Enhancing an attack to DSA schemes

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

8th International Conference on Algebraic Informatics
June 30 – July 4, 2019
Niš, Serbia
presented by M.Adamoudis

(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 unique 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 $h : \{0,1\}^* \to \{0,\ldots,q-1\}$
  2. $k \xleftarrow{\$} \{1,\ldots,q-1\}$ 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:

$$r = ((g^{s^{-1}h(m) \mathrm{mod} q} R^{s^{-1}r \text{ mod } q}) \text{ mod } p) \text{ mod } q.$$

(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. $a \stackrel{\$}{\leftarrow} \{1, \ldots, q-1\}$.
  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 $h : \{0,1\}^* \to \{0,\ldots,q-1\}$.

(EC)DSA background

- SIGNING
  To sign a message $m \in \{0,1\}^*$, follow these steps:
- 1. Publish a hash function $h : \{0,1\}^* \to \{0,\dots,q-1\}$.
- 2. $k \xleftarrow{\$} \{1,\dots,q-1\}$ 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 $h : \{0,1\}^* \to \{0,\ldots,q-1\}$.
- 2. $k \xleftarrow{\$} \{1,\ldots,q-1\}$ 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 (Index Calculus method).
  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. 2003, Nguyen and Shparlinski, *The Insecurity of the Elliptic Curve Digital Signature Algorithm with Partially Known Nonces.*

4. 2013, Liu and Nguyen, *Solving BDD by Enumeration: An Update.*

5. 2013, Draziotis and Poulakis, *Lattice attacks on DSA schemes based on Lagrange's algorithm.*

6. 2014, Faugere, Goyet and Renault, *Attacking (EC)DSA Given Only an Implicit Hint, Selected Area of Cryptography.*

7. 2016, Poulakis, *New lattice attacks on DSA schemes.*

(EC)DSA background

We shall generalize the results of the following paper,
7. 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:

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

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 *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 *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

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

for a suitable sequence $f_q(n) < 1$ and we choose $C_i = -r_i s_i^{-1} \mod q$, 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 choose $C_i = -r_i s_i^{-1} \mod q$, 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 $k'_i = A_1 C_1^{-1} k_i \mod q$ and we call them *derivative ephemeral keys* (these are multiples of the unknown ephemeral keys).

Construction of a (EC)DSA system

- We choose integers
$$A_i \xleftarrow{\$} \Big(\frac{q^{\frac{i}{n+1}+f_q(n)}}{2}, \frac{q^{\frac{i}{n+1}+f_q(n)}}{1.5}\Big),$$
for a suitable sequence $f_q(n) < 1$ and we choose $C_i = -r_i s_i^{-1} \mod q$, 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 $k_i' = A_1 C_1^{-1} k_i \mod q$ 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
$$y_i + A_i x + B_i \equiv 0 \,(\mathrm{mod}\ q) \quad (i = 1, \ldots, n).$$

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.

Attack

- 1. construct the system

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

and set $\mathbf{b} = (0, B_1, \ldots, B_n)$.

Attack

- 1. construct the system

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

  and set $\mathbf{b} = (0, B_1, \ldots, B_n)$.

- 2. Construct the lattice generated by the rows of the DSA matrix

$$A = \begin{bmatrix} -1 & A_1 & A_2 & \ldots & A_n \\ 0 & q & 0 & \ldots & 0 \\ 0 & 0 & q & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & q \end{bmatrix}$$

Attack

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

Attack

- 3. Apply LLL on the rows of A, $B \leftarrow LLL(A)$.
- 4. $\mathbf{s} = (s_1, ..., s_{n+1}) \leftarrow Babai(B, \mathbf{b})$.

Attack

- 3. Apply LLL on the rows of A, $B \leftarrow LLL(A)$.
- 4. $\mathbf{s} = (s_1, ..., s_{n+1}) \leftarrow Babai(B, \mathbf{b})$.
- 5. If $g^{s_1} = R$ (respectively $Q = s_1 P$) return s_1 else return *fail*.

Attack

- The previous attack is based on the following Theorem.

