks that require only one round of the protocol. For several years after RSA-S1 was introduced, the best was the following meet-in-the-middle attack [21, 20]. For simplicity suppose that k and m are even. First choose a random subset $S \subset \{1, 2, \ldots, m\}$ consisting of $m/2$ indices; let S' denote the complement of S. For all possible subsets $T \subset S$ of cardinality $k/2$ and all possible subsets $T' \subset S'$ of cardinality $k/2$ the adversary computes $\prod_{i \in T} x^{d_i} \bmod N$ and $y \left(\prod_{i \in T'} x^{d_i}\right)^{-1} \bmod N$, and sorts and compares these two sets. As soon as a collision occurs, the adversary finds $d = \sum_{i \in T \cup T'} d_i$ modulo the order of x. It is not hard to compute the expected running time of this algorithm, which is roughly $\sqrt{\binom{m}{k}}$ — that is, the squareroot of the number of possible choices of the secret vector $\mathbf{f}$.

3.2. **Proof of security.** A decade after RSA-S1 was introduced, Merkle and Werchner [17] gave a security argument for it in Nechaev's version of the generic group model. They assumed that RSA itself is secure, i.e., that N is suitably chosen so that index calculus methods for factoring N are not feasible. They noted that the attack described above is generic — no special properties of the group $G = (\mathbb{Z}/N\mathbb{Z})^*$ are used. The situation is then analogous to DSA (see §4 below), where $\mathbb{F}_p$ is assumed to have been chosen large enough to preclude the use of index calculus, and so an adversary is stuck with generic DLP algorithms in the order-q subgroup.

Notice that an adversary who wants to find the secret exponent d from x and y is trying to solve a discrete log problem in the group $G = (\mathbb{Z}/N\mathbb{Z})^*$. The group is not of prime order — and in fact the adversary doesn't know its order $\varphi(N)$ — but otherwise we're in a setting similar to that in §§1-2. It might seem that an approach very similar to Shoup's technique in [24] should work here as well.

On p. 102 of [17] the authors define a generic attack on RSA-S1 by essentially generalizing and abstracting the attack described above. They then argue — much as Shoup did — that the only way the adversary can succeed is to obtain a collision, and this is expected to take $O\left(\sqrt{\binom{m}{k}}\right)$ steps.

3.3. **The flaw.** However, within a few years after publication of [17] successful attacks had been mounted against RSA-S1, most dramatically by

Nguyen and Shparlinski [20], who showed that in the case of low public exponent e one can use lattice basis methods to break RSA-S1 in one round with a passive attack.

How is it possible for a "provably secure" system to be so insecure? Clearly an explanation is needed. The lesson Nguyen and Shparlinski draw is that "our result throws doubt on the real significance of security proofs in the generic model, at least for server-aided RSA protocols." We don't believe that that's the right conclusion. The problem is not with the generic group model, but rather with the proof itself, which contains a blatant flaw.

In [17] the proof assumes that any adversary must approach his task as a DLP problem and use some collision-finding method, as was done in Shoup's proof. The authors failed to notice that, alternatively, the attacker can work directly with the subset-sum problem $\sum_{f_i=1} ed_i \equiv ed \equiv 1 \pmod{\varphi(N)}$. It's a little tricky, because $\varphi(N)$ is secret. But under certain conditions (including small e), Nguyen and Shparlinski show how to deal with that by introducing a new variable which takes small values.

The attacker in [20] does not even need to know the group elements x, y; nor does he need to call upon the group operation oracle. But the attack is certainly generic, since it uses no special properties of the group G.

The fallacy in [17] was to ignore the fact that the attacker has not only the input to a DLP, but also the input to a subset-sum problem. It is striking that the Merkle–Werchner definition of a generic attack on RSA-S1 neglects even to include the public exponent e in the input, although obviously the attacker knows this information. Indeed, the type of attack described in that definition does not make use of the public exponent. However, a proof of security is fallacious if it assumes that an attacker will be ignorant of or unable to use part of the public key. As in the case of OAEP, we again see how risky it is to put confidence in a "proof" of security simply on the basis of its initial acceptance by referees and conference attendees.[4]

4. Contrast with the Random Oracle Model

Because the random oracle model has been around longer [2] and has been used far more extensively than the generic group model, cryptographers tend to be more familiar with it. For this reason, when introducing the generic group model, authors often compare it to the random oracle model, and state or imply that they are very similar (e.g., see p. 241 of [28] and p. 74 of

[4] The argument of Merkle–Werchner [17] has two other flaws. In the first place, as they point out, the Nechaev model equates running time to the number of oracle queries and allows the adversary to spend an unlimited amount of time working with parameters that are not group elements. But this means that the adversary could, for example, factor N. In the second place, Merkle and Werchner assume that the exponent d is randomly determined in the course of the implementation of the protocol, whereas in RSA-S1 the value of d is given beforehand. (This is crucial in most real-world applications, for example, small-encryption-exponent RSA.) It appears that their argument, if it were not fallacious for other reasons, would not be hard to modify so as to hold when d is specified in advance.

[26]). This is not quite correct. In reality, the two models relate to real-world cryptography in very different ways.

