Accelerated Verification of ECDSA Signatures

Adrian Antipa[1], Daniel Brown[1], Robert Gallant[1], Rob Lambert[1], René Struik[1], and Scott Vanstone[2]

[1] Certicom Research, Canada
{aantipa,dbrown,rgallant,rlambert,rstruik}@certicom.com
[2] Dept. of Combinatorics and Optimization, University of Waterloo, Canada
savansto@uwaterloo.ca

Abstract. Verification of ECDSA signatures is considerably slower than generation of ECDSA signatures. This paper describes a method that can be used to accelerate verification of ECDSA signatures by more than 40% with virtually no added implementation complexity. The method can also be used to accelerate verification for other ElGamal-like signature algorithms, including DSA.

1 Introduction

The elliptic curve digital signature algorithm (ECDSA) [1, 3, 7] is a widely standardized variant of the original ElGamal signature scheme. As is the case with most ElGamal signature schemes, ECDSA has the property that signature verification is about twice as slow as signature generation (and many times slower if the signer is afforded the luxury of precomputation). The opposite is true in the RSA signature scheme with small encryption exponent e, where signature verification is many times faster than generation. Thus speeding up ECDSA signature verification is a problem of considerable practical importance.

This paper describes a new method that can be used to accelerate signature verification for ECDSA and related signature schemes. The method is quite simple, is independent of the techniques employed for field arithmetic and elliptic curve arithmetic, and requires very little additional resources, such as memory or code space. The advantages of the new method are most apparent for ECDSA*, a slightly modified version of ECDSA. However, the advantages can also be enjoyed if the signer appends a small number of bits ("side information") to standardized ECDSA signatures. We emphasize that, other than this extra information, the new method does not require any changes to the standardized specifications of ECDSA. Thus, the extended signatures are still conformant with the existing ECDSA standards.

In the most favorable setting, if one uses an elliptic curve of prime order over a prime field, only 1 or 2 bits of side information are needed to accelerate signature verification by about 40%.

The remainder of the paper is organized as follows. In §2 we describe ECDSA* and show that its security is equivalent to that of ECDSA. Some relevant mathematical background is collected in §3. The new verification methods for ECDSA* and ECDSA are presented and analyzed in §4. Summary conclusions appear in §5.

2 Modified DSA and ECDSA

We next present a modification of DSA and ECDSA in the general setting of a cyclic group.

1. *System-wide parameters.* Let $\mathbb{G}$ be a (cyclic) group of prime order n with generator G and identity element $\mathcal{O}$. Let $f : \mathbb{G} \to \mathbb{Z}_n$ be a suitable conversion function. Let $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_n$ be a collision-resistant hash function.
2. *Initial set-up.* Each communicating party A selects a random integer $d \in [1, n-1]$ and publishes its public key $Q = dG$. The parameter d is kept private to A.
3. *Signature generation.*
   Input: Message $m \in \{0,1\}^*$, private key d.
   Output: Signature (R, s).
   Actions:
   (a) Select a random integer $k \in [1, n-1]$. Compute $R := kG$ and $r := f(R)$.
   (b) Compute $s \in \mathbb{Z}_n$ from the equation $\mathcal{H}(m) \equiv sk - dr \ (\mathbf{mod}\ n)$.
   (c) If $r = 0$ or $s = 0$, go to Step 3a; otherwise, return (R, s).
4. *Signature verification.*
   Input: Signature (R, s), message $m \in \{0,1\}^*$, public key $Q \in \mathbb{G}$ associated with entity A.
   Output: Acceptance or rejection of signature as originating from A.
   Actions:
   (a) Compute $r := f(R)$.
   (b) Verify that r and s are integers in the interval $[1, n-1]$. If any verification fails, return `'reject signature'`.
   (c) Verify that $R = s^{-1}(eG + rQ)$, where $e := \mathcal{H}(m)$. If verification fails, return `'reject signature'`; otherwise, return `'accept signature'`.

Note 1. The function f is usually defined over a superset of $\mathbb{G}$. If so, one can usually evaluate $f(R)$ without explicitly checking that $R \in \mathbb{G}$ first, since if the verification equation holds and if $s \in \mathbb{Z}_n^*$, then $R \in \mathbb{G}$ (since $G, Q \in \mathbb{G}$).

Observe that if the signature verification equation holds, then $s^{-1}(eG+rQ) = R$ and, in particular, $f(s^{-1}(eG + rQ)) = f(R) = r$. The *original scheme* is the one with signature (r, s) and verification equation $f(s^{-1}(eG + rQ)) = r$, rather than the corresponding quantities (R, s) and $s^{-1}(eG + rQ) = R$ in the modified scheme.

