GLV/GLS Decomposition, Power Analysis, and Attacks on ECDSA Signatures With Single-Bit Nonce Bias

Diego F. Aranha   Pierre-Alain Fouque   Benoît Gerard
Jean-Gabriel Kammerer   Mehdi Tibouchi
Jean-Christophe Zapalowicz

NTT Secure Platform Laboratories

Asiacrypt, 2014–12–08

Overview

- Apology: although we have "power analysis" in the title and this is a side-channel session, SCA is not our main focus.
- Twofold motivation:
  - Break the 2-bit bias barrier of HNP lattice attacks on (EC)DSA
  - Consider how such attacks apply to elliptic curves with fast endomorphisms

Overview

- Apology: although we have "power analysis" in the title and this is a side-channel session, SCA is not our main focus.
- Twofold motivation:
  - Break the 2-bit bias barrier of HNP lattice attacks on (EC)DSA
  - Consider how such attacks apply to elliptic curves with fast endomorphisms

Outline

Attack on ECDSA with 1-bit nonce bias

- HNP attacks on ECDSA
- Our attack on 1-bit bias

GLV/GLS decomposition and HNP

- Curves with fast endomorphisms
- The recomposition approach
- The decomposition approach

Conclusion

Outline

Attack on ECDSA with 1-bit nonce bias

- HNP attacks on ECDSA
- Our attack on 1-bit bias

GLV/GLS decomposition and HNP

- Curves with fast endomorphisms
- The recomposition approach
- The decomposition approach

Conclusion

Outline

Attack on ECDSA with 1-bit nonce bias

- HNP attacks on ECDSA
- Our attack on 1-bit bias

GLV/GLS decomposition and HNP

- Curves with fast endomorphisms
- The recomposition approach
- The decomposition approach

Conclusion

ECDSA

- Most commonly used elliptic curve-based signature scheme
- Slightly contrived variant of Schnorr signatures (all results in this talk also apply to actual Schnorr and similar schemes)
- Description
  - Public params: elliptic curve $E/\mathbb{F}_q$, point P generating a subgroup of large known prime order n
  - Key pair: private key x random in $\mathbb{Z}/n\mathbb{Z}$, public key $Q = [x]P$
  - Signature on m: pair (r, s) computed as

    $k \xleftarrow{\$} (\mathbb{Z}/n\mathbb{Z})^*$
    $(u, v) \leftarrow [k]P$
    $r \leftarrow u \bmod n$
    $s \leftarrow k^{-1}(H(m) + rx) \bmod n$
- Randomness k usually called the nonce; indeed, it *must not* be reused, otherwise:

$$x = (sh' - s'h)/(s'r - sr') \bmod n$$

but a bit of a misnomer, because non-repetition is not enough.

ECDSA

- Most commonly used elliptic curve-based signature scheme
- Slightly contrived variant of Schnorr signatures (all results in this talk also apply to actual Schnorr and similar schemes)
- Description
  - Public params: elliptic curve $E/\mathbb{F}_q$, point P generating a subgroup of large known prime order n
  - Key pair: private key x random in $\mathbb{Z}/n\mathbb{Z}$, public key $Q = [x]P$
  - Signature on m: pair (r, s) computed as

    $k \xleftarrow{\$} (\mathbb{Z}/n\mathbb{Z})^*$
    $(u, v) \leftarrow [k]P$
    $r \leftarrow u \bmod n$
    $s \leftarrow k^{-1}(H(m) + rx) \bmod n$
- Randomness k usually called the nonce; indeed, it *must not* be reused, otherwise:

$$x = (sh' - s'h)/(s'r - sr') \bmod n$$

but a bit of a misnomer, because non-repetition is not enough.

ECDSA

- Most commonly used elliptic curve-based signature scheme
- Slightly contrived variant of Schnorr signatures (all results in this talk also apply to actual Schnorr and similar schemes)
- Description
  - Public params: elliptic curve $E/\mathbb{F}_q$, point P generating a subgroup of large known prime order n
  - Key pair: private key x random in $\mathbb{Z}/n\mathbb{Z}$, public key $Q = [x]P$
  - Signature on m: pair (r, s) computed as

    $k \xleftarrow{\$} (\mathbb{Z}/n\mathbb{Z})^*$
    $(u, v) \leftarrow [k]P$
    $r \leftarrow u \bmod n$
    $s \leftarrow k^{-1}(H(m) + rx) \bmod n$

- Randomness k usually called the nonce; indeed, it *must not* be reused, otherwise:

$$x = (sh' - s'h)/(s'r - sr') \bmod n$$

but a bit of a misnomer, because non-repetition is not enough.

