ATTACKING (EC)DSA WITH PARTIALLY KNOWN MULTIPLES OF NONCES

M.Adamoudis, K.A. Draziotis and D. Poulakis

Athecrypt
January 23, 2021
Athens, Greece

(EC)DSA background

- **Digital Signature Algorithm (DSA)** is a public-key signature scheme developed by NSA (the U.S. National Security Agency). It was proposed by NIST (the U.S. National Institute of Standards and Technology) back in 1991 and has become a FIPS 186 (U.S.Federal Information Processing Standard) called DSS (Digital Signature Standard).

(EC)DSA background

- **Digital Signature Algorithm (DSA)** is a public-key signature scheme developed by NSA (the U.S. National Security Agency). It was proposed by NIST (the U.S. National Institute of Standards and Technology) back in 1991 and has become a FIPS 186 (U.S.Federal Information Processing Standard) called DSS (Digital Signature Standard).
- In 1998, an elliptic curve analogue called Elliptic Curve Digital Signature Algorithm (ECDSA) was proposed and standarized.

(EC)DSA background

- Discrete Logarithm Problem for a group G

  Let $G = \langle g \rangle$ be a cyclic (multiplicative) group of order a prime p. Then the Discrete Logarithm Problem (DLP) is defined as follows: given (G, p, g, g^x) for a uniform random $x \leftarrow \mathbb{Z}_p$, find out x.

(EC)DSA background

- DISCRETE LOGARITHM PROBLEM FOR A GROUP G

  Let $G = \langle g \rangle$ be a cyclic (multiplicative) group of order a prime p. Then the Discrete Logarithm Problem (DLP) is defined as follows: given (G, p, g, g^x) for a uniform random $x \leftarrow \mathbb{Z}_p$, find out x.
- For DSA we use $G = \mathbb{Z}_p^*$ and for the Elliptic Curve DSA we use the group $G = E(\mathbf{F})$ for some elliptic curve E defined over a finite group $\mathbf{F}$.

(EC)DSA background

- PARAMETERS OF DSA.
  1. (p, q) primes in $\{1024, 2048, 3072\} \times \{160, 224, 256\}$ with $q|p-1$.
  2. g : a generator of the prime order q subgroup G of the multiplicative group $\mathbb{F}_p^*$.
  3. $a \xleftarrow{\$} \{1, \ldots, q-1\}$.
  4. $R = g^a \mod p$.
  5. Public key : (p, q, g, R).
  6. Private key : a.

(EC)DSA background

- SIGNING
  To sign a message $m \in \{0,1\}^*$,a user perform following these steps
  1. Publishes a hash function $h : \{0,1\}^* \to \{0,\dots,q-1\}$
  2. $k \xleftarrow{\$} \{1,\dots,q-1\}$ which is the ephemeral key
  3. Computes $r = (g^k \mod p) \mod q$ and
  $$s = k^{-1}(h(m) + ar) \mod q$$
  4. The signature of m is the pair (r, s).

(EC)DSA background

- VERIFICATION
  The signature is valid if and only if we have:

$$r = \left(\left(g^{s^{-1}h(m) \bmod q} R^{s^{-1}r \bmod q}\right) \bmod p\right) \bmod q.$$

(EC)DSA background

- PARAMETERS OF ECDSA
  1. Let E be an elliptic curve over $\mathbb{F}_p$
  2. $P \in E(\mathbb{F}_p)$ with order a prime q of size at least 160 bits and with $q|p-1$.
  3. $a \xleftarrow{\$} \{1, \ldots, q-1\}$.
  4. $Q = aP$.
  5. Public key : (E, p, q, P, Q).
  6. Private key : a.

(EC)DSA background

- SIGNING
  To sign a message $m \in \{0,1\}^*$, follow these steps:
- 1. Publish a hash function $h : \{0,1\}^* \to \{0,\dots,q-1\}$.

(EC)DSA background

- SIGNING
  To sign a message $m \in \{0,1\}^*$, follow these steps:
- 1. Publish a hash function $h : \{0,1\}^* \to \{0,\ldots,q-1\}$.
- 2. $k \xleftarrow{\$} \{1,\ldots,q-1\}$ which is the ephemeral key.

(EC)DSA background

- Signing
  To sign a message $m \in \{0,1\}^*$, follow these steps:
- 1. Publish a hash function $h : \{0,1\}^* \to \{0,\dots,q-1\}$.
- 2. $k \xleftarrow{\$} \{1,\dots,q-1\}$ which is the ephemeral key.
- 3. Compute $kP = (x, y)$ (where x and y are regarded as integers between 0 and $p-1$).

