# Using Randomizers for Batch Verification of ECDSA Signatures

Sabyasachi Karati
Abhijit Das
Dipanwita Roychowdhury

Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur, India
skarati,abhij,drc@cse.iitkgp.ernet.in

October 13, 2012

**Abstract.** Randomizers are popularly used to prevent various types of attacks on batch-verification schemes. Recently, several algorithms based upon symbolic computation are proposed for the batch verification of ECDSA signatures. In this article, we demonstrate that the concept of randomizers can be easily embedded in these symbolic-computation algorithms. The performance degradation caused by randomizers is comparable with that associated with ECDSA*.

**Keywords:** Digital Signature, Elliptic Curve, ECDSA, ECDSA*, Batch Verification, Symbolic Computation, Randomizer, Montgomery Ladder.

## 1 Introduction

In many applications, digital signatures are verified in batches in order to reduce the total verification time. For many digital-signature algorithms, the batch-verification criterion is reasonably obvious. However, the popular ECDSA signature scheme presents some resistance to batch-verification ideas. During ECDSA signature generation, an elliptic-curve point $R$ is computed, and only the $x$-coordinate $r$ of $R$ is included in the signature. Although this does not impair the verification primitive or the security of the scheme, absence of the knowledge of the $y$-coordinate of $R$ leads to special considerations for batch verification of ECDSA signatures. In [2], several symbolic-computation schemes are proposed to address this issue. These schemes are shown, in terms of security, to be equivalent to ECDSA* (the variant of ECDSA, where the entire point $R$ replaces $r$ in the signature).

Batch verification schemes suffer from special attacks because, in some sense, these schemes verify the aggregate of a collection of signatures. Two such attacks on the schemes of [2] are proposed by Bernstein et al. [1].

In the notation of [2], the verification equation for the $i$-th signature is:

$$R_i = u_i P + v_i Q_i.$$

A collection of $t$ signatures is verified as:

$$\sum_{i=1}^{t} R_i = \left(\sum_{i=1}^{t} u_i\right) P + \left(\sum_{i=1}^{t} v_i Q_i\right). \tag{1}$$

Naccache et al. [4] propose the use of randomizers during the aggregation of the verification equations. For randomly chosen non-zero multipliers $\xi_1, \xi_2, \ldots, \xi_t$, we now verify whether the following equality holds:

$$\sum_{i=1}^{t} \xi_i R_i = \left(\sum_{i=1}^{t} \xi_i u_i\right) P + \left(\sum_{i=1}^{t} \xi_i v_i Q_i\right). \tag{2}$$

(The original batch-verification criterion corresponds to $\xi_1 = \xi_2 = \cdots = \xi_t = 1$.) Many attacks on batch-verification schemes can be eliminated (or made infeasible) by using these randomizers. However, the computation of $t$ scalar multiples $\xi_i R_i$ significantly brings down the performance gains achieved by batching.

If only the $x$-coordinates $r_i$ of $R_i$ are available, neither Eqn (1) nor Eqn (2) is directly applicable. In [2], the point $\sum_{i=1}^{t} R_i$ in Eqn (1) is computed symbolically. It is conjectured in [1] that these symbolic-computation algorithms are hard to adapt to the weighted case of Eqn (2).

In this article, we demonstrate that the use of randomizers can be seamlessly embedded in the symbolic-computation algorithms of [2]. We present two techniques (essentially equivalent to one another) highlighting that the introduction of randomizers is no hindrance to the working of the symbolic-computation algorithms on the weighted sum $\sum_{i=1}^{t} \xi_i R_i$. Moreover, the performance degradation caused by randomizers is practically the same in these algorithms as it is in ECDSA*.

