Digital Signature Schemes with Domain Parameters

Serge Vaudenay

EPFL
http://lasecwww.epfl.ch/

Abstract. Digital signature schemes often use domain parameters such as prime numbers or elliptic curves. They can be subject to security threats when they are not treated like public keys. In this paper we formalize the notion of "signature scheme with domain parameter" together with a new adversarial model: the "domain parameter shifting attack". We take ECDSA as a case study. We make a domain parameter shifting attack against ECDSA: an attacker can impersonate a honest signer either by trying to modify the subgroup generator G or, when using point compression representation, by trying to modify the elliptic curve a and b domain parameters. We further propose to fix this ECDSA issue.

1 Introduction

Following pioneer work by Merkle [17], Diffie-Hellman [8], and Rivest-Shamir-Adleman [19], a formal framework for public-key digital signature schemes was proposed. These schemes are used in order to transform an insecure communication channel into a channel which guarantees authentication, provided that an extra authenticating channel can be used for setting up the system. Typically, one uses an authenticating channel in order to transmit a public key. The public key is associated to a secret one. Then, the digital signature algorithm can provide authentication of a document by typically proving that the document has been signed by a process which possesses the secret key. (See Fig. 1.)

Authenticating public keys for a signature scheme is an odd problem since it already needs signatures from a Certification Authority (CA). This requires that the CA public key is initially authenticated by an alternate mean. This "root authentication" is more expensive and critical.

Digital signature algorithms e.g. DSA often rely on domain parameters. DSA [2,4,6] follows a long dynasty of ElGamal schemes [9,10,11,20,21]. It relies on some primes p, q and a q-ordered subgroup of $\mathbf{Z}_p^*$ generated by some g residue. An extra seed is used in order to convince that p and q were randomly generated, thus to make users trust their safety. Domain parameters consist of a (p, q, g, seed) quadruplet. ECDSA [3,6], a variant of DSA, relies on some finite field $\mathbf{F}_q$, some elliptic curve C over $\mathbf{F}_q$ defined by some $a, b \in \mathbf{F}_q$, some prime number n and some n-ordered subgroup of C generated by a point G. Domain parameters consist of a $(q, \text{representation}, a, b, n, G, \text{seed})$ tuple.

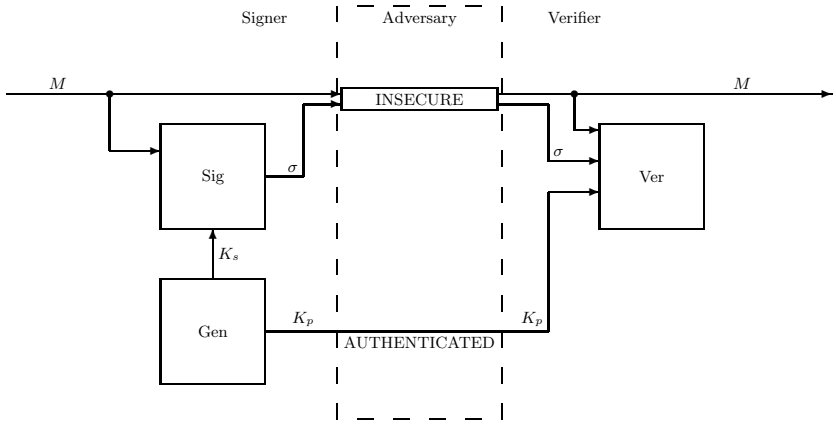

Fig. 1. Digital Signature Scheme.

Typically, making public-private key pairs in RSA [19] is much more expensive than signing or verifying. In DSA [6], the expensive computation for key generation factorizes in making common domain parameters. Then, making key pairs is inexpensive. Indeed, the *raison d'être* of domain parameters is essentially in being set up and validated once for all, e.g. in a ad-hoc network.

Domain parameters face to several security problems.

- They should be authenticated. This means that alternate authentication means have to be used.
- They should be trusted. End users should be protected against malicious choice of domain parameters. This point is particularly relevant in the case of ECDSA since seldom end users master elliptic curve techniques. (Fortunately, ECDSA can rely on some well established standardization bodies like SECG[1] which provide lists of standard elliptic curves.)