ECDSA

- Most commonly used elliptic curve-based signature scheme
- Slightly contrived variant of Schnorr signatures (all results in this talk also apply to actual Schnorr and similar schemes)
- Description
  - Public params: elliptic curve $E/\mathbb{F}_q$, point P generating a subgroup of large known prime order n
  - Key pair: private key x random in $\mathbb{Z}/n\mathbb{Z}$, public key $Q = [x]P$
  - Signature on m: pair (r, s) computed as

    $k \xleftarrow{\$} (\mathbb{Z}/n\mathbb{Z})^*$
    $(u, v) \leftarrow [k]P$
    $r \leftarrow u \bmod n$
    $s \leftarrow k^{-1}(H(m) + rx) \bmod n$
- Randomness k usually called the nonce; indeed, it *must not* be reused, otherwise:

$$x = (sh' - s'h)/(s'r - sr') \bmod n$$

but a bit of a misnomer, because non-repetition is not enough.

The risk of leaky nonces

- Rewrite the relation satisfied by ECDSA signatures as:

$$k = \underbrace{H(m)s^{-1}}_{h} + \underbrace{rs^{-1}}_{c} x \bmod n$$

- We know pairs (h, c) such that $h + cx = k \bmod n$. If we know "something" about k (such as its MSBs), it should translate to information about the private key x! Essentially the hidden number problem.
- Main attack due to Howgrave-Graham & Smart using lattice reduction (reduces HNP to CVP in a suitable lattice)
  - carried out for 2-bit leaks on 160-bit curves (Liu–Nguyen 2013)
  - currently out of reach for for 2-bit leaks on 256-bit curves
  - currently out of reach for for 3-bit leaks on 384-bit curves
  - hard limit: impossible for 1-bit leaks (the hidden vector in the lattice is not short enough to recover, even with a CVP oracle)

The risk of leaky nonces

- Rewrite the relation satisfied by ECDSA signatures as:

$$k = \underbrace{H(m)s^{-1}}_{h} + \underbrace{rs^{-1}}_{c} x \bmod n$$

- We know pairs (h, c) such that $h + cx = k \bmod n$. If we know "something" about k (such as its MSBs), it should translate to information about the private key x! Essentially the hidden number problem.
- Main attack due to Howgrave-Graham & Smart using lattice reduction (reduces HNP to CVP in a suitable lattice)
  - carried out for 2-bit leaks on 160-bit curves (Liu–Nguyen 2013)
  - currently out of reach for for 2-bit leaks on 256-bit curves
  - currently out of reach for for 3-bit leaks on 384-bit curves
  - hard limit: impossible for 1-bit leaks (the hidden vector in the lattice is not short enough to recover, even with a CVP oracle)

The risk of leaky nonces

- Rewrite the relation satisfied by ECDSA signatures as:

$$k = \underbrace{H(m)s^{-1}}_{h} + \underbrace{rs^{-1}}_{c} x \bmod n$$

- We know pairs (h, c) such that $h + cx = k \bmod n$. If we know "something" about k (such as its MSBs), it should translate to information about the private key x! Essentially the hidden number problem.
- Main attack due to Howgrave-Graham & Smart using lattice reduction (reduces HNP to CVP in a suitable lattice)
  - carried out for 2-bit leaks on 160-bit curves (Liu–Nguyen 2013)
  - currently out of reach for for 2-bit leaks on 256-bit curves
  - currently out of reach for for 3-bit leaks on 384-bit curves
  - hard limit: impossible for 1-bit leaks (the hidden vector in the lattice is not short enough to recover, even with a CVP oracle)

Bleichenbacher's attack

- Before even the lattice attack was proposed, Bleichenbacher suggested a different approach to HNP (in the context of DSA), based on a Fourier notion of bias
- Requires many more signatures than the lattice attack for the same parameters, but applies in principle to arbitrarily small biases
- Presented at an IEEE P1363 meeting in 2000, but never formally published. Reintroduced in a paper by De Mulder et al. at CHES 2013.

Bleichenbacher's attack

- Before even the lattice attack was proposed, Bleichenbacher suggested a different approach to HNP (in the context of DSA), based on a Fourier notion of bias
- Requires many more signatures than the lattice attack for the same parameters, but applies in principle to arbitrarily small biases
- Presented at an IEEE P1363 meeting in 2000, but never formally published. Reintroduced in a paper by De Mulder et al. at CHES 2013.

Bleichenbacher's attack

- Before even the lattice attack was proposed, Bleichenbacher suggested a different approach to HNP (in the context of DSA), based on a Fourier notion of bias
- Requires many more signatures than the lattice attack for the same parameters, but applies in principle to arbitrarily small biases
- Presented at an IEEE P1363 meeting in 2000, but never formally published. Reintroduced in a paper by De Mulder et al. at CHES 2013.