On the one hand, although the random oracle model is an idealization of a hash function, it is a realistic one in the following strong sense. Let $h(x)$ be a concrete hash function. Suppose that we supply a sequence of inputs $x_1, x_2, \ldots$ to two black boxes, one of which computes the hash function values $h(x_i)$ and the other of which outputs random values r_i. If h is a good hash function, then unless we actually compute a hash value we should not be able to guess (with greater than a 50% chance of being correct) which of the two sequences is the $h(x_i)$ and which is the r_i. No amount of statistical or number-theoretic testing will help us distinguish between the hash function and a random function. Of course, an algorithm for the hash function is publicly known, so we can distinguish between the two sequences by computing one of the values $h(x_i)$. But nothing short of that is of any help. Thus, it is reasonable to think of a well-constructed real-world hash function as a deterministic function that is essentially indistinguishable from a random function.

In contrast, the generic group model is not a literal description of any of the groups that might be used in real-world cryptography. For instance, encodings of elements in general are easy to distinguish from random strings without even applying the group operation. Here are some examples:

(1) The identity element has a very special encoding. For the multiplicative group of a finite field it is 1, and for an elliptic curve in Weierstrass form it is $(0, 1, 0)$ (in projective coordinates).

(2) In the DSA group, which consists of elements x of $\mathbb{F}_p^*$ of order q (where q is a prime divisor of $p-1$), all group elements are quadratic residues mod p, and so can be distinguished from a random string without even using mod-p group operations (since one can evaluate $\left(\frac{x}{p}\right)$ using quadratic reciprocity). Similarly, the x-coordinates of points on an elliptic curve with equation $y^2 = f(x)$ over $\mathbb{F}_p$ all satisfy the relation $\left(\frac{f(x)}{p}\right) = 1$.

(3) On an elliptic curve with equation $y^2 = f(x)$, a point and its inverse under the group law both have the same x-coordinate.

(4) The DSA group is a subgroup of a group (namely $\mathbb{F}_p^*$) that has a large subset of "small" elements (namely $\{1, 2, \ldots, [\sqrt{p}]\}$) for which the group operation on the encodings of two elements is given by ordinary integer multiplication.

When the generic group model is applied to a concrete discrete-log cryptosystem, such as DSA or ECDSA, one is claiming that the model is an accurate reflection of the actual group being used *only when someone is trying to solve a problem related to discrete logarithms* (such as variants of the Diffie-Hellman problem). The generic group model is merely a convenient way to formally guarantee that the group has no special structure or property that would be of use in solving this problem. If the encodings have special features that lead to algorithms for the problem, the claim is that these algorithms are no faster than the algorithms that don't use such features. For example, the DLP in the DSA group can be solved using index calculus algorithms in the larger group $\mathbb{F}_p^*$ (see (4) above). However, in practice one chooses p large enough compared to q so that these algorithms are slower than $\sqrt{q}$ and so enjoy no advantage over generic algorithms.

It is also important to note that when the generic group model is used to prove a result about the security of a cryptosystem that is based on a concrete group, no claim is being made about the group's generic behavior in situations that are unrelated to the result being proved. For example, suppose that we have a protocol with the unusual property that it can somehow be misused if there is a way to find two group elements that give the same input at a certain stage of the protocol. If this input is the x-coordinate of a point on an elliptic curve, then obviously we're in trouble (see (3) above). That, however, has nothing to do with the question of whether or not the generic group assumption is valid for the elliptic curve in proofs of security against an attacker who isn't helped by having two points with the same x-coordinate.

5. Dubious Critiques

In this section we discuss some criticisms of the generic group model or its uses that do not hold up well under close examination.

5.1. **Fischlin's pitfalls.**

In [13] Fischlin discussed what he called "pitfalls" in using the generic group model in security reductions. His most important example related to the Schnorr signature scheme [22], which had just been proved secure in the generic group model by Schnorr and Jakobsson [23].

We first recall the signature scheme [22].

Schnorr key generation. Let q be a large prime, and let p be a prime such that $p \equiv 1 \pmod{q}$. Let g be a generator of the cyclic subgroup G of order q in $\mathbb{F}_p^*$. Let H be a hash function that takes values in the interval $[0, q-1]$. Each user Alice constructs her keys by selecting a random integer x in the interval $[1, q-1]$ and computing $y = g^x \bmod p$. Alice's public key is y; her private key is x.

Schnorr signature generation. To sign a message m, Alice must do the following:

(1) Select a random integer k in the interval $[1, q-1]$.
(2) Compute $r = g^k \bmod p$, and set $h = H(m, r)$.
(3) Set $s = k + hx \bmod q$.

The signature for the message is the pair of integers (h, s).

Schnorr signature verification. To verify Alice's signature (h, s) on a message m, Bob must do the following:

(1) Obtain an authenticated copy of Alice's public key y.
(2) Verify that h and s are integers in the interval $[0, q-1]$.
(3) Compute $u = g^s y^{-h} \bmod p$ and $v = H(m, u)$.
(4) Accept the signature if and only if $v = h$.