In this paper we propose a definition of "digital signature scheme with domain parameters" (DSSDP) in Sect. 2 and security requirements in Sect. 2.2. For our case study, we rephrase the ECDSA specifications in these settings in Sect. 3.1. We first review in Sect. 3.2 a known domain parameter issue from [24] and the proposed fix. We show that the fix is not enough by describing some new attacks in Sect. 3.3 and Sect. 3.4. We finally propose to update ECDSA.

2 Digital Signature Schemes with Domain Parameters

2.1 Definition

Digital signature schemes are traditionally defined as a set of three algorithms: key generation, signature, and verification algorithms. This definition does not

[1] `http://www.secg.org`

capture the notion of domain parameters. As a matter of fact, many current digital schemes use domain parameters as an intuitive notion which should be quite clear from context even though it was not properly formalized. We address this issue by proposing the following definition. We believe that it reflects the common intuition.

Definition 1. *A digital signature scheme with domain parameter (DSSDP) is a set of five algorithms.*

SetUp(t)**:** *an algorithm which generates a domain parameter P using a security parameter t.*
Val(t, P)**:** *an algorithm which verifies the validity of P.*
Gen(t, P)**:** *an algorithm which generates public-secret key pairs (K_p, K_s).*
Sig(t, P, K_s, M)**:** *an algorithm which produces a signature σ on a digital document M.*
Ver(t, P, K_p, M, σ)**:** *an algorithm which verifies signed-documents.*[2]

Val and Ver are deterministic predicates. SetUp, Gen, and Sig are probabilistic algorithms, hence they output random values. **Completeness** *is achieved by the following properties.*

1. *For any t, if we let $P \leftarrow \mathrm{SetUp}(t)$, then the $\mathrm{Val}(t, P)$ predicate holds with probability 1.*
2. *For any t and M, if we let $P \leftarrow \mathrm{SetUp}(t)$, $(K_p, K_s) \leftarrow \mathrm{Gen}(t, P)$ and $\sigma \leftarrow \mathrm{Sig}(t, P, K_s, M)$, then the $\mathrm{Ver}(t, P, K_p, M, \sigma)$ predicate holds with probability 1.*

Here, the left arrow notation, e.g. in $P \leftarrow \mathrm{SetUp}(t)$ means that P is random following the distribution generated by the SetUp probabilistic algorithm.

For simplicity reasons we assume that there is a canonical way to extract t from P so there is no need for having t as a parameter for Gen, Sig and Ver. We still need to verify that P is consistent with a chosen t in Val though.

The usage of this scheme is as depicted on Fig. 2.

Set up. Everyone agrees on a security parameter t. Some central service first generates $P \leftarrow \mathrm{SetUp}(t)$ and broadcasts it.
Public key authentication. The signer checks that $\mathrm{Val}(t, P)$ holds and generates $(K_p, K_s) \leftarrow \mathrm{Gen}(P)$. She then sends K_p to the verifiers in an authenticated way (e.g. using a certificate obtained from a Certificate Authority).
Signature. To sign M, the signer computes $\sigma \leftarrow \mathrm{Sig}(P, K_s, M)$ and sends M, σ to the verifier. The verifier checks that $\mathrm{Val}(t, P)$ and $\mathrm{Ver}(P, K_p, M, \sigma)$ hold.

We emphasize that we do not assume that broadcasting P is secure nor that the P issuer is trusted. In particular the signer and the verifier may receive different domain parameters due to malicious broadcast.

Other studies like Menezes-Smart [16] define a signature scheme with an additional algorithm: a public key validation scheme which is important when considering multi-user settings. We omit it in this paper.

[2] Signature schemes with message recovery do not take M as an input but rather produce it as an output. We do not consider it in this paper for simplicity reasons.