(EC)DSA background

- SIGNING
  To sign a message $m \in \{0,1\}^*$, follow these steps:
- 1. Publish a hash function $h : \{0,1\}^* \to \{0,\dots,q-1\}$.
- 2. $k \xleftarrow{\$} \{1,\dots,q-1\}$ which is the ephemeral key.
- 3. Compute $kP = (x, y)$ (where x and y are regarded as integers between 0 and $p-1$).
- 4. Compute $r = x \bmod q$ and

$$s = k^{-1}(h(m) + ar) \bmod q$$

The signature of m is (r, s).

(EC)DSA background

- VERIFICATION

  For the verification procedure we calculate,

$$u_1 = s^{-1}h(m) \bmod q, \quad u_2 = s^{-1}r \bmod q, \quad u_1P + u_2Q = (x_0, y_0).$$

We accept the signature if and only if $r = x_0 \mod q$.

(EC)DSA background

- (EC)DSA ATTACKS IN DISCRETE LOGARITHM
  1. For classic DSA we have subexponential algorithm (e.g. Index Calculus method,General Number Field Sieve).
  2. For ECDSA we have only exponential algorithms (e.g. Pollard Rho, Shank's Algorithm).

(EC)DSA background

- (EC)DSA ATTACKS ON SIGNING EQUATION

$$s = k^{-1}(h(m) + ar) \bmod q.$$

(EC)DSA background

- (EC)DSA ATTACKS ON SIGNING EQUATION

$$s = k^{-1}(h(m) + ar) \bmod q.$$

- These attacks work for both classic DSA and ECDSA.

(EC)DSA background

- (EC)DSA ATTACKS ON SIGNING EQUATION

$$s = k^{-1}(h(m) + ar) \bmod q.$$

- These attacks work for both classic DSA and ECDSA.
- Attacks on signing equation are based on lattice theory and the goal is to solve a linear system of congruences where unknown variables are the private key a and the ephemeral keys (or some multiples of them).

(EC)DSA background

- (EC)DSA ATTACKS ON SIGNING EQUATION

$$s = k^{-1}(h(m) + ar) \bmod q.$$

- These attacks work for both classic DSA and ECDSA.
- Attacks on signing equation are based on lattice theory and the goal is to solve a linear system of congruences where unknown variables are the private key a and the ephemeral keys (or some multiples of them).
- To apply these attacks we need some (polynomial) number of signatures (r_i, s_i).

(EC)DSA background

There are many papers that apply attacks to signing equation using lattice based methods.

1. 2001, Howgrave-Graham and Smart, *Lattice Attacks on Digital Signature Schemes.*
2. 2002, Blake and Garefalakis, *On the security of the digital signature algorithm.*
3. 2002, Nguyen and Shparlinski, *The Insecurity of the Digital Signature Algorithm with Partially Known Nonces.*
4. 2003, Nguyen and Shparlinski, *The Insecurity of the Elliptic Curve Digital Signature Algorithm with Partially Known Nonces.*
5. 2013, Liu and Nguyen, *Solving BDD by Enumeration: An Update.*
6. 2013, Draziotis and Poulakis, *Lattice attacks on DSA schemes based on Lagrange's algorithm.*
7. 2014, Faugere, Goyet and Renault, *Attacking (EC)DSA Given Only an Implicit Hint, Selected Area of Cryptography.*
8. 2016, Poulakis, *New lattice attacks on DSA schemes.*

Lattices

- **Lattices**
  Let $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$ linearly independent vectors of $\mathbb{R}^m$. The set

$$\mathcal{L} = \left\{ \sum_{j=1}^{n} \alpha_j \mathbf{b}_j : \alpha_j \in \mathbb{Z}, 1 \leq j \leq n \right\}$$

is called a *lattice* and the set $\mathcal{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ a basis of $\mathcal{L}$.

Lattices

- **Approximate Closest Vector Problem**
  We define the approximate Closest Vector Problem ($CVP_{\gamma_n}(L)$) as follows: Given a lattice $\mathcal{L} \subset \mathbb{Z}^m$ of rank n and a vector $\mathbf{t} \in \mathbb{R}^m$, find a vector $\mathbf{u} \in \mathcal{L}$ such that, for every $\mathbf{u}' \in \mathcal{L}$ we have:

$$\|\mathbf{u} - \mathbf{t}\| \leq \gamma_n \|\mathbf{u}' - \mathbf{t}\| \quad \text{(for some real number } \gamma_n \geq 1\text{)}.$$

Lattices

- **Approximate Closest Vector Problem**
  We define the approximate Closest Vector Problem ($CVP_{\gamma_n}(L)$) as follows: Given a lattice $\mathcal{L} \subset \mathbb{Z}^m$ of rank n and a vector $\mathbf{t} \in \mathbb{R}^m$, find a vector $\mathbf{u} \in \mathcal{L}$ such that, for every $\mathbf{u}' \in \mathcal{L}$ we have:

$$\|\mathbf{u} - \mathbf{t}\| \leq \gamma_n \|\mathbf{u}' - \mathbf{t}\| \text{ (for some real number } \gamma_n \geq 1).$$

- We say that we have a CVP oracle, if we have an efficient probabilistic algorithm that solves CVP_{γ_n} for $\gamma_n = 1$.