Fischlin shows that, if the hash function is modified in a certain way (without losing collision-resistance), one can obtain a scheme that is susceptible to forgery. He defines a new hash function H depending not only on m and r, but also on the group parameters and the public key: $h = H(p, q, g, y, m, r)$. Here is Fischlin's definition (slightly simplified). For most m and r, the hash value h is defined to be equal to $H_1(m, r) + \frac{q-1}{2}$, where H_1 is a hash function that takes values in $[1, (q-1)/2]$ rather than in $[1, q-1]$. If a certain very unusual relation holds, however, then h is defined to be $H_1(m_1)$, where we split m into two parts $m = m_1 m_2$ with m_2 consisting of the last $[\log_2 p]$ bits of m; here m_2 is interpreted as an element of $\mathbb{F}_p^*$. The condition under which $H(p, q, g, y, m, r) = H_1(m_1)$ is that $r = gy^{-H_1(m_1)} \bmod p$ and also $m_2 = g^r \bmod p$.

Fischlin comments that if H_1 is collision-resistant, then so is the new hash function (for fixed p, q, g, y and varying m, r). But an adversary can easily forge a signature. He picks m_1 at random, sets $r = gy^{-H_1(m_1)} \bmod p$ and $m_2 = g^r \bmod p$. Setting $h = H_1(m_1)$, he obtains the valid signature $(h, 1)$ for the message m.

As Fischlin points out, in the proof in [23] using the generic group model the hash function cannot query the group-operation oracle, and so the above construction does not contradict the result in [23]. Moreover, no real-world hash function would ever incorporate in its definition operations in the group G that are derived from the signature scheme. It is standard cryptographic practice not to intermingle ingredients from different primitives that are used in a protocol. Indeed, it is well known that if one intermingles two parts of a protocol that are individually secure, the result might be insecure. (For a discussion of a similar issue in the context of hybrid-encryption schemes, see §6 of [15], where the uninstantiable random-oracle-model scheme in [1] is critiqued.) Thus, Fischlin's construction does not point to any "pitfall" in the generic group model. Rather, it serves as a useful reminder of the dangers that lurk in pathological constructions that violate standard cryptographic practice.

We next discuss an even more peculiar construction used in an attempt to find a weakness in the generic group model.

5.2. **Dent's weaknesses.** In [12] Dent carries over to the generic group model the type of argument that Canetti, Goldreich and Halevi [9, 10] used in their attempt to undermine the random oracle model. We argued in §6 of [15] that the extremely contrived constructions in [9, 10] and similar papers do not in fact give any reason to lose confidence in the random oracle model. By the same token, if we examine the construction in [12], we see that it in no way shows a "weakness" in the generic group model.

The main content of [12] is contained in §5.2, where the construction is given. Since that subsection is preceded by several pages of formalism that are difficult to read, many potential readers are likely to be deterred from examining the argument. This is unfortunate. In reality, the construction is in essence a simple one.

Dent modifies a signature scheme S that is secure against chosen-message attack in the generic group model to get a new signature scheme S' that is still secure in the generic group model but is insecure with any concrete group. Stripped of the formalism in [12], here is a rough (but reasonably accurate) description of his procedure. Let $\sigma_1, \sigma_2, \ldots$ be an enumeration of all efficiently computable encoding algorithms for families of cyclic groups. (This set is enumerable, because the set of all computer programs is enumerable.) If a message m consists of an integer j followed by the steps in the computation of $\sigma_j^q(j)$ (where σ_j^q is an encoding of a cyclic group of order q), and if $\sigma_j^q(j)$ is equal to $\sigma(j)$, where σ is the encoding used in S, then the S'-signature is the S-signature with the secret key appended. If m is not of this form, then the S'-signature is the S-signature without the secret key appended. In the generic group model, where σ is a random encoding given by an oracle, for any j there is negligible probability that $\sigma_j^q(j) = \sigma(j)$. But with any concrete group with encoding σ_j^q, the chosen-message attacker need only ask Alice to sign a message consisting of j and the steps in the computation of $\sigma_j^q(j)$, and Alice foolishly sends him her secret key.

The notion of a signature scheme in which the signer includes her secret key in the signature for certain types of messages is bizarre, to say the least. A contrived argument of the sort in [9, 10] and [12] has no relation to cryptographic practice and is not of much value in understanding the subtle strengths and weaknesses of an important concept such as the generic group model (or the random oracle model). Rather, constructions of the type described in the last paragraph tend to diminish the credibility of theoretical cryptography, with the unfortunate result that many people in government and industry take a skeptical view of theory.

5.3. **The Stern–Pointcheval–Malone-Lee–Smart flaw.**

We next analyze the criticism in [27] of D. Brown's security proof for the elliptic curve digital signature algorithm (ECDSA) in the generic group model [6, 7]. First we describe the ECDSA.

ECDSA key generation. E is an elliptic curve defined over $\mathbb{F}_p$, and P is a point of prime order q in $E(\mathbb{F}_p)$; these are system-wide parameters. For simplicity, we shall suppose that p is a prime, although the construction can easily be adapted to a prime power p as well. Each user Alice constructs her keys by selecting a random integer x in the interval $[1, q-1]$ and computing $Q = xP$. Alice's public key is Q; her private key is x.

ECDSA signature generation. To sign a message having hash value h, $0 < h < q$, Alice must do the following:

(1) Select a random integer k in the interval $[1, q-1]$.
(2) Compute $kP = (x_1, y_1)$ and set $r = x_1 \bmod q$ (where x_1 is regarded as an integer between 0 and $p-1$). (Note: If $r = 0$, then go back to Step 1 and select another k.)
(3) Compute $k^{-1} \bmod q$ and set $s = k^{-1}(h+xr) \bmod q$. (Note: If $s = 0$, then go back to Step 1.)