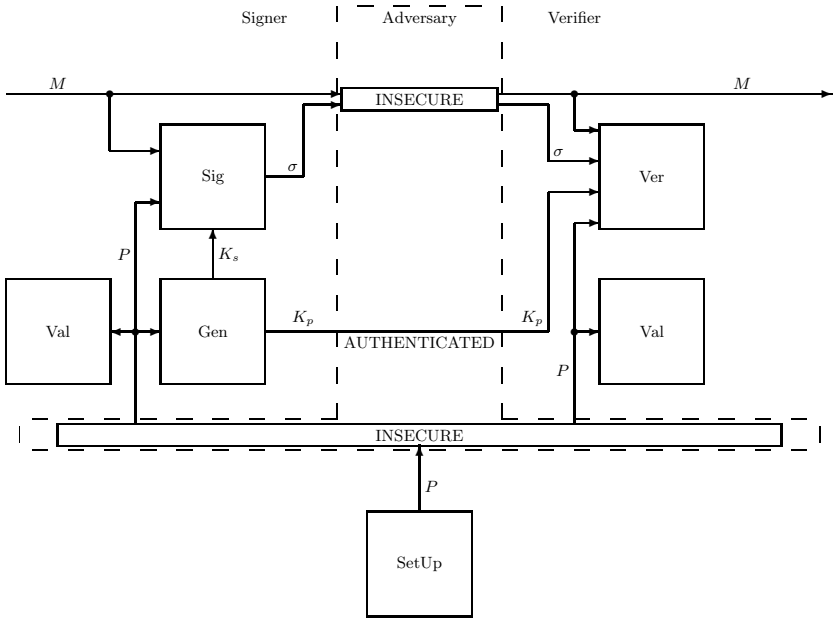

Fig. 2. Digital Signature Scheme with Domain Parameters.

2.2 Security Requirements

Here we formulate the security requirements on DSSDP.

Resistance to existential forgery. An adversary is given P, K_p, access to the $\mathrm{Sig}(P, K_s, .)$ oracle, and aims at forging a (M, σ) pair such that $\mathrm{Ver}(P, K_p, M, \sigma)$ holds without querying the oracle with M. We say that the signature scheme is (B_t, ε_t)-secure if for any t and any adversary of complexity bounded by B_t, we have $\Pr[\text{success}; P \leftarrow \mathrm{SetUp}(t), (K_p, K_s) \leftarrow \mathrm{Gen}(P)] \leq \varepsilon_t$.

This was proposed by Goldwasser, Micali and Rivest [12,13] as the strongest security requirement. Menezes and Smart proposed to extend it in a multi-user setting [16]. Here, an alternate goal for the adversary is to output (M, σ) and (K'_p, K'_s) such that both $\mathrm{Ver}(P, K_p, M, \sigma)$ and $\mathrm{Ver}(P, K'_p, M, \sigma)$ hold and that (K'_p, K'_s) is a valid key pair. In this case, the adversary is allowed to query M to the $\mathrm{Sig}(P, K_s, .)$ oracle. This is called a "key substitution attack".

Domain parameter integrity. The definition is the same, but for the goal of the adversary which is now to forge a (P', M, σ) triplet such that $P \neq P'$, $\mathrm{Val}(t, P')$ and $\mathrm{Ver}(P', K_p, M, \sigma)$ hold without querying the oracle with M.

We call this condition "domain parameter integrity" since it makes clear that when the verifier checks that $\mathrm{Val}(t, P)$ and $\mathrm{Ver}(P, K_p, M, \sigma)$ hold then his P is the same domain parameter than the one of the signer. Hence K_p is bound to P.

This condition may look strange. Actually, it becomes mandatory when we do not want to rely on trusting the P issuer or when P is not strongly authenticated.

Note that if the domain parameter P is considered as a part of the public key K_p, then this security requirement is already implicitly included in the Menezes-Smart one. [16] mentions this fact (Remark 5(ii)) but completely omits it in the analysis. (Otherwise our results from Sect. 3 would contradict their Theorem 9.)

Domain parameter consistency. For any t, the set of all P such that $\mathrm{Val}(t, P)$ holds is exactly the set of all possible outputs for $\mathrm{SetUp}(t)$.

This means that Val accepts no more domain parameters than those generated by SetUp. This prevents e.g. from having n domain parameters which are not prime numbers in ECDSA.

Trust in domain parameters. When the above security requirements hold for a fixed domain parameter P, we say that P is trusted. We say that the DSSDP provides trust in domain parameters if any P for which $\mathrm{Val}(t, P)$ holds is trusted.

This means that the above properties do not only hold *on average* for any random domain parameter which was honestly generated, but for *any* acceptable domain parameter. Its purpose is to prevent from unknown attacks by malicious selection of domain parameters (like e.g. hiding trapdoors in an elliptic curve).