Overview of Bleicherbacher's technique

- The HNP problem reduces to the following: we are given samples (h_j, c_j) such that, for the hidden secret x, the MSBs of the values $k_j = h_j + c_j x$ vanish.
- The sampled bias of a set of points $V = (v_0, \cdots, v_{L-1})$ in $\mathbb{Z}/n\mathbb{Z}$ defined as $B_n(V) = \frac{1}{L}\sum_{j=0}^{L-1} e^{2\pi i \cdot v_j/n}$
- Given the (h_j, c_j), consider the vector $V = (v_j)$ given by $v_j = h_j + c_j \cdot w$ for some $w \in \mathbb{Z}/n\mathbb{Z}$. One can check that:
  - if $w \neq x$, $B_n(V) \approx 1/\sqrt{L}$ is negligible
  - if $w = x$, $B_n(V)$ is close to 1
  - hence a distinguisher, but not useable because n choices for w
- Bleichenbacher idea: broaden the peak in bias by reducing the c_j's.

Overview of Bleicherbacher's technique

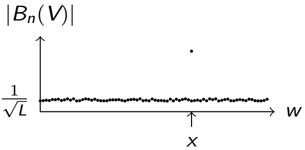
- The HNP problem reduces to the following: we are given samples (h_j, c_j) such that, for the hidden secret x, the MSBs of the values $k_j = h_j + c_j x$ vanish.
- The sampled bias of a set of points $V = (v_0, \cdots, v_{L-1})$ in $\mathbb{Z}/n\mathbb{Z}$ defined as $B_n(V) = \frac{1}{L}\sum_{j=0}^{L-1} e^{2\pi i \cdot v_j/n}$

- Given the (h_j, c_j), consider the vector $V = (v_j)$ given by $v_j = h_j + c_j \cdot w$ for some $w \in \mathbb{Z}/n\mathbb{Z}$. One can check that:
  - if $w \neq x$, $B_n(V) \approx 1/\sqrt{L}$ is negligible
  - if $w = x$, $B_n(V)$ is close to 1
  - hence a distinguisher, but not useable because n choices for w
- Bleichenbacher idea: broaden the peak in bias by reducing the c_j's.

Overview of Bleicherbacher's technique

- The HNP problem reduces to the following: we are given samples (h_j, c_j) such that, for the hidden secret x, the MSBs of the values $k_j = h_j + c_j x$ vanish.
- The sampled bias of a set of points $V = (v_0, \cdots, v_{L-1})$ in $\mathbb{Z}/n\mathbb{Z}$ defined as $B_n(V) = \frac{1}{L}\sum_{j=0}^{L-1} e^{2\pi i \cdot v_j/n}$

- Given the (h_j, c_j), consider the vector $V = (v_j)$ given by $v_j = h_j + c_j \cdot w$ for some $w \in \mathbb{Z}/n\mathbb{Z}$. One can check that:
  - if $w \neq x$, $B_n(V) \approx 1/\sqrt{L}$ is negligible
  - if $w = x$, $B_n(V)$ is close to 1
  - hence a distinguisher, but not useable because n choices for w

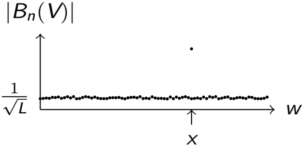


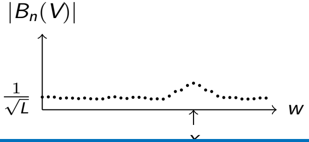
- Bleichenbacher idea: broaden the peak in bias by reducing the c_j's.

Overview of Bleicherbacher's technique

- The HNP problem reduces to the following: we are given samples (h_j, c_j) such that, for the hidden secret x, the MSBs of the values $k_j = h_j + c_j x$ vanish.
- The sampled bias of a set of points $V = (v_0, \cdots, v_{L-1})$ in $\mathbb{Z}/n\mathbb{Z}$ defined as $B_n(V) = \frac{1}{L}\sum_{j=0}^{L-1} e^{2\pi i \cdot v_j/n}$

- Given the (h_j, c_j), consider the vector $V = (v_j)$ given by $v_j = h_j + c_j \cdot w$ for some $w \in \mathbb{Z}/n\mathbb{Z}$. One can check that:
  - if $w \neq x$, $B_n(V) \approx 1/\sqrt{L}$ is negligible
  - if $w = x$, $B_n(V)$ is close to 1
  - hence a distinguisher, but not useable because n choices for w
- Bleichenbacher idea: broaden the peak in bias by reducing the c_j's.





Outline

Attack on ECDSA with 1-bit nonce bias

- HNP attacks on ECDSA
- Our attack on 1-bit bias

GLV/GLS decomposition and HNP

- Curves with fast endomorphisms
- The recomposition approach
- The decomposition approach

Conclusion

Implementing Bleichenbacher's algorithm (I)

- Key step of Bleicheinbacher's algorithm: reducing the c_j's by finding small linear combinations between them
- De Mulder et al. use lattice reduction
- How Bleichenbacher suggested doing it is not completely clear (iterative collision search on MSBs?)
- We take a straightforward sort-and-difference approach:
  - sort the (c_j, h_j) list according to c_j
  - substract each c_j from the next largest one
  - (repeat)
- Starting from a list of $L = 2^\ell$ samples, we reduce the size of the c_j's by roughly ℓ bits per iteration (justified by order statistic arguments)