The signature for the message is the pair of integers (r, s).

ECDSA signature verification. To verify Alice's signature (r, s) on a message, Bob must do the following:

(1) Obtain an authenticated copy of Alice's public key Q.
(2) Verify that r and s are integers in the interval $[1, q-1]$, and compute the hash value h of the message.
(3) Compute $u_1 = s^{-1}h \bmod q$ and $u_2 = s^{-1}r \bmod q$.
(4) Compute $u_1P + u_2Q = (x_0, y_0)$ and, regarding x_0 as an integer between 0 and $p-1$, set $v = x_0 \bmod q$.
(5) Accept the signature if and only if $v = r$.

Notice that if Alice generated her signature correctly, then $u_1P + u_2Q = (u_1 + xu_2)P = kP$ because $k \equiv s^{-1}(h + xr) \pmod q$, and so $v = r$.

ECDSA was first proposed in 1992 by Vanstone [29] as an elliptic curve analogue of the Digital Signature Algorithm that had been proposed by the National Institute of Standards and Technology. The most important advantage that it has over DSA is the possibility of working over much smaller fields $\mathbb{F}_p$. Indeed, in ECDSA p is of the same order of magnitude as q, whereas in DSA it has to be chosen much larger (because of the availability of index calculus algorithms to find discrete logarithms in $\mathbb{F}_p^*$).

The main difference between DSA and ECDSA is that in Step 2 of DSA signature generation r is defined as the least nonnegative residue modulo q of the group element $g^k \in \mathbb{F}_p^*$, where g is a fixed generator of the order-q subgroup and g^k is regarded as an integer in $[1, p-1]$ and then reduced modulo q.

In [6] (see also [7]) Brown constructed a security reduction for ECDSA that gave the following result: *If the elliptic curve group can be modeled by a generic group and if the hash function satisfies certain reasonable assumptions, then ECDSA is secure against chosen-message attack by an existential forger.* A central role in Brown's proof was played by what he called the "conversion function" $r = f(kP)$ that goes from a group element to an integer mod q in Step 2 of signature generation. In the case of ECDSA this function is "almost invertible" in the sense that, given an arbitrary integer r mod q, with nonnegligible probability one can efficiently find a point whose x-coordinate (regarded as an integer in $[0, p-1]$) reduces to r modulo q.[5] The almost-invertibility property allowed him in essence to transfer the intractability of the discrete log problem on a generic group to the set of r-values.

Brown observed that his proof does not work for DSA because no one knows any reasonable way to find an element in $\mathbb{F}_p^*$ that has order q and reduces modulo q to a given r. In other words, the DSA conversion function $r = f(g^k)$ is apparently "one-way." In the next section we shall discuss possible interpretations of the curious circumstance that ECDSA enjoys a "provable security" property that DSA apparently does not.

Brown's result attracted attention because it was the first security reduction constructed for ECDSA itself, rather than for modified versions of the signature scheme. Because ECDSA is recommended by many standards bodies (IEEE, ISO, ANSI, NIST, etc.), it is important to look carefully at Brown's work.

In [27] the authors criticize Brown's proof harshly. They found what they claim to be a "flaw" in his use of the generic group model. The authors of [27] point out that ECDSA has certain special properties that it would not have if the elliptic curve were a generic group. They give two concrete examples, both coming from the fact that on an elliptic curve given by a Weierstrass equation the points P and $-P$ have the same x-coordinate.

First of all, they observe that if (r, s) is a valid signature for a message m, then so is $(r, q-s)$, because in Steps 3 and 4 of signature verification one obtains $u_1P + u_2Q = (x_0, -y_0)$ if s is replaced by $q - s$.

In the second place, they show that, given two messages with hash values h_1 and h_2, it is possible to choose one's private key x in such a way that both messages have the same signature. Namely, select random k and determine r as in Step 2 of signature generation. Now choose x so that $-xr$ is the average of h_1 and h_2, that is, $x = -(2r)^{-1}(h_1 + h_2) \bmod q$. Then $(h_1 + xr) + (h_2 + xr) \equiv 0 \pmod{q}$, and so the pair (r, s) with $s = k^{-1}(h_1 + xr) \bmod q$ will serve as a signature for both messages.

These two properties of ECDSA — which in [27] are called "malleability" and "signature duplication," respectively — have nothing to do with the standard definition of forging a signature. Thus, the authors of [27] do not claim that these properties contradict the conclusion of Brown's theorem. Nor do they claim that malleability or signature duplication cause problems in any real-world use of ECDSA. Rather, they argue that Brown's use of the generic group model is flawed because the assumption that elliptic curves are generic groups would also imply unmalleability (and presumably also the impossibility of signature duplication).

This argument is based on a misunderstanding of the generic group assumption that perhaps comes from regarding it as very similar to the random

[5] One also has to know that the inversion algorithm finds points that are randomly distributed as r varies.

oracle model for hash functions. In the latter case a hash function should for all practical purposes appear to take random values. But when working with a concrete group one should never think that it is in practice indistinguishable from a generic group. As explained in §4, any such group will fail certain easy randomness tests that a generic group would pass. It is not surprising that contrived cryptographic properties, such as malleability and signature duplication, can be found that highlight the difference between a concrete group and a generic group. However, the authors of [27] have not demonstrated any flaw in Brown's work.