Abuse of trust in domain parameter may happen. As a famous example we mention the Bleichenbacher attack [7] against the ElGamal signature in which domain parameters p and g are chosen such that signatures are easy to forge.

Finally we formulate the following security definition. The adversary model is depicted on Fig. 3.

Definition 2. *Given a DSSDP as in Def. 1, we consider adversaries which are given P, K_p, access to the $\mathrm{Sig}(P, K_s, .)$ oracle, and which aim at forging a (P', M, σ) triplet such that $\mathrm{Val}(t, P')$ and $\mathrm{Ver}(P', K_p, M, \sigma)$ hold without querying the oracle with M. When $P' \neq P$ we call it a "domain parameter shifting attack". We say that the signature scheme is (B_t, ε_t)-secure if for any t, any adversary of complexity bounded by B_t, and any P such that $\mathrm{Val}(t, P)$, we have $\Pr[\mathrm{success}; (K_p, K_s) \leftarrow \mathrm{Gen}(P)] \leq \varepsilon_t$.*

2.3 Reduction to Traditional Signature Schemes

Obviously, traditional digital signature schemes are also DSSDP in which the domain parameter is void. Conversely, we can easily transform the DSSDP into a classical digital signature scheme as follows.

$\mathrm{Gen}'(t)$: run $P \leftarrow \mathrm{SetUp}(t)$ and $(K_p, K_s) \leftarrow \mathrm{Gen}(P)$, set $K'_p = (P, K_p)$, $K'_s = (P, K_s)$ and output (K'_p, K'_s).
$\mathrm{Sig}'(K'_s, M)$: same as $\mathrm{Sig}(P, K_s, M)$.
$\mathrm{Ver}'(K'_p, M, \sigma)$: check that $\mathrm{Val}(t, P)$ and $\mathrm{Ver}(P, K_p, M, \sigma)$ hold.

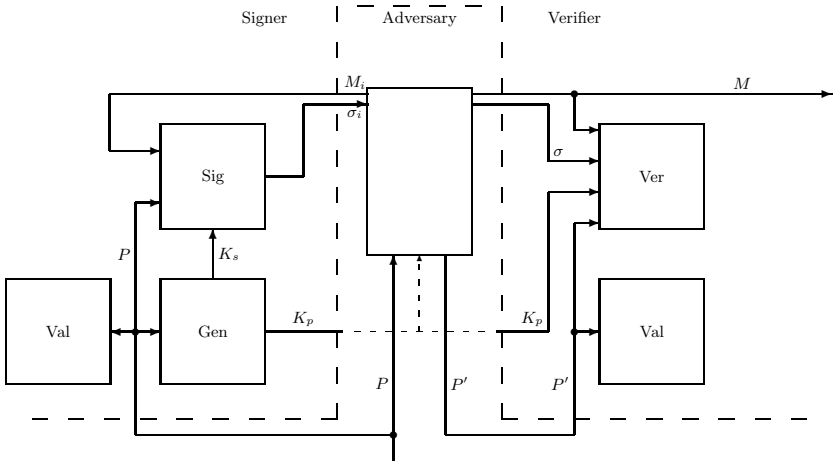

Fig. 3. Adversary for a Digital Signature Scheme with Domain Parameters.

Here P becomes a part of the public key. This is a typical situation in client-server communications: servers send their (P, K_p) pair in a X.509 [14] certificate. This works provided that inclusion of domain parameters in the certificate is mandatory. This is not the case in [14] which suggests that domain parameters can be transmitted "by other means". In this paper we concentrate on using DSSDP with common domain parameters e.g. in ad-hoc networks, which excludes treating domain parameters as part of public keys.

3 ECDSA

3.1 Definition

We summarize the ECDSA with notations from ANSI X9.62 [3][3] in our format. We use the hash function SHA-1 [5] which we denote H.

SetUp(t)**:** do as follows. The security parameter t specifies the field size and type (either a field of characteristic two, or a large prime field).