It is easy to see that, in either scheme, a signature σ obtained from the signature generation algorithm with input (m, d) is always accepted by the corresponding signature verification algorithm with input (σ, m, Q), where $Q = dG$.

The following result shows that the original and the modified schemes are equally secure.

Theorem 2. *Consider the original and the modified signature schemes. The following statements are equivalent:*

1. *(r, s) is a valid signature with respect to (m, Q) in the original signature scheme;*
2. *(R, s) is a valid signature with respect to (m, Q) in the modified signature scheme for precisely one point R in the set $\varphi(r) := \{X \in \mathbb{G} \mid f(X) = r\}$.*

Moreover, one can efficiently convert a valid signature in either scheme to a valid one in the corresponding scheme. Thus, the original and the modified schemes are equally secure.

Proof. Let $R := s^{-1}(eG + rQ)$ for some $s \in \mathbb{Z}_n^*$. One has $R \in \mathbb{G}$, since $G, Q \in \mathbb{G}$. It follows that $R \in \varphi(r)$ if and only if $f(R) = r$. The result now follows by comparing the conditions under which either scheme accepts signatures. $\square$

We will exploit the relationship between a signature scheme and its modified scheme in the remainder of the paper. In particular, we are interested in specific instantiations that correspond to DSA and ECDSA.

*DSA and DSA**

The DSA* scheme is an instantiation of the modified signature scheme with the following system-wide parameters: $\mathbb{G} \subseteq \mathbb{Z}_p^*$ is a subgroup of prime order n of the multiplicative group $\mathbb{Z}_p^*$, where p is a suitably chosen prime number, and $|\mathbb{Z}_p^*| = p - 1 = nh$. The conversion function $f : \mathbb{Z}_p \to \mathbb{Z}_n$ is defined by $f(x) := ((x \textbf{ mod } p) \textbf{ mod } n)$. We define the hash function $\mathcal{H}$ by $\mathcal{H}(x) :=$ SHA-1(x) $(\textbf{mod } n)$. For these parameters, the original scheme corresponds to the *Digital Signature Algorithm* (DSA) as standardized in [3].

*ECDSA and ECDSA**

The ECDSA* scheme is an instantiation of the modified signature scheme with the following system-wide parameters: $\mathbb{G} \subseteq E(\mathbb{F}_q)$ is a subgroup of prime order n of some suitably chosen elliptic curve group $E(\mathbb{F}_q)$, $\mathbb{F}_q$ is a finite field, and $|E(\mathbb{F}_q)| = nh$. (We assume h is small, so $n \approx q$.) The conversion function $f : E(\mathbb{F}_q) \to \mathbb{Z}_n$ is defined by $f(x, y) := \overline{x}$ $(\textbf{mod } n)$, where (x, y) is an elliptic curve point in affine representation and where $\overline{x}$ is the standard[3] integer representation of the field element $x \in \mathbb{F}_q$. We define the hash function $\mathcal{H}$ by $\mathcal{H}(x) :=$ SHA-1(x) $(\textbf{mod } n)$. For these parameters, the original scheme corresponds to the *Elliptic Curve Digital Signature Algorithm* (ECDSA) as standardized in [1, 3].

[3] see, e.g., [1]

3 Mathematical Preliminaries

In this section, we introduce some well-known results from number theory that facilitate the exposition in the rest of the paper.

Definition 3. *(Simultaneous Diophantine Approximation Problem)*
Let $\xi_1, \ldots, \xi_\ell \in \mathbb{R}$, let $\varepsilon > 0$, and let B be a positive integer. Find integers $p_1, \ldots, p_\ell, q$ with $q \in [1, B]$ such that

$$\left|\xi_i - \frac{p_i}{q}\right| \leq \frac{\varepsilon}{q} \quad \textit{for all } i, 1 \leq i \leq \ell. \tag{1}$$

This problem has a solution with $\varepsilon \geq B^{-1/\ell}$, as was shown by Dirichlet [2] based on an argument involving the pigeon-hole principle.

Theorem 4. *[2], [6, Theorem 200] Let $\xi_1, \ldots, \xi_\ell \in \mathbb{R}$ and let B be a positive integer. Then the system of inequalities*

$$\left|\xi_i - \frac{p_i}{q}\right| < \frac{1}{qB} \quad \textit{for all } i, 1 \leq i \leq \ell$$

has an integer solution with $p_1, \ldots, p_\ell, q \in \mathbb{Z}$ and $q \in [1, B^\ell]$.

We are mainly interested in the following corollary, which states that integers in $\mathbb{Z}_n$ can be written as rational numbers involving integers that are significantly smaller (in absolute value) than n.