Implementing Bleichenbacher's algorithm (I)

- Key step of Bleicheinbacher's algorithm: reducing the c_j's by finding small linear combinations between them
- De Mulder et al. use lattice reduction
- How Bleichenbacher suggested doing it is not completely clear (iterative collision search on MSBs?)
- We take a straightforward sort-and-difference approach:
  - sort the (c_j, h_j) list according to c_j
  - substract each c_j from the next largest one
  - (repeat)
- Starting from a list of $L = 2^\ell$ samples, we reduce the size of the c_j's by roughly ℓ bits per iteration (justified by order statistic arguments)

Implementing Bleichenbacher's algorithm (I)

- Key step of Bleicheinbacher's algorithm: reducing the c_j's by finding small linear combinations between them
- De Mulder et al. use lattice reduction
- How Bleichenbacher suggested doing it is not completely clear (iterative collision search on MSBs?)
- We take a straightforward sort-and-difference approach:
  - sort the (c_j, h_j) list according to c_j
  - substract each c_j from the next largest one
  - (repeat)
- Starting from a list of $L = 2^\ell$ samples, we reduce the size of the c_j's by roughly ℓ bits per iteration (justified by order statistic arguments)

Implementing Bleichenbacher's algorithm (I)

- Key step of Bleicheinbacher's algorithm: reducing the c_j's by finding small linear combinations between them
- De Mulder et al. use lattice reduction
- How Bleichenbacher suggested doing it is not completely clear (iterative collision search on MSBs?)
- We take a straightforward sort-and-difference approach:
  - sort the (c_j, h_j) list according to c_j
  - substract each c_j from the next largest one
  - (repeat)
- Starting from a list of $L = 2^\ell$ samples, we reduce the size of the c_j's by roughly ℓ bits per iteration (justified by order statistic arguments)

Implementing Bleichenbacher's algorithm (I)

- Key step of Bleicheinbacher's algorithm: reducing the c_j's by finding small linear combinations between them
- De Mulder et al. use lattice reduction
- How Bleichenbacher suggested doing it is not completely clear (iterative collision search on MSBs?)
- We take a straightforward sort-and-difference approach:
  - sort the (c_j, h_j) list according to c_j
  - substract each c_j from the next largest one
  - (repeat)
- Starting from a list of $L = 2^\ell$ samples, we reduce the size of the c_j's by roughly ℓ bits per iteration (justified by order statistic arguments)

Implementing Bleichenbacher's algorithm (II)

- Rest of the algorithm (as in De Mulder et al.):
  - Once the c_j's are short enough, carry out an FFT computation to find the peak
  - Rank the candidate peaks to reveal the MSBs of x, and iterate the attack to find the remaining bits
- Main difficulties:
  - Every reduction step squares the bias
  - The correct guess of x not always the highest ranked
  - On 1-bit bias, non-trivial engineering project: very costly in data, memory and CPU

Implementing Bleichenbacher's algorithm (II)

- Rest of the algorithm (as in De Mulder et al.):
  - Once the c_j's are short enough, carry out an FFT computation to find the peak
  - Rank the candidate peaks to reveal the MSBs of x, and iterate the attack to find the remaining bits
- Main difficulties:
  - Every reduction step squares the bias
  - The correct guess of x not always the highest ranked
  - On 1-bit bias, non-trivial engineering project: very costly in data, memory and CPU

Implementation results

b	1	2	3	4	5
$B_n(\mathbf{K})$	0.6366198	0.9003163	0.9744954	0.9935869	0.9983944

α	Fraction of c_j's reduced by $\ell - \beta$ bits in a list of 2^ℓ				
	$\beta = -2$	$\beta = -1$	$\beta = 0$	$\beta = 1$	$\beta = 2$
$1st$ iteration	0.22	0.39	0.63	0.86	0.98
$2nd$ iteration	0.031	0.12	0.36	0.75	0.94
$3rd$ iteration	$3.2\ 10^{-3}$	0.025	0.17	0.64	0.89
$4th$ iteration	$3.0\ 10^{-4}$	$4.6\ 10^{-3}$	0.069	0.53	0.84
$5th$ iteration	$2.0\ 10^{-5}$	$6.7\ 10^{-4}$	0.022	0.40	0.79

Implementation results

b	1	2	3	4	5
$B_n(\mathbf{K})$	0.6366198	0.9003163	0.9744954	0.9935869	0.9983944

α	Fraction of c_j's reduced by $\ell - \beta$ bits in a list of 2^ℓ				
	$\beta = -2$	$\beta = -1$	$\beta = 0$	$\beta = 1$	$\beta = 2$
$1st$ iteration	0.22	0.39	0.63	0.86	0.98
$2nd$ iteration	0.031	0.12	0.36	0.75	0.94
$3rd$ iteration	$3.2\ 10^{-3}$	0.025	0.17	0.64	0.89
$4th$ iteration	$3.0\ 10^{-4}$	$4.6\ 10^{-3}$	0.069	0.53	0.84
$5th$ iteration	$2.0\ 10^{-5}$	$6.7\ 10^{-4}$	0.022	0.40	0.79