1. Choose the finite field $\mathbf{F}_q$ according to t.
2. Either select a standard elliptic curve or use the following scheme.
   - Prime field case: pseudo-randomly generate a string c from a seed and translate it into a field element. Pick field elements a and b such that $a^3/b^2 \equiv c \pmod{q}$ and consider the curve $y^2 = x^3 + ax + b$ over $\mathbf{F}_q$. Note that the j-invariant of the curve is $j = 6912\frac{c}{4c+27}$.

[3] The only difference with [3] is that $\mathbf{F}_q$ denotes the finite field even when q is prime, instead of $\mathbf{F}_p$.

– Characteristic two case: pseudo-randomly generate a string c from a seed, translate it into a field element and call it b. Pick a field element a and consider the curve $y^2 + xy = x^3 + ax^2 + b$ over $\mathbf{F}_q$. Note that the j-invariant of the curve is $j = \frac{1}{b}$.

3. For q prime, check that $4a^3 + 27b^2 \bmod q \neq 0$. For q a power of two, check that $b \neq 0$. If this is not the case, go back to Step 2.
4. Count the number of points in the elliptic curve and isolate a prime factor n greater than 2^{160}. If this does not work, go back to Step 2.
5. Check the MOV and anomalous condition for C. If this does not hold, go back to Step 2.
6. Pick a random point on the elliptic curve and raise it to the cofactor of n power in order to get G. If G is the point at infinity, try again.

Finally set $P = (q, \text{representation}, a, b, n, G, \text{seed})$.

Val$(t, q, \text{representation}, a, b, n, G, \text{seed})$**:**

1. Check that q is an odd prime or a power of 2 of appropriate size. In the latter case, check that the field representation choice is valid.
2. Check that a, b, x_G, y_G (where $G = (x_G, y_G)$) lie in $\mathbf{F}_q$.
3. Check that seed certifies a and b by generating c again and checking that $\frac{a^3}{b^2} = c$ or $b = c$ depending on the field type.
4. For q prime, check that $4a^3 + 27b^2 \bmod q \neq 0$. For q a power of two, check that $b \neq 0$. Check that G lies in the elliptic curve. Check that n is a prime greater than both 2^{160} and $4\sqrt{q}$. Check that $nG = \mathcal{O}$, the neutral element. Check the MOV and anomalous condition.

Gen$(q, \text{representation}, a, b, n, G)$**:** pick an integer d in $[1, n-1]$, compute $Q = dG$. Output $(K_p, K_s) = (Q, d)$.

Sig$(q, \text{representation}, a, b, n, G, d, M)$**:** pick k in $[1, n-1]$ at random and compute $(x_1, y_1) = kG$, $r = \overline{x_1} \bmod n$, and $s = \frac{H(M)+dr}{k} \bmod n$. (Here $\overline{x_1}$ is simply a standard way to convert a field element x_1 into an integer.) If $r = 0$ or $s = 0$, try again. Output the signature $\sigma = (r, s)$

Ver$(q, \text{representation}, a, b, n, G, Q, M, r, s)$**:** check that $Q \neq \mathcal{O}$, $Q \in C$, and $nQ = \mathcal{O}$. Check that r and s are in $[1, n-1]$ and that $r = \overline{x_1} \bmod n$ for $(x_1, y_1) = u_1 G + u_2 Q$, $u_1 = \frac{H(M)}{s} \bmod n$, and $u_2 = \frac{r}{s} \bmod n$.

The signature is a pair of integers. The public key is a point on a curve. So the standard should define a standard way for representing an integer, a point, and therefore a field element. In addition we need a standard way to represent the domain parameters: the field representation, the curve definition, etc. ANSI X9.62 [3] extensively defines all this.

To illustrate, we take the example with a 192-bit prime q from [3, Sect. J.3.1, p. 152]. This example is also called "Curve P-192" in FIPS 186 [6] and `secp192r1` in SEC2 [1]. We consider a domain parameter P defined by

$$q = 6277101735386680763835789423207666416083908700390324961279$$

$$a = \texttt{ffffffff ffffffff ffffffff fffffffe ffffffff fffffffc}$$

$$b = \texttt{64210519 e59c80e7 0fa7e9ab 72243049 feb8deec c146b9b1}$$

$$n = 6277101735386680763835789423176059013767194773182842284081$$
$$G = \texttt{03 188da80e b03090f6 7cbf20eb 43a18800 f4ff0afd 82ff1012}$$
$$\text{seed} = \texttt{3045ae6f c8422f64 ed579528 d38120ea e12196d5}$$