Corollary 5. *Let $\alpha_1, \ldots, \alpha_\ell \in \mathbb{Z}$ and let B be a positive integer. Then the system of congruence relationships*

$$v(\alpha_1, \ldots, \alpha_\ell) \equiv (u_1, \ldots, u_\ell) \pmod{n}$$

has an integer solution with $|u_1|, \ldots, |u_\ell| < n/B$ and $v \in [1, B^\ell]$.

Proof. We apply Theorem 4 with $\xi_i := \alpha_i/n$. Let $p_1, \ldots, p_\ell, q$ be integers with $q \in [1, B^\ell]$, such that

$$\left|\frac{\alpha_i}{n} - \frac{p_i}{q}\right| < \frac{1}{qB} \quad \text{for all } i, 1 \leq i \leq \ell.$$

Consequently, one has

$$|q\alpha_i - p_i n| < n/B \quad \text{for all } i, 1 \leq i \leq \ell.$$

Now, define $u_i := q\alpha_i - p_i n$ and $v := q$. Since $v\alpha_i \equiv u_i \pmod{n}$, the result follows. □

We give some examples that turn out to be useful later on. All examples assume that n is a prime number and that $B^\ell < n$, so as to ensure that v is invertible modulo n. We will ignore rounding issues involving the parameter B, since these obscure the exposition and are irrelevant in our applications (which use very large integers n).

Example 6. ($\ell = 1$) Let n be a prime number. Any integer $x \in \mathbb{Z}_n$ can be written as $x \equiv u/v \ (\mathbf{mod}\ n)$, where u and v are integers such that $|u| < n^{1-\varepsilon}$ and $1 \le v \le n^{\varepsilon}$ (take $B := n^{\varepsilon}$ with $0 < \varepsilon < 1$). In particular, one may have $|u| < \sqrt{n}$ and $1 \le v \le \sqrt{n}$ (take $\varepsilon := \frac{1}{2}$) or $|u| < n^{2/3}$ and $1 \le v \le \sqrt[3]{n}$ (take $\varepsilon := 1/3$).

This example can be generalized as follows.

Example 7. ($\ell \ge 1$) Let n be a prime number. Any integers $x_1, \ldots, x_\ell \in \mathbb{Z}_n$ can be written as $x_i \equiv u_i/v \ (\mathbf{mod}\ n)$ $(1 \le i \le \ell)$, where $u_1, \ldots, u_\ell$, and v are integers such that $|u_1|, \ldots, |u_\ell| < n^{1-\varepsilon}$ and $1 \le v \le n^{\ell\varepsilon}$ (take $B := n^{\varepsilon}$ with $0 < \varepsilon < 1/\ell$). In particular, one may have $|u_1|, \ldots, |u_\ell| < n^{\ell/(\ell+1)}$ and $1 \le v \le n^{\ell/(\ell+1)}$ (take $\varepsilon := 1/(\ell+1)$) or $|u_1|, \ldots, |u_\ell| < n^{2\ell/(2\ell+1)}$ and $1 \le v \le n^{\ell/(2\ell+1)}$ (take $\varepsilon := 1/(2\ell+1)$).

It is well-known that the problem of finding good solutions to the simultaneous Diophantine approximation problem is equivalent to the problem of finding 'short' vectors in a particular lattice. We make this statement more precise and (for completeness) provide a short proof.

Theorem 8. *[8] Let $\xi_1, \ldots, \xi_\ell \in \mathbb{R}$, let $0 < \varepsilon < 1$, and let B be a positive integer. Consider the lattice $\mathcal{L}(A) = \{\mathbf{x}A \mid \mathbf{x} \in \mathbb{Z}^{\ell+1}\}$ that is generated by the rows of the matrix A defined by*

$$A = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \\ \xi_1 & \xi_2 & \cdots & \xi_\ell & \varepsilon/B \end{pmatrix}$$

The lattice $\mathcal{L}(A)$ has a nonzero vector $\mathbf{y}$ with (small) norm $||\mathbf{y}||_\infty \le \varepsilon$ if and only if the simultaneous Diophantine approximation problem defined by Equation (1) has an integer solution $p_1, \ldots, p_\ell, q \in \mathbb{Z}$ with $q \in [1, B]$.