Successfully implemented: SECG P160 R1 curve (C++, RELIC, FFTW)

- 2^{33} signatures
- 4 sort-and-difference (remove 4 $\times$ 32 bits)
- 52.5% reduced signatures
- 0.00072792 final bias
- FFT on 32 bits
- 30 MSB retrieved
- 1 terabyte
- 1150 CPU-hours

Outline

Attack on ECDSA with 1-bit nonce bias

- HNP attacks on ECDSA
- Our attack on 1-bit bias

GLV/GLS decomposition and HNP

- Curves with fast endomorphisms
- The recomposition approach
- The decomposition approach

Conclusion

Outline

Attack on ECDSA with 1-bit nonce bias

- HNP attacks on ECDSA
- Our attack on 1-bit bias

GLV/GLS decomposition and HNP

- Curves with fast endomorphisms
- The recomposition approach
- The decomposition approach

Conclusion

Curves with fast endomorphisms

- The most costly operation in conventional elliptic curve crypto is elliptic curve scalar multiplication; e.g. in ECDSA signature generation, the computation $[k]P$
- Special curves can be used to increase the efficiency of such schemes: curves endowed with some fast endomorphism ψ
- This technique is used in almost all recent record-breaking implementations of ECC
- Example special curves:
  - (Koblitz curves over binary fields)
  - Gallant–Lambert–Vanstone (GLV) curves over prime fields
  - Galbraith–Lin–Scott (GLS) curves over quadratic extensions
  - more recent work (Ben Smith's $\mathbb{Q}$-curves...)

Curves with fast endomorphisms

- The most costly operation in conventional elliptic curve crypto is elliptic curve scalar multiplication; e.g. in ECDSA signature generation, the computation $[k]P$
- Special curves can be used to increase the efficiency of such schemes: curves endowed with some fast endomorphism ψ
  - on a prime order subgroup, ψ is the multiplication by some explicit (usually full size) constant λ
  - to carry out scalar multiplication by k, write $k = k_1 + k_2\lambda$ (k_1, k_2 of half size); then $[k]P = [k_1]P + [k_2]\psi(P)$
  - double exponentiation: ≈ 1.7-fold speed-up
- This technique is used in almost all recent record-breaking implementations of ECC
- Example special curves:
  - (Koblitz curves over binary fields)
  - Gallant–Lambert–Vanstone (GLV) curves over prime fields
  - Galbraith–Lin–Scott (GLS) curves over quadratic extensions
  - more recent work (Ben Smith's $\mathbb{Q}$-curves...)

Curves with fast endomorphisms

- The most costly operation in conventional elliptic curve crypto is elliptic curve scalar multiplication; e.g. in ECDSA signature generation, the computation $[k]P$
- Special curves can be used to increase the efficiency of such schemes: curves endowed with some fast endomorphism ψ
- This technique is used in almost all recent record-breaking implementations of ECC
- Example special curves:
  - (Koblitz curves over binary fields)
  - Gallant–Lambert–Vanstone (GLV) curves over prime fields
  - Galbraith–Lin–Scott (GLS) curves over quadratic extensions
  - more recent work (Ben Smith's $\mathbb{Q}$-curves...)

Curves with fast endomorphisms

- The most costly operation in conventional elliptic curve crypto is elliptic curve scalar multiplication; e.g. in ECDSA signature generation, the computation $[k]P$
- Special curves can be used to increase the efficiency of such schemes: curves endowed with some fast endomorphism ψ
- This technique is used in almost all recent record-breaking implementations of ECC
- Example special curves:
  - (Koblitz curves over binary fields)
  - Gallant–Lambert–Vanstone (GLV) curves over prime fields
  - Galbraith–Lin–Scott (GLS) curves over quadratic extensions
  - more recent work (Ben Smith's $\mathbb{Q}$-curves...)

Decomposition vs. recomposition

- In many algorithms (including ECDSA), we want to compute a random scalar multiplication
- With endomorphisms, two natural approaches considered in the literature:
  - Decomposition: pick k at random, and then use an algorithm (lattice reduction, continued fractions, etc.) to find k_1, k_2 of half size such that $k = k_1 + k_2\lambda \bmod n$
  - Recomposition: pick k_1 and k_2 at random, implicitly choosing $k = k_1 + k_2\lambda \bmod n$
- We are interested in the implications of these two approaches on vulnerability to HNP-type attacks

Decomposition vs. recomposition

- In many algorithms (including ECDSA), we want to compute a random scalar multiplication
- With endomorphisms, two natural approaches considered in the literature:
  - Decomposition: pick k at random, and then use an algorithm (lattice reduction, continued fractions, etc.) to find k_1, k_2 of half size such that $k = k_1 + k_2\lambda \bmod n$
  - Recomposition: pick k_1 and k_2 at random, implicitly choosing $k = k_1 + k_2\lambda \bmod n$
- We are interested in the implications of these two approaches on vulnerability to HNP-type attacks