Babai's Algorithm

- Is a polynomial bit-operations algorithm that given a lattice and a target vector not in lattice, provides a lattice vector that is quite *close* to the target vector.

Babai's Algorithm

- Is a polynomial bit-operations algorithm that given a lattice and a target vector not in lattice, provides a lattice vector that is quite *close* to the target vector.
- On input a lattice $\mathcal{L}$ and a vector $\mathbf{t} \in \mathbb{R}^m$ the algorithms provides a lattice vector $\mathbf{x} \in L$ such that

$$||\mathbf{x} - \mathbf{t}|| \leq 2^{n/2} dist(L, \mathbf{t}).$$

Construction of a (EC)DSA system

- Say we have n messages m_i $(i = 1, \ldots, n)$ signed with (EC)DSA system and (r_i, s_i) their signatures. So we have the n signing equations:

$$s_i = k_i^{-1}(h(m_i) + ar_i) \bmod q,$$

where k_i are the ephemeral keys and a is the secret key.

Construction of a (EC)DSA system

- We choose integers

$$A_i \xleftarrow{\$} \Big(\frac{q^{\frac{i}{n+1}+f_q(n)}}{2}, \frac{q^{\frac{i}{n+1}+f_q(n)}}{1.5}\Big),$$

for a suitable sequence $f_q(n) < 1$ and we set $C_i = -r_i s_i^{-1} \mod q$, and

$$B_i = -A_i C_i^{-1} s_i^{-1} h(m_i) \mod q.$$

Construction of a (EC)DSA system

- We choose integers

$$A_i \xleftarrow{\$} \Big(\frac{q^{\frac{i}{n+1}+f_q(n)}}{2}, \frac{q^{\frac{i}{n+1}+f_q(n)}}{1.5}\Big),$$

for a suitable sequence $f_q(n) < 1$ and we set $C_i = -r_i s_i^{-1} \mod q$, and

$$B_i = -A_i C_i^{-1} s_i^{-1} h(m_i) \mod q.$$

- Further we set $\mathbf{s} = (a, k_1', \ldots, k_n')$, where $k_i' = A_i C_i^{-1} k_i \mod q$ and we call them *derivative ephemeral keys* (these are multiples of the unknown ephemeral keys).

Construction of a (EC)DSA system

- We choose integers

$$A_i \xleftarrow{\$} \left(\frac{q^{\frac{i}{n+1}+f_q(n)}}{2}, \frac{q^{\frac{i}{n+1}+f_q(n)}}{1.5}\right),$$

for a suitable sequence $f_q(n) < 1$ and we set $C_i = -r_i s_i^{-1} \mod q$, and

$$B_i = -A_i C_i^{-1} s_i^{-1} h(m_i) \mod q.$$

- Further we set $\mathbf{s} = (a, k'_1, \ldots, k'_n)$, where $k'_i = A_i C_i^{-1} k_i \mod q$ and we call them *derivative ephemeral keys* (these are multiples of the unknown ephemeral keys).
- After simple manipulations we get that $\mathbf{s}$ satisfies the $n \times (n+1)$ linear system

$$y_i + A_i x + B_i \equiv 0 \,(\mathrm{mod}\ q) \quad (i = 1, \ldots, n).$$

Conditional Babai Attack

- **Definition 1**
  Let q be a prime with $\ell-$bits and $x, c \in \mathbb{Z}_q$. Let $\mathcal{A}$ be a probabilistic polynomial algorithm which accepts (c, x, ℓ, PK), where PK is the public key of (EC)DSA-scheme, and returns

Conditional Babai Attack

- **Definition 1**
  Let q be a prime with $\ell-$bits and $x, c \in \mathbb{Z}_q$. Let $\mathcal{A}$ be a probabilistic polynomial algorithm which accepts (c, x, ℓ, PK), where PK is the public key of (EC)DSA-scheme, and returns
- 0, if the binary length of $cx \bmod q$ is ℓ bits,