(note that the leading "03" of G means that y_G is odd and that the remaining represents x_G, and that a, b, and G are in hexadecimal notations).[4] We can easily decompress G and compute the y coordinate. We obtain

$$y_G = 1740503322936220314048575522802194103640234889273866506 41.$$

Note that the validation algorithm checks that the j-invariant comes from the seed. We consider a secret and public key defined by

$$d = 651056770906015076056810763456358567190100156695615665659$$
$$Q = \texttt{02 62b12d60 690cdcf3 30babab6 e69763b4 71f994dd}$$

(note that the leading "02" of Q means that y_Q is even).

3.2 Trusting Elliptic Curves

As shown in Koblitz [15], trust in elliptic curves is subject to many property checks. They are necessary due to the existence of weak elliptic curves, e.g. curves of trace one [18,22].

The use of the seed in the domain parameters should ensure the confidence that the elliptic curve was randomly — and therefore not maliciously — selected. It was shown to be necessary in the case of DSA due to possible malicious choice of p and q prime domain parameters (see [23]).

As mentioned in [24], the seed in ECDSA does not perform a good job in the characteristic two case since we can first select an elliptic curve then choose the finite field representation which validates it. It was proposed to tweak SetUp (and Val accordingly) as follows in the same spirit (i.e. the elliptic curve up to an isomorphism only is generated from the seed and not the a and b domain parameters).

Curves over $\mathbf{F}_q$ with $q > 3$ prime:

1. Choose a prime $q > 3$ and consider $\mathbf{F}_q$.
2. Generate a random bit string c and bit from seed and q.
3. Translate c into a field element.
4. Select arbitrarily a and b such that $a^3/b^2 \equiv c \pmod q$ and $\left(\frac{b}{q}\right) = (-1)^{\text{bit}}$, and take the elliptic curve defined by a, b, q.

Curves over $\mathbf{F}_q$ of characteristic 2:

1. Choose q a power of 2 and consider $\mathbf{F}_q$. We choose a representation of $\mathbf{F}_q$ (i.e. an irreducible polynomial).
2. Generate a random bit string c and bit from seed, q, and the field representation choice.

[4] Note that $q = 2^{192} - 2^{64} - 1$.

3. Translate c into a field element and call it b.
4. Select arbitrarily a such that $\mathrm{Tr}(a) = \text{bit}$ and take the elliptic curve defined over $\mathbf{F}_q$ by a, b, q.[5]

3.3 Subgroup Integrity

There is actually no integrity protection for G. The attacker can actually replace G by a random power of Q. In this case she knows the discrete logarithm of Q in this basis and can forge signatures which will pass the Ver test for any message.

More precisely, following the model which is depicted on Fig. 3, the adversary performs a domain parameter shifting attack as follows.

1. Intercept the domain parameters P and extract G, n, and necessary materials in order to perform computations in the elliptic curve.
2. Get K_p and extract Q.
3. Pick an arbitrary $d' \in \mathbf{Z}_n^*$ and construct P' from P in which G is replaced by $G' = ({d'}^{-1} \bmod n).Q$.
4. Send P' to the verifier.
5. Forge signatures by using the Sig algorithm with d replaced by d'.

Note that no oracle call is made to the signer.

As a trivial example an attacker who intercepts P and Q can forge a domain parameter P' by replacing G by Q. The attacker can thus forge signatures for the public key Q with domain parameter P' by taking a secret key set to $d' = 1$. Making other examples with less trivial d' is quite straightforward.

This attack tells us that G must be protected by specific means. For instance random selection can be demonstrated by generating it from the seed.