6. Substantive Questions

We now raise some questions that we believe are more substantive than the criticisms discussed in the previous section.

6.1. **Weak implementations.** As mentioned in the Introduction, one of the attractions of elliptic curve cryptography (ECC) is that the group structure on elliptic curves in general is thought to have no special features that could lead to efficient algorithms for finding discrete logarithms. Therefore, it may seem reasonable to model the curve by a generic group when analyzing the security of a system. However, such a statement must be used with caution. For instance, if a supersingular elliptic curve is chosen, then the discrete log problem can be solved in subexponential time, and the generic group assumption is invalid.

Even if a 'cryptographically strong' elliptic curve $E(\mathbb{F}_p)$ is chosen, the details of implementation can sometimes affect the validity of the generic group model. Just as a poorly implemented hash function falls far short of its idealization in the random oracle model, so also a careless implementation of an ECC protocol is likely to have security weaknesses that should not exist under the generic group assumption. We give two examples that are due to Naccache–Smart–Stern [18] and Smart [26].

6.1.1. *Projective coordinates.* Suppose that a public key $Q = xP$ is computed from the secret integer $x = x_t 2^t + \cdots + x_2 2^2 + x_1 2 + x_0$ using the standard left-to-right double-and-add method. Suppose also that in order to improve efficiency of the computation points are computed and stored in a form — for example, in Jacobian coordinates (X, Y, Z) — in which a point has many different equivalent representations. If the point Q is made public immediately after the computation — without first converting it to affine coordinates or replacing (X, Y, Z) by a different $(\lambda^2 X, \lambda^3 Y, \lambda Z)$ — then an attacker has a good chance of being able to learn the first few bits of x.

For example, as pointed out in [18], to find x_0 the attacker need only determine whether or not the Jacobian coordinates $Q = (X, Y, Z)$ were obtained as a result of the Jacobian coordinate doubling formulas. If one tries to find $Q' = (X', Y', Z')$ such that $Q' = 2Q$, then solving for Z' involves taking a fourth root in the field $\mathbb{F}_p$. Suppose, for instance, that $p \equiv 1$

(mod 4). Then only 25% of the nonzero elements of $\mathbb{F}_p$ have fourth roots. If the attacker is unable to solve for Z', she guesses that $x_0 = 1$, and if she is able to take the fourth root, she guesses that $x_0 = 0$. Then whenever $x_0 = 0$ she always guesses correctly, and when $x_0 = 1$ she is correct approximately 75% of the time.

In contrast, in a generic group the representation of an element is unique, and it couldn't possibly reveal whether or not the last step used to arrive at the element was doubling of another element.

6.1.2. *Malleable encryption.* Suppose that an encryption scheme has the property that a ciphertext c can easily be modified to get another valid encryption c' of the same message. Such a system is obviously insecure under chosen-ciphertext attack, because the attacker is allowed to ask the decryption oracle for the decryption of any $c' \neq c$.

Suppose, for instance, that we are using a generalized ElGamal system of the following form. As usual, $Q = xP$ is the public key, and x is the secret key. Let $f_k(m)$ be a symmetric encryption function that depends on a key k, and let $V : E(\mathbb{F}_p) \longrightarrow K$ be a publicly known function from the points on the curve to the key-space of the symmetric encryption function. To encrypt a message m, Bob chooses a random integer ℓ, computes ℓP and ℓQ, and sends the ciphertext $c = (\ell P, f_{V(\ell Q)}(m))$. Alice decrypts by computing $x\ell P = \ell Q$ and hence the key $V(\ell Q)$.[6]

As pointed out in [26], if one chooses a key derivation function V for the symmetric system that depends only on the x-coordinate of the point,[7] then one can get another valid encryption of m by simply replacing the first component of the ciphertext c by $-\ell P$. In contrast, under the generic group assumption, Smart shows that the ECIES system is secure against chosen-ciphertext attack for arbitrary V.[8]

Smart's example of malleable encryption serves as a reminder that the key derivation function should be constructed carefully.

6.2. **How to interpret security reductions?** The questions to ask about a security reduction in the generic group model are the same ones that should be asked about any provable security result (see [15]): How should the result be interpreted? What type of guarantee (if any) does it give us? Among the various versions of a given type of cryptosystem, what security advantages

[6] This is a simplified version of the ECIES system analyzed in [26]. In ECIES, the ciphertext also includes a message authentication code, which, however, is irrelevant to our discussion.

[7] Here we are supposing that the curve is in Weierstrass form.

[8] If one wants to use a function V that depends only on a point's x-coordinate, then a trivial modification will fix this problem. Namely, define the first component of the ciphertext to be just the x-coordinate of ℓP. After all, inclusion of the y-coordinate is superfluous, and is equivalent to appending a random bit to the ciphertext. Any secure encryption scheme will become insecure under chosen-ciphertext attack if one appends a random bit to the ciphertext.

(or disadvantages) do the ones supported by reductionist security arguments have compared to the versions for which no such provable security result is known?

In the case of Brown's theorem on ECDSA in [6], it is particularly interesting to ask what one can conclude from the fact that his proof apparently cannot be made to work for DSA. Our opinion is: not much.