Conditional Babai Attack

- **Definition 1**
  Let q be a prime with $\ell-$bits and $x, c \in \mathbb{Z}_q$. Let $\mathcal{A}$ be a probabilistic polynomial algorithm which accepts (c, x, ℓ, PK), where PK is the public key of (EC)DSA-scheme, and returns
- 0, if the binary length of $cx \bmod q$ is ℓ bits,
- 1, if the binary length of $cx \bmod q$ is $\ell - 1$ bits,
- 2, if the binary length of $cx \bmod q$ is $< \ell - 1$ bits.
  We call such an oracle, length DSA oracle

Conditional Babai Attack

- **Definition 2**
  Let $\mathcal{B}$ be a probabilistic polynomial algorithm which accepts a pair (x, ℓ, PK), where PK is the public key of (EC)DSA-scheme and $x \in \mathbb{Z}_q$, and returns
- True, if the binary length of $q - x$ is $\ell - 1$ bits

Conditional Babai Attack

- **Definition 2**
  Let $\mathcal{B}$ be a probabilistic polynomial algorithm which accepts a pair (x, ℓ, PK), where PK is the public key of (EC)DSA-scheme and $x \in \mathbb{Z}_q$, and returns
- True, if the binary length of $q - x$ is $\ell - 1$ bits
- False, otherwise.
  We call such an oracle binary length DSA oracle

Conditional Babai Attack

- **Attack**
  Input : A public key (p, q, g, R) of a DSA scheme or a public key (E, p, q, P, Q) of a ECDSA scheme. Further, n signed messages are given.
  Output : The secret key a or Fail.

Conditional Babai Attack

- 1. construct the system

$$y_i + A_i x + B_i \equiv 0 \pmod{q} \quad (i = 1, \ldots, n).$$

Conditional Babai Attack

- 1. construct the system

$$y_i + A_i x + B_i \equiv 0 \pmod{q} \quad (i = 1, \ldots, n).$$

- 2 Let k_i' as previously the derivative ephemeral key corresponding to the nonce k_i.

Conditional Babai Attack

- **For** $i = 1, \ldots, n$,
  3a. if $\mathcal{A}(k_i') = 0$, **then**
  if $\mathcal{B}(k_i') =$ True, consider the congruence,

$$(-y_i) + (-A_i)x + (-B_i) \equiv 0 \pmod{q}.$$

else, consider the congruence,

$$(2^{\ell-2} - y_i) + (-A_i)x + (-2^{\ell-2} - B_i) \equiv 0 \pmod{q}.$$

3b. if $\mathcal{A}(k_i') = 1$, **then** do not modify the $i-$ equation.
3c. if $\mathcal{A}(k_i') = 2$, **then** consider the congruence,

$$(2^{\ell-2} + y_i) + A_i x + (-2^{\ell-2} + B_i) \equiv 0 \pmod{q}.$$

Conditional Babai Attack

- **3d.**Let $A'_1, \ldots, A'_n$ and $B'_1, \ldots, B'_n$ be the coefficients of variable x and the constant terms, respectively, of the congruences constructed in steps **3a**, **3b** and **3c**. Thus, we have the following system:

$$y_i + A'_i x + B'_i \equiv 0 \pmod{q} \quad (i = 1, \ldots, n).$$

Conditional Babai Attack

- **4.** Construct the lattice generated by the rows of the DSA matrix

$$A = \begin{bmatrix} -1 & A'_1 & A'_2 & \dots & A'_n \\ 0 & q & 0 & \dots & 0 \\ 0 & 0 & q & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & q \end{bmatrix}$$

Further, set $\mathbf{b} = (0, B'_1, \dots, B'_n)$ and
$\mathbf{e} = (2^{\ell-2} + 2^{\ell-3}, \dots, 2^{\ell-2} + 2^{\ell-3})$.

Conditional Babai Attack

- 5. Apply LLL on the rows of A, $B \leftarrow LLL(A)$.

Conditional Babai Attack

- 5. Apply LLL on the rows of A, $B \leftarrow LLL(A)$.
- 6. $\mathbf{w} = (w_1, ..., w_{n+1}) \leftarrow Babai(B, \mathbf{b} + \mathbf{e})$.

Conditional Babai Attack

- 5. Apply LLL on the rows of A, $B \leftarrow LLL(A)$.
- 6. $\mathbf{w} = (w_1, ..., w_{n+1}) \leftarrow Babai(B, \mathbf{b} + \mathbf{e})$.
- **7.** If the first coordinate w_1 of $\mathbf{w}$ satisfy $g^{w_1} = R$, (respectively $Q = w_1 P$) in $\mathbb{F}_p^*$, return w_1, else return fail.