Proof. Let $p_1, \ldots, p_\ell, q$ be integers, with $q \in [1, B]$, such that the system of inequalities defined by Equation (1) is satisfied. Then $\mathbf{y} = \mathbf{x}A$, with $\mathbf{x} := (-p_1, \ldots, -p_\ell, q)$, has norm $||\mathbf{y}||_\infty \le \varepsilon$, since $\mathbf{y} = (q\xi_1 - p_1, \ldots, q\xi_\ell - p_\ell, q\varepsilon/B)$. Conversely, if $\mathbf{y} = (q\xi_1 - p_1, \ldots, q\xi_\ell - p_\ell, q\varepsilon/B)$ has norm $||\mathbf{y}||_\infty \le \varepsilon$, then $|q| \in [1, B]$ and $p_1, \ldots, p_\ell, q$ (or their negatives, if $q < 0$) are a solution to the system of inequalities defined by Equation (1). $\square$

Thus, one can use any algorithm that finds 'short' vectors in lattices to find good solutions to the simultaneous diophantine approximation problem.

Although the L^3 lattice basis reduction algorithm is commonly used to find short vectors, our ultimate goal is to improve the time to verify a signature, and the L^3 algorithm may be too cumbersome to meet this goal. Later, we wish to write an element $x \in \mathbb{Z}_n$ as $x \equiv u/v \ (\textbf{mod}\ n)$, where u and v are integers such that $|u| < \sqrt{n}$ and $|v| < \sqrt{n}$, as in Example 6. We outline how to use the Euclidean algorithm directly to find u and v, and refer the reader to [4] for more details.

When the Extended Euclidean Algorithm is used to find the greatest common divisor of n and x (which is 1, since n is prime), the algorithm produces a sequence of equations

$$s_i n + t_i x = r_i \text{ for } i = 0, 1, 2, \ldots, \tag{2}$$

where $s_0 = 1$, $t_0 = 0$, $r_0 = n$, $s_1 = 0$, $t_1 = 1$, $r_1 = x$, and $r_i \geq 0$ for all i. At each step of the algorithm, the value of $|r_i|$ decreases, and the extended Euclidean algorithm stops when this value becomes 1. At that iteration i of the algorithm when r_i first becomes less than $\sqrt{n}$, it can be shown that very likely the value of $|t_i|$ is also close to $\sqrt{n}$. Choosing $u = r_i$ and $v = t_i$, we see from (2) that $x \equiv u/v \ (\textbf{mod}\ n)$, and so we have found appropriate values of u and v.

The procedure above produces integers u and v with $|u|, |v| \sim \sqrt{n}$ such that $x \equiv u/v \ (\textbf{mod}\ n)$. It is easy to see that the same procedure can be used to write $x \equiv u/v \ (\textbf{mod}\ n)$ with $|u| \sim n^{2/3}$ and $|v| \sim n^{1/3}$ or, more generally, with $|u| \sim n^{1-\varepsilon}$ and $|v| \sim n^{\varepsilon}$ (where $0 < \varepsilon < 1$). Thus, all representations of an integer $x \in \mathbb{Z}_n$ described in Example 6 can be realized using the extended Euclidean algorithm. This procedure can be generalized to produce simultaneous representations of integers $x_1, \ldots, x_\ell \in \mathbb{Z}_n$ in this format, as described in Example 7, using lattice basis reduction techniques for $(\ell+1)$-dimensional lattices. This can be done quite efficiently for $\ell = 1, 2, 3$ using, e.g., the techniques described in [9].

4 Fast Verification of ElGamal-Like Signatures

In this section, we describe a method for considerably speeding up the verification of signatures in ElGamal-like signature schemes, as described in §2. We then exploit the relationship between these signature schemes and their modified schemes to explore speed-ups of signature verifications in the latter. Although our method applies quite generally, we are mainly interested in the concrete speed-ups of signature verifications in ECDSA* (§4.1) and ECDSA (§4.2).

For the modified signature schemes, checking a signature (R, s) over some message m relative to public key Q involves checking that $s^{-1}(eG + rQ) = R$, where $e := \mathcal{H}(m)$. Here, the point G is a system-wide parameter and can be considered fixed, while the points Q and R cannot be considered fixed, since these vary according to signer and message. The main idea of the paper is that one may replace this equation by any (nonzero) scalar multiple hereof and evaluate the transformed equation instead. Thus, one may replace the original signature verification equation by