Brown offers two explanations for this phenomenon. The first is that, since the DSA conversion function $r = f(g^k)$ appears to be almost-bijective and one-way (and therefore not almost-invertible), DSA is sufficiently different from ECDSA so that different methods would be required to prove the security of DSA. The second explanation is that this absence of almost-invertibility might indicate an as yet undiscovered security weakness in DSA. He writes:

> Because an almost-invertible function is nearly an identity function — indeed, the identity function is the best example of an almost-invertible function — and an identity function clearly cannot add complexity to the design of a protocol, an almost-invertible function does not add any essential complexity to a protocol. Thus DSA is more complicated than ECDSA because its conversion function departs more from being an identity function. An attack may lie hidden in this complexity until proved otherwise.

This last speculative comment strikes us as implausible. It seems very unlikely that the ability to recover kP from $r = f(kP)$ in ECDSA should make a forger's task more difficult. On the contrary, it is probably more likely that the same feature that enables a researcher to construct a security reduction might also enable an attacker in certain circumstances to construct a security breach.

There are good reasons to use ECDSA rather than DSA. However, the fact that Brown's security reduction doesn't carry over to DSA is not one of them.

6.3. **When are they really a step forward?** When someone expresses skepticism about the type of assurance one gets from a so-called "provable security" result, a common response is: Well, it's better than not having anything. However, a new question arises in connection with protocols and reductionist security arguments that are designed with the purpose of avoiding the use of the random oracle assumption. Namely, is the new security argument more or less reassuring than the one using the random oracle model that it replaces? In particular, how does a security result based on the generic group model compare with a security argument using the random oracle model? We describe an example of such a result and the questions it raises.

In [5] Boneh, Lynn, and Shacham constructed short signatures that they showed to be secure in the random oracle model assuming intractability of the Computational Diffie-Hellman (CDH) problem. Three years later Boneh and Boyen [4] proposed a new variant of the signature scheme, which they designed with the objective of obtaining a provable security result without using the random oracle model. They paid a price for avoiding the random oracle assumption. In the first place, a Boneh-Boyen signature is about twice as long as a Boneh-Lynn-Shacham signature. In the second place, the assumption about CDH is replaced by what Boneh and Boyen call the Strong Diffie-Hellman (SDH) assumption, which has been much less extensively studied than the CDH and is presumably a stronger assumption. In the third place, in order "to gain some confidence" in the intractability of this possibly easier SDH problem, in §5 of [4] they use the generic group assumption to derive a lower bound on its computational complexity.

The m-SDH problem in a group G of prime order q is the problem, given group elements $g, g^x, g^{x^2}, \ldots, g^{x^m}$ (where x is an unknown integer mod q), of constructing a pair (c, h) such that $h^{x+c} = g$ (where c is a nonzero integer mod q and h is a group element). The difficulty of this problem can be shown to be less than or equal to that of the CDH problem (which requires the construction of g^{xy} given g, g^x, and g^y). In §5 of [4] the authors prove that m-SDH in a generic group with a pairing cannot be solved in fewer than (roughly) $\sqrt{q/m}$ operations.

The group G that is used in the type of cryptosystem in [5] and [4] is called a Gap Diffie-Hellman group [5]. It must have an efficiently computable bilinear pairing $e: \ G \times G \rightarrow G_T$.[9] (See [14] for a detailed treatment of pairings.) The existence of such a pairing implies that the Decision Diffie-Hellman (DDH) problem (this is the problem, given g, g^x, g^y, and g^z, of determining whether or not $z \equiv xy \pmod{q}$) is efficiently solvable. Informally speaking, the Gap Diffie-Hellman property means that computational problems such as the DLP and CDH are much harder than the DDH — that is, in the inequalities DDH $\leq$ CDH $\leq$ DLP the first is a strict inequality with a large "gap" in difficulty.

The security of a pairing-based protocol rests on the hope that there's a big gap between the DDH and the problem underlying the protocol (and that, in practice, there is no faster way to solve this underlying problem than to solve the DLP in G (or in G_T)). For the signature scheme in [4] the underlying problem is m-SDH, where m is a bound on the number of signature queries allowed in a chosen-message attack.

The question of a gap between Decision Diffie-Hellman and m-SDH is subtle. It is not even strictly accurate to speak of a "gap" in the usual sense, since in a general group we do not know that DDH $\leq$ m-SDH (in other words, it is not known whether an oracle for m-SDH can be used to

[9] One can work in a more general setting with two groups G_1 and G_2 having a pairing $e: \ G_1 \times G_2 \rightarrow G_T$ and an efficiently computable isomorphism from G_2 to G_1. For simplicity, we are describing the special case $G_1 = G_2$. However, our remarks carry over to the more general setting as well.

efficiently solve DDH). While it may be reasonable to conjecture that m-SDH is hard in the groups G that are used in pairing-based cryptography, the fact that we know m-SDH $\leq$ CDH but do not even know m-SDH $\geq$ DDH should give us pause.

It was because of the shakiness of the m-SDH assumption that the authors of [4] felt the need to give a justification of this assumption using the generic group model. Thus, in order to avoid using the random oracle assumption, they had to resort to the generic group assumption. This is not necessarily bad. However, as we discussed in §4, the random oracle assumption is a reasonably accurate idealization of a concrete hash function. It is interesting that, in order to avoid this assumption, the authors of [4] used a group model that is a much weaker reflection of reality than the hash function model that they mistrusted.