## 2 Numeric Computation of $\xi_i R_i$

Suppose that only the $x$-coordinate of $R = (r, y)$ is provided. We treat the $y$-coordinate of $R$ as a symbol satisfying the elliptic-curve equation:

$$y^2 = r^3 + ar + b.$$

Any non-zero multiple $uR$ of $R$ is of the form $(h, ky)$, where $h$ and $k$ are field elements fully determined by the known $x$-coordinate $r$ (and $u$). For proving this, we let $P_1 = (h_1, k_1 y)$ and $P_2 = (h_2, k_2 y)$ be two non-zero multiples of $R$. The following easily verifiable formulas show that the sum $P_3 = P_1 + P_2$ and the double $P_4 = 2P_1$ (if non-zero) can again be expressed in the form $P_3 = (h_3, k_3 y)$ and $P_4 = (h_4, k_4 y)$.

$$\begin{aligned}
\lambda &= \frac{k_2 - k_1}{h_2 - h_1} & \lambda' &= \frac{3h_1^2 + a}{2k_1} \\
h_3 &= (r^3 + ar + b)\lambda^2 - h_1 - h_2 & h_4 &= (r^3 + ar + b)^{-1}\lambda'^2 - 2h_1 \\
k_3 &= \lambda(h_1 - h_3) - k_1 & k_4 &= (r^3 + ar + b)^{-1}\lambda'(h_1 - h_4) - k_1
\end{aligned}$$

Upon $(r, 1)$ as input, we precompute the quantity $r^3 + ar + b$ and its inverse, and run the standard repeated double-and-add loop with these revised addition and doubling formulas. At the end of the loop, the two computed field elements $h, k$ yield the desired multiple $\xi R = (h, ky)$. In short, we do not need to carry out any symbolic computation at all for obtaining $\xi R$. Moreover, these formulas readily adapt to faster variants of the standard double-and-add point-multiplication algorithm (like the windowed variant or, for that matter, any addition-chain-based variant).

The modified addition formula involves only one extra field multiplication (by the precomputed quantity $r^3 + ar + b$) than the standard elliptic-curve addition formula. Point doubling requires two extra field multiplications (each by the precomputed inverse $(r^3 + ar + b)^{-1}$). This implies that the modified double-and-add point-multiplication algorithms are slightly slower than their standard counterparts.

After $\xi_i R_i = (h_i, k_i y_i)$ are computed for all $i = 1, 2, \ldots, t$, we supply these points as input to the symbolic-computation algorithms for batch verification. The $y_i$ values continue to satisfy the equations $y_i^2 = r_i^3 + ar_i + b$ to be used in the symbolic simplification process, where $r_i$ are available from the ECDSA signatures.

The knowledge of the entire points $R_1, R_2$ allows us to compute $\xi_1 R_1 + \xi_2 R_2$ using a single double-and-add loop, yielding noticeable speedup over two point multiplications. If the $y$-coordinates of $R_1$ and $R_2$ are treated as symbols $y_1, y_2$, then too $\xi_1 R_1 + \xi_2 R_2$ can be computed seminumerically. Any non-zero point of the form $uR_1 + vR_2$ can be expressed as $(h + jy_1 y_2, ky_1 + ly_2)$ for field elements $h, j, k, l$ uniquely determined by the $x$-coordinates $r_1, r_2$ (and $u, v$) alone. Addition and doubling of such points can be rephrased numerically in terms of these field elements. However, the resulting formulas are somewhat clumsy, and are not expected to benefit the computation of $\xi_1 R_1 + \xi_2 R_2$ in a single double-and-add loop. For the weighted sum of three or more points, this idea of seminumeric computation can be theoretically extended, but chances of getting practical benefits are rather slim.

## 3 Using Montgomery Ladders

It is easy to see that if only the $x$-coordinate $r$ of a point $R$ is provided, the $x$-coordinate of any multiple $\xi R$ can be uniquely determined. The Montgomery ladder [3] is a concrete algorithmic realization of this idea. The ladder never uses nor computes the $y$-coordinate of any point in its repeated double-and-add point-multiplication loop.