Attack

- The previous attack is based on the following Theorem.
- **Theorem.**
  If
  $$\|\mathbf{s}\| < \frac{1}{4}\, q^{\frac{n}{n+1}+f_q(n)}.$$
  then, $\mathbf{s} = \mathbf{w} - \mathbf{b}$, where $\mathbf{w} = CVP(B, \mathbf{b})$.

Attack

- The previous attack is based on the following Theorem.
- **Theorem.**
  If
  $$\|\mathbf{s}\| < \frac{1}{4}\, q^{\frac{n}{n+1}+f_q(n)}.$$
  then, $\mathbf{s} = \mathbf{w} - \mathbf{b}$, where $\mathbf{w} = CVP(B, \mathbf{b})$.
- In our attack we used Babai, which behaves as a CVP oracle for moderate dimension.

Results

- We implemented the previous attack.

Results

- We implemented the previous attack.
- Since Babai does not always provide the closest vector and also the integers A_i are chosen randomly, we get a probabilistic attack.

Results

- We implemented the previous attack.
- Since Babai does not always provide the closest vector and also the integers A_i are chosen randomly, we get a probabilistic attack.
- Further, we tested our attack for solutions that does not satisfy the theorem (i.e. for larger keys).

Results

- We consider $n = 14$ signed messages, secret key with $147-$ bits (i.e. small enough, comparing with its original length which is $160-$ bits) and derivative ephemeral keys with binary length 145 bits.

Results

- We consider $n = 14$ signed messages, secret key with $147-$ bits (i.e. small enough, comparing with its original length which is $160-$ bits) and derivative ephemeral keys with binary length 145 bits.
- Then, we managed to find always the secret key.

Results

- We consider $n = 14$ signed messages, secret key with $147-$ bits (i.e. small enough, comparing with its original length which is $160-$ bits) and derivative ephemeral keys with binary length 145 bits.
- Then, we managed to find always the secret key.
-

bits:(Skey, Der.Ep.keys)	suc.rate
$(147, 145)$	100%

Results

- We consider $n = 206$ signed messages. We generated 100 random DSA systems. For preprocessing (i.e. before we apply Babai) we used BKZ with blocksize 70.

Results

- We consider $n = 206$ signed messages. We generated 100 random DSA systems. For preprocessing (i.e. before we apply Babai) we used BKZ with blocksize 70.
-

bits:(Skey, Der.Ep.keys)	suc.rate
$(158, 157)$	17%
$(158, 155)$	100%
$(157, 157)$	23.3%
$(157, 156)$	100%

Heuristic Improvement of the previous attack

- We can further improve the previous results. The idea is to use another target vector instead of $\mathbf{b} = (0, B_1, \ldots, B_n)$. We consider the following vector
$$\mathbf{b} = (\varepsilon, \varepsilon + B_1, \ldots, \varepsilon + B_n),$$
where $\varepsilon = 2^{159} - 2^{157}$.

Heuristic Improvement

- We consider $n = 206$ messages. We generated 100 random DSA systems, with secret key 160 bits and derivative ephemeral keys with 159 bits.

Heuristic Improvement

- We consider $n = 206$ messages. We generated 100 random DSA systems, with secret key 160 bits and derivative ephemeral keys with 159 bits.
- For preprocessing we used BKZ with blocksize 85.The time execution per example was about 2 minutes in an I3 Intel CPU.

Heuristic Improvement

- We consider $n = 206$ messages. We generated 100 random DSA systems, with secret key 160 bits and derivative ephemeral keys with 159 bits.
- For preprocessing we used BKZ with blocksize 85.The time execution per example was about 2 minutes in an I3 Intel CPU.
-

bits:(Skey, Der.Ep.keys)	suc.rate
$(160, 159)$	62%

Heuristic Improvement

- We consider $n = 206$ messages. We generated 100 random DSA systems, with secret key 160 bits and derivative ephemeral keys with 159 bits.
- For preprocessing we used BKZ with blocksize 85.The time execution per example was about 2 minutes in an I3 Intel CPU.
-

bits:(Skey, Der.Ep.keys)	suc.rate
$(160, 159)$	62%

- This result improves, in some sense, the result of Liu and Nguyen, where with 100 signatures and knowing 2 least significant bits of the ephemeral keys, they computed the secret key with success rate 23% and in 4185 seconds on average per instance

Thank you!