$$(ves^{-1})G + (vrs^{-1})Q - vR = \mathcal{O} \tag{3}$$

for any $v \in \mathbb{Z}_n^*$. In particular, if one chooses v such that both v and $u := vrs^{-1}(\mathbf{mod}\ n)$ are integers that are significantly smaller than n, such as having only half the bit-length of n, one can often considerably speed-up the computation of the left-hand side of this equation and, thereby, the verification itself. The potential acceleration is suggested by the observation that for most groups $\mathbb{G}$ of practical interest to us, the cost of computing a multiple kP of some unknown point $P \in \mathbb{G}$ is proportional to the bit-size of the scalar multiple k involved. If P is a fixed point and some storage is available, one can accelerate the point multiplication kP by precomputing some data that depends solely on P and storing this for subsequent use. If P is an unknown point, similar acceleration techniques can be used, but are less effective, since one cannot amortize their cost over time. A similar observation can be made regarding the cost of multiple point multiplications, where one computes a linear combination of points in $\mathbb{G}$ and can accelerate the computation by carrying out common elements hereof only once, rather than for several points separately. Also here, the cost of computing a multiple point multiplication usually depends on the maximum bit-size over all scalar multiples corresponding to unknown points. This suggests that one can indeed considerably accelerate the verification of Equation (3) by choosing $v \in \mathbb{Z}_n^*$ such that the scalars $u := vrs^{-1}(\mathbf{mod}\ n)$ and v corresponding to variable points Q and $-R$, respectively, both have relatively small bit-size.

One can find a suitable value for v by writing $x := r/s\ (\mathbf{mod}\ n)$ as $x \equiv u/v\ (\mathbf{mod}\ n)$, where u and v are integers of approximately half the bit-size of n or, more generally, of $1-\varepsilon$ and ε times the bit-size of n, respectively, where $0 < \varepsilon < 1$ (see Example 6). For efficient algorithms for computing u and v of this form, see §3.

We now compare in more detail the cost of checking the original signature verification equation with that of checking the transformed equation discussed above, to give some more insight into the potential acceleration. We then give some concrete speed-ups for ECDSA* and ECDSA in §4.1 and §4.2.

We denote by t the bit-length of n, i.e., $t := \lceil \log_2(n+1) \rceil$, where n is the order of the group $\mathbb{G}$. For convenience, we denote $k \sim L$ iff $|k| \lesssim L$. We assume that sufficient storage is available to store a few points of $\mathbb{G}$.

We distinguish two cases, depending on whether a (nontrivial) fixed multiple of the public key Q is available. This multiple of Q may be made available to the verifier by including it in the signer's certificate for Q.

Case 1: Only one point multiple of Q available (Q itself)

One can write the original signature verification equation in the following form:

$$\lambda G + \mu Q = R, \text{ where } \lambda, \mu \sim n, \tag{4}$$

Similarly, one can write the transformed signature verification equation in the following form:

$$\lambda_0 G_0 + \lambda_1 G_1 + uQ - vR = \mathcal{O}, \text{ where } \lambda_0, \lambda_1, u, v \sim \sqrt{n}, \tag{5}$$

where $G_0 := G$ and where $G_1 := 2^{\lceil t/2 \rceil} G$ is a precomputed multiple of G.

Notice that the transformed equation involves 4 half-size point multiplications, whereas the original equation involves 2 full-size point multiplications.

Case 2: Two point multiples of Q available (Q itself and another multiple of this point)

One can write the original signature verification equation in the following form:

$$\lambda_0 G_0 + \lambda_1 G_1 + \mu_0 Q_0 + \mu_1 Q_1 = R, \text{ where } \lambda_0, \lambda_1, \mu_0, \mu_1 \sim \sqrt{n}, \tag{6}$$

where $G_0 := G$, where $G_1 := 2^{\lceil t/2 \rceil} G$ is a precomputed multiple of G, where $Q_0 := Q$, and where $Q_1 := 2^{\lceil t/2 \rceil} Q$ is a precomputed multiple of Q.
Similarly, one can write the transformed signature verification equation in the following form:

$$\lambda_0 G_0 + \lambda_1 G_1 + \lambda_2 G_2 + u_0 Q_0 + u_1 Q_1 - vR = \mathcal{O}, \text{ where } \lambda_0, \lambda_1, \lambda_2, u_0, u_1, v \sim \sqrt[3]{n}, \tag{7}$$

where $G_0 := G$, where $G_1 := 2^{\lceil t/3 \rceil} G$ and $G_2 := 2^{2\lceil t/3 \rceil} G$ are precomputed multiples of G, where $Q_0 := Q$, and where $Q_1 := 2^{\lceil t/3 \rceil} Q$ is a precomputed multiple of Q.

Notice that the transformed equation involves 6 third-size point multiplications, whereas the original equation involves 4 half-size point multiplications.

4.1 Fast Verification of ECDSA* Signatures

In the previous section, we obtained a potential acceleration of signature verifications for a family of ElGamal-like signature schemes. Here, we explore the concrete speed-up for ECDSA* .

From the case analysis in the previous section, it follows that the transformation of the signature verification equation significantly reduces the size of the scalar multiples involved in this equation (if one can store a single precomputed multiple of the generator of $\mathbb{G}$). As a result, we might expect a considerable speed-up of signature verifications, since this eliminates a large number of point doubling operations, a common element in multiple point multiplications. We expect significant improvements, both for prime curves and for random binary curves. For Koblitz curves, we can only expect a marginal improvement, since for those curves the Frobenius map (which assumes the role of point doubling) comes almost for free.