We can use Montgomery ladders in our context as follows. Let the $x$-coordinates $r_1, r_2, \ldots, r_t$ of $R_1, R_2, \ldots, R_t$ be available from ECDSA signatures. We choose random multipliers $\xi_1, \xi_2, \ldots, \xi_t$, and compute the $x$-coordinates $\bar{r}_i$ of $\xi_i R_i$ using Montgomery ladders for $i = 1, 2, \ldots, t$. We supply the $x$-coordinates $\bar{r}_1, \bar{r}_2, \ldots, \bar{r}_t$ as input to the symbolic-computation algorithms for batch verification. The corresponding $y$-coordinates $\bar{y}_i$ now satisfy

$$\bar{y}_i^2 = \bar{r}_i^3 + a\bar{r}_i + b$$

for all $i$. These relations are to be used now in the symbolic simplification process.

The double-and-add loop based on Montgomery ladders is comparable in performance with the double-and-add loop of the standard point-multiplication algorithm. Although Montgomery ladders never compute any $y$-coordinate, their formulas for $x$-coordinates involve more field operations than standard point multiplication. Moreover, irrespective of the bits of the multiplier, each iteration in the Montgomery-ladder loop performs one doubling and one addition. Finally, it is not always beneficial to extend Montgomery ladders to windowed or addition-chain-based variants.

Our preliminary experiments tend to suggest that the numeric computation of the $x$-coordinate of $\xi R$ (as discussed in Section 2) is slightly faster than the computation based on the Montgomery ladder. Moreover, the numeric computation also supplies the other coordinate of $\xi R$ as a multiple of $y$. Indeed, these numeric formulas provide an alternative to the Montgomery ladder not only in our current context but in other contexts too, including the context of [3] where the Montgomery ladder is introduced. However, point multiplication using Montgomery ladders is more resistant to side channel attacks than the numeric algorithm.

## 4 Effect on Performance

The extra security offered by the randomizers $\xi_1, \xi_2, \ldots, \xi_t$ comes at a cost. We now need to compute $t$ additional point multiplications (or something equivalent to that). This performance degradation affects all variants of the DSA family of signatures, including ECDSA and ECDSA*. The previous two sections establish that the symbolic-manipulation algorithms for ECDSA batch verification are affected too, but not more severely than those in which the entire points $R$ are available as input.

Fortunately, however, we can make a compromise. The randomizers need not be of full lengths (that is, of lengths close to that of the prime order $q$ of the relevant elliptic-curve group). Much smaller randomizers typically suffice to frustrate most attacks on batch-verification schemes. For example, if we take $q$ to be a $512$-bit prime, and if we choose $64$-bit randomizers, then for a batch of size eight, the total overhead of using randomizers is about the same as that of only one point multiplication.

## 5 Conclusion

This article illustrates that the ECDSA batch-verification algorithms based on symbolic manipulation are fully compatible with the use of randomizers, and are equivalent to ECDSA* batch verification in this regard, in terms of both performance and security. An alternative to the Montgomery ladder is also proposed.

## References

1. D. J. Bernstein, J. Doumen, T. Lange and J.-J. Oosterwijk, *Faster batch forgery identification*, to appear in IndoCrypt 2012.
2. S. Karati, A. Das, D. Roychowdhury, B. Bellur, D. Bhattacharya and A. Iyer, *Batch verification of ECDSA signatures*, AfricaCrypt 2012, Lecture Notes in Computer Science 7374, 1–18, 2012.
3. P. L. Montgomery, *Speeding up Pollard and elliptic curve methods of factorization*, Mathematics of Computation 48(177), 243–264, 1987.
4. D. Naccache, D. M'Raïhi, D. Rapheali and S. Vaudenay, *Can D.S.A. be improved: Complexity trade-offs with the digital signature standard*, EuroCrypt 1994, Lecture Notes in Computer Science 950, 77–85, 1994.