Decomposition vs. recomposition

- In many algorithms (including ECDSA), we want to compute a random scalar multiplication
- With endomorphisms, two natural approaches considered in the literature:
  - Decomposition: pick k at random, and then use an algorithm (lattice reduction, continued fractions, etc.) to find k_1, k_2 of half size such that $k = k_1 + k_2\lambda \bmod n$
  - Recomposition: pick k_1 and k_2 at random, implicitly choosing $k = k_1 + k_2\lambda \bmod n$
- We are interested in the implications of these two approaches on vulnerability to HNP-type attacks

Decomposition vs. recomposition

- In many algorithms (including ECDSA), we want to compute a random scalar multiplication
- With endomorphisms, two natural approaches considered in the literature:
  - Decomposition: pick k at random, and then use an algorithm (lattice reduction, continued fractions, etc.) to find k_1, k_2 of half size such that $k = k_1 + k_2\lambda \bmod n$
  - Recomposition: pick k_1 and k_2 at random, implicitly choosing $k = k_1 + k_2\lambda \bmod n$
- We are interested in the implications of these two approaches on vulnerability to HNP-type attacks

Decomposition vs. recomposition

- In many algorithms (including ECDSA), we want to compute a random scalar multiplication
- With endomorphisms, two natural approaches considered in the literature:
  - Decomposition: pick k at random, and then use an algorithm (lattice reduction, continued fractions, etc.) to find k_1, k_2 of half size such that $k = k_1 + k_2\lambda \bmod n$
  - Recomposition: pick k_1 and k_2 at random, implicitly choosing $k = k_1 + k_2\lambda \bmod n$
- We are interested in the implications of these two approaches on vulnerability to HNP-type attacks

Outline

Recomposition with GLS

- Recomposition certainly presents more interesting theoretical challenges
- We look at the specific case of curves generated with the quadratic GLS method:
  - E_0 over a prime field $\mathbb{F}_p$; order: $p+1-t, |t| \leqslant 2\sqrt{p}$
  - E quadratic twist of E_0 over $\mathbb{F}_{p^2}$
  - we assume further that $E(\mathbb{F}_{p^2})$ is of prime order n; then: $n = (p-1)^2 + t^2$ and $\lambda = \sqrt{-1} = t^{-1}(p-1) \bmod n$
- In this setting, two possible ways of carrying out recomposition:
  - Careful way: pick k_1, k_2 uniformly at random in $[0, \sqrt{n})$. This is secure!
  - Careless way: pick k_1, k_2 uniformly at random in $[0, 2^m)$ with $m = \lfloor \frac{1}{2} \log_2 n \rfloor$. Can be broken with Bleichebacher's attack!

Recomposition with GLS

- Recomposition certainly presents more interesting theoretical challenges
- We look at the specific case of curves generated with the quadratic GLS method:
  - E_0 over a prime field $\mathbb{F}_p$; order: $p + 1 - t, |t| \leqslant 2\sqrt{p}$
  - E quadratic twist of E_0 over $\mathbb{F}_{p^2}$
  - we assume further that $E(\mathbb{F}_{p^2})$ is of prime order n; then: $n = (p-1)^2 + t^2$ and $\lambda = \sqrt{-1} = t^{-1}(p-1) \bmod n$
- In this setting, two possible ways of carrying out recomposition:
  - Careful way: pick k_1, k_2 uniformly at random in $[0, \sqrt{n})$. This is secure!
  - Careless way: pick k_1, k_2 uniformly at random in $[0, 2^m)$ with $m = \lfloor \frac{1}{2} \log_2 n \rfloor$. Can be broken with Bleichebacher's attack!

Recomposition with GLS

- Recomposition certainly presents more interesting theoretical challenges
- We look at the specific case of curves generated with the quadratic GLS method:
  - E_0 over a prime field $\mathbb{F}_p$; order: $p + 1 - t, |t| \leqslant 2\sqrt{p}$
  - E quadratic twist of E_0 over $\mathbb{F}_{p^2}$
  - we assume further that $E(\mathbb{F}_{p^2})$ is of prime order n; then: $n = (p-1)^2 + t^2$ and $\lambda = \sqrt{-1} = t^{-1}(p-1) \bmod n$
- In this setting, two possible ways of carrying out recomposition:
  - Careful way: pick k_1, k_2 uniformly at random in $[0, \sqrt{n})$. This is secure!
  - Careless way: pick k_1, k_2 uniformly at random in $[0, 2^m)$ with $m = \lfloor \frac{1}{2} \log_2 n \rceil$. Can be broken with Bleichebacher's attack!