A precise analysis is difficult to give, due to the large number of point multiplication methods available. We compare the cost of checking the original signature verification with that of checking the transformed equation by combining the Non-Adjacent Form (NAF) representation for scalars with the multiple point multiplication method, and using the Joint Sparse Form (JSF); for details, see [10], [5, Chapter 3, §3.3.3]. We distinguish the same two cases as in the previous section.

We adopt the following notation: By A and D, we denote the cost of a single point addition and point doubling operation, respectively. By m, we denote the bit-length of finite field elements in $\mathbb{F}_q$, i.e., $m := \lceil \log_2 q \rceil$. We assume the elliptic curve $E(\mathbb{F}_q)$ to have a small co-factor h, so that $m \approx \lceil \log_2 n \rceil$. As before, we denote $k \sim L$ iff $|k| \lesssim L$.

Case 1: Only one point multiple of Q available (Q itself)

Consider the original signature verification equation in the format of Equation (4) and represent λ and μ in JSF. Compute the left-hand side of this equation via multiple point multiplication. Since $\lambda, \mu \sim n$, this gives a running time of approximately $(m/2+2)A+mD$ operations. Similarly, consider the transformed signature verification equation in the format of Equation (5) and represent λ_0 and λ_1, resp. u and v, in JSF. Compute the left-hand side of this equation via multiple point multiplication. Since $\lambda_0, \lambda_1, u, v \sim \sqrt{n}$, this gives a running time of approximately $(m/2+4)A+(m/2)D = (2 \cdot m/4+4)A+(m/2)D$ operations[4].

Case 2: Two point multiples of Q available (Q itself and another multiple of this point)

Consider the original signature verification equation in the format of Equation (6) and represent λ_0 and λ_1, resp. μ_0 and μ_1, in JSF. Compute the left-hand side of this equation via multiple point multiplication. Since $\lambda_0, \lambda_1, \mu_0, \mu_1 \sim \sqrt{n}$, this gives a running time of approximately $(m/2+4)A+(m/2)D = (2 \cdot m/4+4)A+(m/2)D$ operations. Similarly, consider the transformed signature verification equation in the format of Equation (7) and represent λ_0 and λ_1, λ_2 and u_0, resp. u_1 and v, in JSF. Compute the left-hand side of this equation via multiple point multiplication. Since $\lambda_0, \lambda_1, \lambda_2, u_0, u_1, v \sim \sqrt[3]{n}$, this gives a running time of approximately $(m/2+6)A+(m/3)D = (3 \cdot m/6+6)A+(m/3)D$ operations.

Rough analysis for P-384 curve
We provide a rough analysis of the relative efficiency of the ECDSA* verification procedure described in this paper compared to the traditional procedure for ECDSA verification. Our analysis is based on the elliptic curve curve P-384

[4] Here, 4 point additions account for computing $G_0 \pm G_1$ and $Q_0 \pm Q_1$, which is necessary for using the 2 JSFs.

defined over a 384-bit prime field and specified by NIST [3]. All NIST prime curves have co-factor $h = 1$, so by Hasse's theorem, the bit-size of the order of the cyclic group $\mathbb{G}$ is approximately $m = 384$. We consider each of the scenarios described under Case 1 and Case 2 above. We ignore the relatively minor cost of running the Extended Euclidean Algorithm to compute the half-size integers u and v.

We assume the points to be represented using Jacobian coordinates and that the cost S of a squaring in $\mathbb{Z}_q$ is slightly less than the cost M of a multiplication (we take $S \approx 0.8M$). With Jacobian coordinates, one has $A \approx 8M + 3S$ and $D \approx 4M + 4S$ (see [5, Table 3.3]). Substitution of $S \approx 0.8M$ now yields $A \approx 10.4M$ and $D \approx 7.2M$ and, hence, $D/A \approx 0.69$.

If the verifier has access only to the public key Q of the signer and not to a multiple hereof (Case 1), the cost of verifying the signature using the traditional verification method is roughly $194A + 384D \approx 459A$, while the corresponding figure for the new method is $196A + 192D \approx 328A$. As a result, the new method yields a speed-up of 40% over the old method.

Similarly, if the verifier has access to the public key Q of the signer and a suitable multiple hereof (Case 2), the corresponding figures are roughly $196A + 192D \approx 328A$ for the traditional method and $198A + 128D \approx 286A$ for the new method. Thus, in this case, the new method yields a speed-up of 15% over the old method.