Moreover, a more serious difficulty with the provable security result for the signature scheme in [4] soon came to light. Note that the Boneh-Boyen lower bound $\sqrt{q/m}$ for the difficulty of m-SDH is weaker by a factor of $\sqrt{m}$ than the lower bound $\sqrt{q}$ for the difficulty of CDH in the generic group model. At first it seemed that the factor $\sqrt{m}$ was not a cause for concern, and that the true difficulty of the m-SDH problem was probably $\sqrt{q}$ as in the case of CDH. However, at Eurocrypt 2006 Cheon [11], using the same attack that had been described earlier in a different setting by Brown and Gallant [8], showed that m-SDH can be solved — and in fact the discrete logarithm x can be found — in $\sqrt{q/m_0}$ operations if $m_0 \leq m$ divides $q-1$ and $m_0 < q^{1/3}$.[10]

Thus, based on current knowledge, the true difficulty of the hard problem underlying the signature scheme in [4] in general seems to be less than the difficulty of the problem underlying the earlier scheme in [5]. So it turns out that the price that Boneh and Boyen had to pay to dispense with the random oracle model was steep indeed.

Finally, to end on an upbeat note, we observe that the Brown-Gallant-Cheon attack actually attests to the power of the generic group model. After all, using this model the lower bound that Boneh and Boyen were able to obtain was $\sqrt{q/m}$, not $\sqrt{q}$. Thus, in this case the generic group model was powerful enough to give a "warning" (that went unheeded) about the likelihood of a $\sqrt{m}$-attack on the Strong Diffie-Hellman problem. In view of all our cautionary comments in §4, we find it surprising that the generic group model was capable of "predicting" the $\sqrt{m}$-attack. We would not have expected this.

[10] To minimize the impact of the attack one could add to the protocol in [4] the condition that q be of the form $2p_1+1$ with p_1 prime. However, in [11] an attack is also given when $q+1$, rather than $q-1$, is divisible by m_0. Thus, one would probably want to add the condition that q be both of the form $2p_1+1$ and $12p_2-1$ with p_1 and p_2 prime. In any case, it is important to note that the need for such a condition was not anticipated when the protocol in [4] was proposed, and that no such condition is needed for the earlier protocol in [5].

Both the random oracle and generic group models are idealizations — mathematical abstractions that many people distrust and disparage because they seem to poorly reflect the messy world of practical cryptography. Some doubt that such theoretical constructs can be reliable and useful for real-world cryptography. Yet both models have held up pretty well — sometimes, as we have seen, better than we had a right to expect. As Eugene Wigner [30] famously remarked, "...the enormous usefulness of mathematics in the natural sciences is something bordering on the mysterious and...there is no rational explanation for it."

7. Conclusion

Here is a summary of our conclusions that differ from those of earlier authors:

(1) When we investigate how it was possible for the security proof of RSA-S1 using the generic group assumption to be followed soon after by a successful attack on RSA-S1, we see that the explanation is not that there is something wrong with the generic group model, but rather that the proof itself contained a fatal flaw.

(2) When the generic group model (or the random oracle model) fails in a contrived setting that violates standard cryptographic practice, that does not indicate any weakness in the model.

(3) When applied to discrete-log protocols, the generic group model is more limited than the random oracle model in the sense that it is supposed to be an accurate reflection of only one aspect of the concrete group, namely, absence of special properties related to finding discrete logs.

(4) Just as a poorly implemented hash function falls far short of its idealization in the random oracle model, so also a careless implementation of an ECC protocol is likely to have security weaknesses that should not exist under the generic group assumption.

(5) Despite the claims in [27], no flaw has been found in D. Brown's use of the generic group model to study the security of ECDSA.

(6) The fact that Brown's security result does not carry over to DSA or to certain modified versions of ECDSA does not by itself imply that ECDSA is more secure than those other variants.

(7) At present there is no good reason to prefer a variant of a protocol that is supported by a security argument using the generic group assumption over a variant that is supported by an argument using the random oracle assumption. Moreover, if a protocol based on the generic group assumption relies on a less widely studied version of the underlying hard problem than a similar protocol based on the random oracle assumption, then there might well be a loss of true real-world security in using the former rather than the latter.

Finally, we wish to emphasize that security proofs in the generic group model — or any security proofs for that matter — do not relieve us of the obligation to examine the details of implementation very carefully for pitfalls that are not anticipated in the model. Despite the current popularity of formal reductionist security arguments, the need for old-fashioned testing, practical analysis, and common sense is as great as ever.

Acknowledgments

We would like to thank Dan Brown for his valuable comments on an earlier draft of the paper, Nigel Smart for bringing papers [18] and [26] to our attention, and Alex Dent for sharing his points of view. Needless to say, all the opinions expressed in this article are the sole responsibility of the authors.

References

[1] M Bellare, A. Boldyreva, and A. Palacio, An uninstantiable random-oracle-model scheme for a hybrid-encryption problem, *Advances in Cryptology – Eurocrypt 2004*, LNCS 3027, Springer-Verlag, 2004, pp. 171-188.

[2] M. Bellare and P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols, *Proc. First Annual Conf. Computer and Communications Security*, ACM, 1993, pp. 62-73.

[3] M. Bellare and P. Rogaway, Optimal asymmetric encryption — how to encrypt with RSA, *Advances in Cryptology – Eurocrypt '94*, LNCS 950, Springer-Verlag, 1994, pp. 92-111.