3.4 Elliptic Curve Integrity

The integrity of the elliptic curve up to an isomorphism (more precisely with a fixed j-invariant) may also be problematic. Indeed, in the prime case, if an attacker is given $P = (q, a, b, n, G, \text{seed})$ and Q from a signer, she can try to forge a consistent $P' = (q, a', b', n, G, \text{seed})$ with $a' = au^4 \bmod q$ and $b' = bu^6 \bmod q$ and such that G and Q still lie in the elliptic curve defined by P' and that she knows the discrete logarithm in basis G. When G and Q are fully specified, this is not possible since (a', b') must be the unique solution of

$$y_G^2 - x_G^3 = a'x_G + b'$$
$$y_Q^2 - x_Q^3 = a'x_Q + b'$$

which is (a, b). When G and Q are specified with point compression, only their x coordinates are given together with one bit of the y coordinate. The attacker can thus for instance try to find u such that Q suddenly becomes equal to $2.G$

[5] Here $\mathrm{Tr}(a)$ denotes $a + a^2 + a^4 + a^8 + \cdots + a^{\frac{q}{2}}$.

in the new curve. By using the expression of the x coordinate of $2.G$ we obtain the equation

$$x_Q = \frac{(3x_G^2 + au^4)^2}{4(x_G^3 + au^4 x_G + bu^6)} - 2x_G \tag{1}$$

which is a polynomial equation in u^2 of degree 3. It is thus likely to lead to at least one solution! More generally, one can try to solve $(d'.G)_1 = x_Q$ for a small integer d' where $d'.G$ stands for the computation in the new curve defined by the unknown u and $(d'.G)_1$ is its x coordinate.

More precisely, following the model which is depicted on Fig. 3, in the prime field case with point compression, the adversary performs a domain parameter shifting attack as follows.

1. Intercept the domain parameters P and extract q, a, b, G.
2. Get K_p and extract Q.
3. Solve Eq. (1) in u and set $d' = 2$. If there is no solution, increment d' and solve $(d'.G)_1 = x_Q$ until it has solutions.
4. Compute $a' = au^4 \bmod q$ and $b' = bu^6 \bmod q$.
5. Construct P' from P in which a and b are replaced by a' and b' respectively.
6. Send P' to the verifier.
7. Forge signatures by using the Sig algorithm with d replaced by d'.

Note that no oracle call is made to the signer.

Similar observations hold in the characteristic two case.

We illustrate the attack with our domain parameters example.

An attacker who intercepts P and the public key Q needs to solve Eq. (1) modulo q. This leads to two solutions

$$u_1 = 231134335139140593734522468707266159406956283909983063190 2$$

$$u_2 = 396575838399527482649056473613500482201434586129049432937 7.$$

The first solution $u = u_1$ leads to

$$a' = \texttt{24f70108 7a05f49c 67119ba6 bba22c93 697a5cc8 5f936eb5}$$

$$b' = \texttt{31a2b88a dd0b97c0 fdf876b3 e505cc3b 22378efb 5a6d4eb7}.$$

So the attacker can forge a domain parameter P' from P where a and b are replaced by a' and b' respectively. Since both elliptic curves have the same j-invariant the new one will pass the validation algorithm with the same seed.

On this new curve the y coordinate of G decompresses as

$$y'_G = 111631535859431414668945647216563456779285726721083842217$$

and we can check that the x coordinate of $2.G$ is equal to x_Q and that its y coordinate is even on this new elliptic curve. So $2.G = Q$ on this elliptic curve. The attacker can thus forge signatures for the public key Q with domain parameter P' by taking a secret key set to $d' = 2$.

This attack tells us that we must have a stronger link than what [24] suggested between the domain parameters and the public key, e.g. generate a and b from the seed.

4 Conclusion

We have formalized the notion of digital signature schemes with domain parameters. We formulated security requirements. We demonstrated that ECDSA does not satisfy the domain parameter integrity condition due to the lack of validation procedure for the subgroup generator G domain parameter and elliptic curve a and b domain parameters and that the fix which was proposed in [24] is not sufficient. Finally, we propose an appropriate way to update the scheme in one of the two following ways.

1. Really generate a, b, G from the seed at random.
2. Forget about DSSDP and make sure that P is authenticated together with K_p by the certificate authority.