Implementation on ARM7TDMI platform
We implemented the fast verification procedure of ECDSA* signatures on an ARM7TDMI processor running at a 50MHz clock rate and compared this with the traditional ECDSA verification procedure. We assumed the same scenario as considered in the rough analysis above (in particular, we chose the same curve P-384), but *did* consider the cost of computing the half-size integers u and v required for fast verification. We restricted ourselves to Case 1. The following data was obtained:

signature generation time:	~100 ms;
traditional ECDSA verification time:	209 ms;
fast ECDSA* verification time:	154 ms.
speed-up:	36%.

The experimental data supports the rough analysis we did above for the signature verification step.

4.2 Fast Verification of ECDSA Signatures

In the previous section, we obtained a speed-up of ECDSA* signature verification. To make this efficiency improvement applicable to ECDSA as well, one needs to convert the ECDSA signature (r, s) over some message m to a corresponding ECDSA* signature (R, s) over the same message, i.e., one needs to reconstruct the ephemeral elliptic curve point R from the signature component r. Obviously, one could compute $R := s^{-1}(eG+rQ)$ directly, but this has the same computational cost as the traditional method for ECDSA signature verification and can, therefore, never yield an acceleration of signature verification. In this section, we consider alternative and more efficient mechanisms for reconstructing R from r. First, however, we provide a general framework.

By Theorem 2, one has $R \in \{(x, y) \in \mathbb{G} \mid f(\overline{x}) = r\}$ (the set of *candidate points*). Notice that for each $r \neq 0$, the set of candidate points has even cardinality, since elliptic curve points and their inverses have the same x-coordinate (and $\mathbb{G}$ has no points of order 2). Thus, one cannot uniquely extract the value of R from the set of candidate points alone, since candidate points come in pairs $(R, -R)$. It turns out, however, that in most cases there is only 1 candidate point (up to taking inverses in $\mathbb{G}$). In all cases, one can apply the fast ECDSA* verification procedure for each candidate point that is not discarded based on some side information and accept the ECDSA signature if and only if any verification using such an *admissible point* succeeds. By Theorem 2, this alternative procedure is equivalent to the original signature verification procedure for ECDSA.

The ECDSA verification cost via this procedure depends on the number of admissible points. In particular, if the set of admissible points is a singleton set, then ECDSA signature verification has the same cost as ECDSA* signature verification and the speed-up determined in §4.1 is attainable. We are now ready to discuss alternative mechanisms for reconstructing R from r.

Append explicit side information to ECDSA signatures
A simple method to make sure that the set of admissible points contains only one point is to supplement a traditional ECDSA signature (r, s) with some additional information, so as to ensure that one can uniquely (and efficiently) reconstruct R from r and this side information.

To illustrate this, consider an elliptic curve $E(\mathbb{Z}_q)$ of prime order n defined over the prime field $\mathbb{Z}_q$. We consider two cases. If $n > q$, the x-coordinate of R is equal to r. Thus, a single bit of side information is sufficient to efficiently determine the y-coordinate of R. If $n < q$, one can deduce from Hasse's theorem that $q < 2n$. Hence, the x-coordinate of R is equal to r or $r + n$, with the correct value being determined by a single extra bit of information. Thus, in this case, two bits of side information are sufficient to efficiently determine the point R.

More generally, if the elliptic curve $E(\mathbb{F}_q)$ defined over the finite field $\mathbb{F}_q$ has co-factor h, the x-coordinate of R can assume at most $h + 1$ values (h if $|E(\mathbb{F}_q)| > q$). Thus, $\lceil \log_2(h + 1) \rceil + 1$ bits of side information are sufficient to efficiently determine the point R.

Since most elliptic curves used in practice have a small co-factor (e.g., the NIST curves [3] have co-factor $h = 1$ (for prime curves) or $h = 2, 4$ (for binary curves)), adding a few bits of side information suffices to uniquely reconstruct R from r.

Observe that sending r with side information is similar to sending R in compressed form, but allows the use of ECDSA, rather than requiring the use of the modified scheme ECDSA* .

The general procedure for ECDSA signature is described below.