[4] D. Boneh and X. Boyen, Short signatures without random oracles, *Advances in Cryptology – Eurocrypt 2004*, LNCS 3027, Springer-Verlag, 2004, pp. 56-73.

[5] D. Boneh, B. Lynn, and H. Shacham, Short signatures from the Weil pairing, *Advances in Cryptology – Asiacrypt 2001*, LNCS 2248, Springer-Verlag, 2001, pp. 514-532.

[6] D. Brown, Generic groups, collision resistance, and ECDSA, *Designs, Codes and Cryptography*, **35** (2005), 119-152.

[7] D. Brown, On the provable security of ECDSA, in I. Blake, G. Seroussi, and N. Smart, eds., *Advances in Elliptic Curve Cryptography*, Cambridge University Press, 2005, pp. 21-40.

[8] D. Brown and R. Gallant, The static Diffie-Hellman problem, http://eprint.iacr.org/2004/306.

[9] R. Canetti, O. Goldreich, and S. Halevi, The random oracle model, revisited, *Proc. 30th Annual Symp. Theory of Computing*, ACM, 1998, pp. 209-218.

[10] R. Canetti, O. Goldreich, and S. Halevi, The random oracle model, revisited, http://eprint.iacr.org/1998/011.

[11] J. Cheon, Security analysis of the Strong Diffie-Hellman problem, *Advances in Cryptology – Eurocrypt 2006*, LNCS 4004, Springer-Verlag, 2006, pp. 1-11.

[12] A. Dent, Adapting the weaknesses of the random oracle model to the generic group model, *Advances in Cryptology – Asiacrypt 2002*, LNCS 2501, Springer-Verlag, 2002, pp. 100-109.

[13] M. Fischlin, A note on security proofs in the generic model, *Advances in Cryptology – Asiacrypt 2000*, LNCS 1976, Springer-Verlag, 2000, pp. 458-469.

[14] S. Galbraith, Pairings, Ch. IX of I. F. Blake, G. Seroussi, and N. P. Smart, eds., *Advances in Elliptic Curve Cryptography*, Vol. 2, Cambridge University Press, 2005.

[15] N. Koblitz and A. Menezes, Another look at "provable security," to appear in *Journal of Cryptology*.

[16] T. Matsumoto, K. Kato, and H. Imai, Speeding up secret computations with insecure auxiliary devices, *Advances in Cryptology – Crypto '88*, LNCS 403, Springer-Verlag, 1990, pp. 497-506.

[17] J. Merkle and R. Werchner, On the security of server-aided RSA protocols, *Proceedings PKC '98*, LNCS 1431, Springer-Verlag, 1998, pp. 99-116.

[18] D. Naccache, N. Smart, and J. Stern, Projective coordinates leak, *Advances in Cryptology – Eurocrypt 2004*, LNCS 3027, Springer-Verlag, 2004, pp. 257-267.

[19] V. I. Nechaev, Complexity of a deterministic algorithm for the discrete logarithm, *Mathematical Notes*, **55** (2) (1994), pp. 165-172.

[20] P. Nguyen and I. Shparlinski, On the insecurity of a server-aided RSA protocol, *Advances in Cryptology – Asiacrypt 2001*, LNCS 2248, Springer-Verlag, 2001, pp. 21-35.

[21] B. Pfitzmann and M. Waidner, Attacks on protocols for server-aided RSA computation, *Advances in Cryptology – Eurocrypt '92*, LNCS 658, Springer-Verlag, 1993, pp. 153-162.

[22] C. Schnorr, Efficient signature generation for smart cards, *Journal of Cryptology*, **4** (1991), pp. 161-174.

[23] C. Schnorr and M. Jakobsson, Security of signed ElGamal encryption, *Advances in Cryptology – Asiacrypt 2000*, LNCS 1976, Springer-Verlag, 2000, pp. 73-89.

[24] V. Shoup, Lower bounds for discrete logarithms and related problems, *Advances in Cryptology – Eurocrypt '97*, LNCS 1233, Springer-Verlag, 1997, pp. 256-266.

[25] V. Shoup, OAEP reconsidered, *Advances in Cryptology – Crypto 2001*, LNCS 2139, Springer-Verlag, 2001, pp. 239-259.

[26] N. Smart, The exact security of ECIES in the generic group model, *Cryptography and Coding*, LNCS 2260, Springer-Verlag, 2001, pp. 73-84.

[27] J. Stern, D. Pointcheval, J. Malone-Lee, and N. Smart, Flaws in applying proof methodologies to signature schemes, *Advances in Cryptology – Crypto 2002*, LNCS 2442, Springer-Verlag, 2002, pp. 93-110.

[28] D. Stinson, *Cryptography: Theory and Practice*, 2nd ed., CRC Press, 2002.

[29] S. A. Vanstone, Responses to NIST's proposal, *Communications of the ACM*, **35** (1992), pp. 50-52.

[30] E. P. Wigner, The unreasonable effectiveness of mathematics in the natural sciences, *Comm. Pure Appl. Math.*, **13** (1960), pp. 1-14.

Department of Mathematics, Box 354350, University of Washington, Seattle, WA 98195 U.S.A.

E-mail address: `koblitz@math.washington.edu`

Department of Combinatorics & Optimization, University of Waterloo, Waterloo, Ontario N2L 3G1 Canada

E-mail address: `ajmeneze@uwaterloo.ca`