5 Acknowledgments

The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

References

1. SEC 2: Recommended Elliptic Curve Cryptography Domain Parameters. v1.0, Certicom Research, 2000.
2. ANSI X9.30. Public Key Cryptography for the Financial Services Industry: Part 1: The Digital Signature Algorithm (DSA). American National Standard Institute. American Bankers Association. 1997.
3. ANSI X9.62. Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). American National Standard Institute. American Bankers Association. 1998.
4. ISO/IEC 14888. Information Technology — Security Techniques — Digital Signatures with Appendix. ISO/IEC, Geneva, Switzerland, 1998.
5. Secure Hash Standard. *Federal Information Processing Standard* publication #180-1. U.S. Department of Commerce, National Institute of Standards and Technology, 1995.
6. Digital Signature Standard (DSS). *Federal Information Processing Standards* publication #186-2. U.S. Department of Commerce, National Institute of Standards and Technology, 2000.
7. D. Bleichenbacher. Generating ElGamal Signatures without Knowing the Secret Key. In *Advances in Cryptology EUROCRYPT'96*, Zaragoza, Spain, Lecture Notes in Computer Science 1070, pp. 10–18, Springer-Verlag, 1996.
8. W. Diffie, M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, vol. IT-22, pp. 644–654, 1976.
9. T. ElGamal. Cryptography and Logarithms over Finite Fields. PhD Thesis, Stanford University, 1984.

10. T. ElGamal. A Public-key Cryptosystem and a Signature Scheme based on Discrete Logarithms. In *Advances in Cryptology CRYPTO'84*, Santa Barbara, California, U.S.A., Lecture Notes in Computer Science 196, pp. 10–18, Springer-Verlag, 1985.
11. T. ElGamal. A Public-key Cryptosystem and a Signature Scheme based on Discrete Logarithms. *IEEE Transactions on Information Theory*, vol. IT-31, pp. 469–472, 1985.
12. S. Goldwasser, S. Micali, R.L. Rivest. A "Paradoxical" Solution to the Signature Problem. In *Advances in Cryptology CRYPTO'84*, Santa Barbara, California, U.S.A., Lecture Notes in Computer Science 196, pp. 467, Springer-Verlag, 1985.
13. S. Goldwasser, S. Micali, R.L. Rivest. A Digital Signature Scheme Secure against Adaptive Chosen-Message Attacks. *SIAM Journal on Computing*, vol. 17, pp. 281–308, 1988.
14. R. Housley, W. Ford, W. Polk, D. Solo. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. Internet Standard. RFC 2459, 1999.
15. N. Koblitz. CM-Curves with good Cryptographic Properties. In *Advances in Cryptology CRYPTO'91*, Santa Barbara, California, U.S.A., Lecture Notes in Computer Science 576, pp. 279–287, Springer-Verlag, 1992.
16. A. Menezes, N. Smart. Security of Signature Schemes in a Multi-User Setting. To appear in Designs, Codes and Cryptography.
17. R. C. Merkle. Secure Communications over Insecure Channels. *Communications of the ACM*, vol. 21, pp. 294–299, 1978.
18. J. Monnerat. Computation of the Discrete Logarithm on Elliptic Curves of Trace One — Tutorial. Technical report EPFL/IC/2002/49, EPFL, 2002.
19. R. L. Rivest, A. Shamir and L. M. Adleman. A Method for Obtaining Digital Signatures and Public-key Cryptosystem. In *Communications of the ACM*, vol. 21, pp. 120–126, 1978.
20. C. P. Schnorr. Efficient Identification and Signature for Smart Cards. In *Advances in Cryptology CRYPTO'89*, Santa Barbara, California, U.S.A., Lecture Notes in Computer Science 435, pp. 235–251, Springer-Verlag, 1990.
21. C. P. Schnorr. Efficient Identification and Signature for Smart Cards. *Journal of Cryptology*, vol. 4, pp. 161–174, 1991.
22. N. P. Smart. The Discrete Logarithm Problem on Elliptic Curves of Trace One. *Journal of Cryptology*, vol. 12, pp. 193–196, 1999.
23. S. Vaudenay. Hidden Collisions on DSS. In *Advances in Cryptology CRYPTO'96*, Santa Barbara, California, U.S.A., Lecture Notes in Computer Science 1109, pp. 83–88, Springer-Verlag, 1996.
24. S. Vaudenay. The Security of DSA and ECDSA — Bypassing the Standard Elliptic Curve Certification Scheme. In *Public Key Cryptography'03*, Miami, Florida, USA, Lecture Notes in Computer Science 2567, pp. 309–323, Springer-Verlag, 2003.