Recomposition with GLS

- Recomposition certainly presents more interesting theoretical challenges
- We look at the specific case of curves generated with the quadratic GLS method:
  - E_0 over a prime field $\mathbb{F}_p$; order: $p+1-t, |t| \leqslant 2\sqrt{p}$
  - E quadratic twist of E_0 over $\mathbb{F}_{p^2}$
  - we assume further that $E(\mathbb{F}_{p^2})$ is of prime order n; then: $n = (p-1)^2 + t^2$ and $\lambda = \sqrt{-1} = t^{-1}(p-1) \bmod n$
- In this setting, two possible ways of carrying out recomposition:
  - Careful way: pick k_1, k_2 uniformly at random in $[0, \sqrt{n})$. This is secure!
  - Careless way: pick k_1, k_2 uniformly at random in $[0, 2^m)$ with $m = \lfloor \frac{1}{2} \log_2 n \rceil$. Can be broken with Bleichebacher's attack!

Recomposition with GLS

- Recomposition certainly presents more interesting theoretical challenges
- We look at the specific case of curves generated with the quadratic GLS method:
  - E_0 over a prime field $\mathbb{F}_p$; order: $p + 1 - t, |t| \leqslant 2\sqrt{p}$
  - E quadratic twist of E_0 over $\mathbb{F}_{p^2}$
  - we assume further that $E(\mathbb{F}_{p^2})$ is of prime order n; then: $n = (p-1)^2 + t^2$ and $\lambda = \sqrt{-1} = t^{-1}(p-1) \bmod n$
- In this setting, two possible ways of carrying out recomposition:
  - Careful way: pick k_1, k_2 uniformly at random in $[0, \sqrt{n})$. This is secure!
  - Careless way: pick k_1, k_2 uniformly at random in $[0, 2^m)$ with $m = \lfloor \frac{1}{2} \log_2 n \rfloor$. Can be broken with Bleichebacher's attack!

Security of the "careful way"

- When k_1, k_2 are chosen uniformly at random in $[0, \sqrt{n})$, $k = k_1 + k_2\lambda$ is statistically close to uniform in $\mathbb{Z}/n\mathbb{Z}$, hence security!
- Proof idea: show that $(k_1, k_2) \mapsto k_1 + k_2\lambda$ induces an injective map $[0, p-1)^2 \to \mathbb{Z}/n\mathbb{Z}$
  - if $(x, y) \neq (x', y')$ have the same image, the fact that $\lambda^2 = -1 \bmod n$ yields $(x - x')^2 + (y - y')^2 = n$
  - but n, as a prime, has only one representation as a sum of two squares: $n = (p-1)^2 + t^2$
  - therefore, $|x - x'|$ or $|y - y'|$ must be $p - 1$, which is impossible.
- Good for quadratic GLS. Unfortunately, the proof doesn't immediately generalize to other curves with endomorphisms (e.g. GLV with $D = -3$?)

Security of the "careful way"

- When k_1, k_2 are chosen uniformly at random in $[0, \sqrt{n})$, $k = k_1 + k_2\lambda$ is statistically close to uniform in $\mathbb{Z}/n\mathbb{Z}$, hence security!
- Proof idea: show that $(k_1, k_2) \mapsto k_1 + k_2\lambda$ induces an injective map $[0, p-1)^2 \to \mathbb{Z}/n\mathbb{Z}$
  - if $(x, y) \neq (x', y')$ have the same image, the fact that $\lambda^2 = -1 \bmod n$ yields $(x - x')^2 + (y - y')^2 = n$
  - but n, as a prime, has only one representation as a sum of two squares: $n = (p-1)^2 + t^2$
  - therefore, $|x - x'|$ or $|y - y'|$ must be $p - 1$, which is impossible.
- Good for quadratic GLS. Unfortunately, the proof doesn't immediately generalize to other curves with endomorphisms (e.g. GLV with $D = -3$?)

Security of the “careful way”

- When k_1, k_2 are chosen uniformly at random in $[0, \sqrt{n})$, $k = k_1 + k_2\lambda$ is statistically close to uniform in $\mathbb{Z}/n\mathbb{Z}$, hence security!
- Proof idea: show that $(k_1, k_2) \mapsto k_1 + k_2\lambda$ induces an injective map $[0, p-1)^2 \to \mathbb{Z}/n\mathbb{Z}$
  - if $(x, y) \neq (x', y')$ have the same image, the fact that $\lambda^2 = -1 \bmod n$ yields $(x - x')^2 + (y - y')^2 = n$
  - but n, as a prime, has only one representation as a sum of two squares: $n = (p-1)^2 + t^2$
  - therefore, $|x - x'|$ or $|y - y'|$ must be $p - 1$, which is impossible.
- Good for quadratic GLS. Unfortunately, the proof doesn’t immediately generalize to other curves with endomorphisms (e.g. GLV with $D = -3$?)