Conditional Babai Attack

- *The case* $\mathcal{A}(k_i') = 0$. Assume without loss of generality that $\ell = 160$. We consider the following assumption:
  Assumption-1. All the derivative ephemeral keys have 160$-$*bits.*
  Then, we can exploit the fact that $q - a$ and $q - k_i'$ have at most 159$-$bits.

Conditional Babai Attack

- *The case* $\mathcal{A}(k_i') = 0$. Assume without loss of generality that $\ell = 160$. We consider the following assumption:
  Assumption-1. All the derivative ephemeral keys have $160-$*bits.*
  Then, we can exploit the fact that $q - a$ and $q - k_i'$ have at most $159-$bits.
- Construct the *DSA*-system as follows:
  3a. if $\mathcal{B}(k_i') =$ True, consider the congruence,

$$(-y_i) + A_i(-x) + (-B_i) \equiv 0 \pmod{q}.$$

else, consider the congruence,

$$(2^{\ell-2} - y_i) + A_i(-x) + (-2^{\ell-2} - B_i) \equiv 0 \pmod{q}.$$

Conditional Babai Attack

- The previous attack is based on the following Theorem.

Conditional Babai Attack

- The previous attack is based on the following Theorem.
- **Theorem.**
  Let $\mathbf{s} = (a, A'_1 C_1^{-1} k_1 \mod q, \ldots, A'_n C_n^{-1} k_n \mod q)$ the solution of the DSA system. If

$$\|\mathbf{s} - \mathbf{e}\| < \frac{1}{4}\, q^{\frac{n}{n+1} + f_q(n)}$$

  for some $\mathbf{e} \in \mathbb{R}^{n+1}$ then, $\mathbf{s} = \mathbf{w} - \mathbf{b}$, where $\mathbf{w} = CVP(B, \mathbf{b} + \mathbf{e})$.

Conditional Babai Attack

- The previous attack is based on the following Theorem.
- **Theorem.**
  Let $\mathbf{s} = (a, A'_1 C_1^{-1} k_1 \mod q, \ldots, A'_n C_n^{-1} k_n \mod q)$ the solution of the DSA system. If

$$\|\mathbf{s} - \mathbf{e}\| < \frac{1}{4}\, q^{\frac{n}{n+1} + f_q(n)}$$

  for some $\mathbf{e} \in \mathbb{R}^{n+1}$ then, $\mathbf{s} = \mathbf{w} - \mathbf{b}$, where $\mathbf{w} = CVP(B, \mathbf{b} + \mathbf{e})$.
- In our attack we used Babai, which behaves as a CVP oracle for moderate dimension.

Experimental Results

- We consider $n = 204$ messages. We generated 100 random DSA systems, with secret key 160 bits and derivative ephemeral keys are $< q$.

Experimental Results

- We consider $n = 204$ messages. We generated 100 random DSA systems, with secret key 160 bits and derivative ephemeral keys are $< q$.
- For preprocessing we used BKZ with blocksize 70.The time execution per example was about 1 minute in an I3 Intel CPU.

Experimental Results

- We consider $n = 204$ messages. We generated 100 random DSA systems, with secret key 160 bits and derivative ephemeral keys are $< q$.
- For preprocessing we used BKZ with blocksize 70.The time execution per example was about 1 minute in an I3 Intel CPU.
- $f_q(n) = \min\left\{\frac{1}{n+1}, \frac{\ln\left(-3q^{-\frac{1}{n+1}}+\sqrt{96+9q^{-\frac{2}{n+1}}}\right)-\ln 8}{\ln q}\right\} - 10^{-10}.$

Experimental Results

- We consider $n = 204$ messages. We generated 100 random DSA systems, with secret key 160 bits and derivative ephemeral keys are $< q$.
- For preprocessing we used BKZ with blocksize 70.The time execution per example was about 1 minute in an I3 Intel CPU.
- $f_q(n) = \min\left\{\frac{1}{n+1}, \frac{\ln\left(-3q^{-\frac{1}{n+1}}+\sqrt{96+9q^{-\frac{2}{n+1}}}\right)-\ln 8}{\ln q}\right\} - 10^{-10}.$
-

bits:Skey	suc.rate
160	64%

Experimental Results

- **All the derivative ephemeral keys have** $160-$ **bits.**

bits:(Skey, Der.Ep.keys)	suc.rate
$(160, 160)$	83%

The (wall) time execution per example was about 2 minutes in an I3 Intel CPU (this time is dominated by the preprocessing step). So, having a binary length oracle we can find the secret key.

Thank you!