Accelerated ECDSA signature verification.
Input: Signature (r, s), message $m \in \{0, 1\}^*$, public key $Q \in \mathbb{G}$.
Output: Acceptance or rejection of signature relative to Q.
ACTIONS:

1. Verify that r and s are integers in the interval $[1, n-1]$. If any verification fails, return `'reject signature'`.
2. Compute the set of candidate points $\varphi(r) := \{(x, y) \in \mathbb{G} \mid f(\overline{x}) = r\}$.
3. Determine the set of admissible points $\mathcal{R} := \overline{\varphi}(r) \subseteq \varphi(r)$ by filtering out those candidate points that do not satisfy side constraints. If this set is empty, return `'reject signature'`.
4. Compute $e := \mathcal{H}(m)$.
5. Write $x := r/s$ as $x \equiv u/v \pmod{n}$, where u and v are integers that are significantly smaller than n.
6. Select an arbitrary point $R \in \mathcal{R}$. Compute $S := (v \cdot es^{-1})G + uQ - vR$. Set $\mathcal{R} := \mathcal{R} \setminus \{R\}$.
7. While ($S \neq \mathcal{O}$ and $\mathcal{R} \neq \emptyset$) do the following:
   (a) Select an arbitrary point $R' \in \mathcal{R}$.
   (b) Compute $S' := S + v(R - R')$.
   (c) Set $(R, S) := (R', S')$ and $R := \mathcal{R} \setminus \{R\}$.
8. If $S = \mathcal{O}$, return `'accept signature'`; otherwise, return `'reject signature'`.

Analysis of computational workload
The computational workload of the above algorithm is determined by the cost of computing admissible points and the cost of ECDSA* signature verifications. If an ECDSA signature gives rise to t admissible candidate points, then the expected workload of the above algorithm is $(t + 1)/2$ ECDSA* verifies. For example, if $h = 1$ and no side information is available, then $t = 2$ and the average workload is $1\frac{1}{2}$ ECDSA* verifies, which is still less than a traditional verification. If side information is available and $t = 1$ then only a single ECDSA* verification is required. In general, it can be shown that there are at most $2(h + 1)$ possible choices for the R-value. Clearly, the most favourable case is where $t = 1$. Note that the incremental cost of computing another ECDSA* verification is the cost of computing $v(R - R')$.

5 Conclusions

We have presented a method for accelerating ECDSA verification by roughly 40%. The only price one pays for the speedup is that a small number of bits needs to be appended to traditional ECDSA signatures. We emphasize that this change does not affect conformance to the existing ECDSA standards.

The speedups are also applicable to verification of DSA signatures $\sigma = (r, s)$. However, the side information one needs to efficiently recover R from r will be greater than the size of σ itself. Thus the advantage that DSA enjoys over RSA in terms of signature size is lost.

It is also evident that the speedups apply to the elliptic curve versions of many other variants of the ElGamal signature scheme.

References

1. ANSI X9.62-1998, *Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*, American National Standard for Financial Services, American Bankers Association, January 7, 1999.
2. G.L. Dirichlet, 'Verallgemeinerung eines Satzes aus der Lehrere von Kettenbrüchen nebst einigen Anwendungen auf die Theorie der Zahlen,' Bericht über die zur Bekanntmachung geeigneter Verhandlungen der Königlich Preussischen Akademie der Wissenschaften zu Berlin, pp. 93-95, 1842.
3. FIPS Pub 186-2, *Digital Signature Standard (DSS)*, Federal Information Processing Standards Publication 186-2, US Department of Commerce/National Institute of Standards and Technology, Gaithersburg, Maryland, USA, January 27, 2000. (Includes change notice, October 5, 2001.)
4. R. Gallant, R. Lambert, S.A. Vanstone, 'Fast Point Multiplication on Elliptic Curves with Efficient Endomorphisms,' in *Proceedings of Advances in Cryptology – CRYPTO 2001*, Lecture Notes in Computer Science, Vol. 2139, pp. 190-200, 2001.
5. D.R. Hankerson, A.J. Menezes, S.A. Vanstone, *Guide to Elliptic Curve Cryptography*, New York: Springer, 2003.
6. G.H. Hardy, E.M. Wright, *An Introduction to the Theory of Numbers*, Fifth Edition, Oxford: Oxford University Press, 2000.
7. D.J. Johnson, A.J. Menezes, S.A. Vanstone, 'The Elliptic Curve Digital Signature Algorithm (ECDSA),' *International Journal of Information Security*, Vol. 1, pp. 36-63, 2001.
8. L. Lovász, 'An Algorithmic Theory of Numbers, Graphs and Convexity,' CBMS-NSF Regional Conference Series in Applied Mathematics, Band 50, SIAM Publications, 1986.
9. P. Nguyen, D. Stehlé, 'Low-Dimensional Lattice-Basis Reduction Revisited,' in *Proceedings of Algorithmic Number Theory – ANTS VI*, Lecture Notes in Computer Science, Vol. 3076, pp. 338-357, 2004.
10. J. Solinas, 'Low-Weight Binary Representations for Pairs of Integers,' Centre for Applied Cryptographic Research, Corr 2001-41, University of Waterloo, Ontario, Canada, 2001.