Insecurity of the "careless way"

- Suppose now that k_1, k_2 are chosen uniformly in $[0, T)$, with $T = 2^{\lfloor \frac{1}{2} \log_2 n \rfloor}$. Bleichenbacher does not apply directly, because the bias on $k = k_1 + k_2\lambda$ is small:

$$B_n(K) = B_n(K_1) \cdot B_n(\lambda K_2) = \underbrace{\frac{1}{T}\left|\frac{\sin(\pi T/n)}{\sin(\pi/n)}\right|}_{\approx 1} \cdot \underbrace{\frac{1}{T}\left|\frac{\sin(\pi\lambda T/n)}{\sin(\pi\lambda/n)}\right|}_{\text{negligible}}$$

- But the bias on $t \cdot k \bmod n$ is significant:

$$B_n(tK) = B_n(tK_1) \cdot B_n\big((p-1)K_2\big) \approx \left|\frac{\sin(\pi(p-1)T/n)}{\pi(p-1)T/n}\right|$$

- $0.5 < (p-1)T/n < 1$
- if $(p-1)T/n \approx 0.5$ (i.e $n \approx$ power of 2): maximal bias $\Rightarrow B_n(tK) = 2/\pi \approx 0.637$

Insecurity of the "careless way"

- Suppose now that k_1, k_2 are chosen uniformly in $[0, T)$, with $T = 2^{\lfloor \frac{1}{2} \log_2 n \rceil}$. Bleichenbacher does not apply directly, because the bias on $k = k_1 + k_2 \lambda$ is small:

$$B_n(K) = B_n(K_1) \cdot B_n(\lambda K_2) = \underbrace{\frac{1}{T}\left|\frac{\sin(\pi T/n)}{\sin(\pi/n)}\right|}_{\approx 1} \cdot \underbrace{\frac{1}{T}\left|\frac{\sin(\pi \lambda T/n)}{\sin(\pi \lambda/n)}\right|}_{\text{negligible}}$$

- But the bias on $t \cdot k \bmod n$ is significant:

$$B_n(tK) = B_n(tK_1) \cdot B_n\big((p-1)K_2\big) \approx \left|\frac{\sin(\pi(p-1)T/n)}{\pi(p-1)T/n}\right|$$

- $0.5 < (p-1)T/n < 1$
- if $(p-1)T/n \approx 0.5$ (i.e $n \approx$ power of 2): maximal bias $\Rightarrow B_n(tK) = 2/\pi \approx 0.637$

Insecurity of the "careless way"

- Suppose now that k_1, k_2 are chosen uniformly in $[0, T)$, with $T = 2^{\lfloor \frac{1}{2} \log_2 n \rfloor}$. Bleichenbacher does not apply directly, because the bias on $k = k_1 + k_2\lambda$ is small:

$$B_n(K) = B_n(K_1) \cdot B_n(\lambda K_2) = \underbrace{\frac{1}{T}\left|\frac{\sin(\pi T/n)}{\sin(\pi/n)}\right|}_{\approx 1} \cdot \underbrace{\frac{1}{T}\left|\frac{\sin(\pi\lambda T/n)}{\sin(\pi\lambda/n)}\right|}_{\text{negligible}}$$

- But the bias on $t \cdot k \bmod n$ is significant:

$$B_n(tK) = B_n(tK_1) \cdot B_n\big((p-1)K_2\big) \approx \left|\frac{\sin(\pi(p-1)T/n)}{\pi(p-1)T/n}\right|$$

- $0.5 < (p-1)T/n < 1$
- if $(p-1)T/n \approx 0.5$ (i.e $n \approx$ power of 2): maximal bias $\Rightarrow B_n(tK) = 2/\pi \approx 0.637$

Insecurity of the "careless way"

- Suppose now that k_1, k_2 are chosen uniformly in $[0, T)$, with $T = 2^{\lfloor \frac{1}{2} \log_2 n \rfloor}$. Bleichenbacher does not apply directly, because the bias on $k = k_1 + k_2\lambda$ is small:

$$B_n(K) = B_n(K_1) \cdot B_n(\lambda K_2) = \underbrace{\frac{1}{T}\left|\frac{\sin(\pi T/n)}{\sin(\pi/n)}\right|}_{\approx 1} \cdot \underbrace{\frac{1}{T}\left|\frac{\sin(\pi\lambda T/n)}{\sin(\pi\lambda/n)}\right|}_{\text{negligible}}$$

- But the bias on $t \cdot k \bmod n$ is significant:

$$B_n(tK) = B_n(tK_1) \cdot B_n\big((p-1)K_2\big) \approx \left|\frac{\sin(\pi(p-1)T/n)}{\pi(p-1)T/n}\right|$$

- $0.5 < (p-1)T/n < 1$
- if $(p-1)T/n \approx 0.5$ (i.e $n \approx$ power of 2): maximal bias $\Rightarrow B_n(tK) = 2/\pi \approx 0.637$

Concrete attack on the "careless way"

- We successfully applied this variant of Bleichenbacher on a 160-bit GLS curve
- $E_0 : y^2 = x^3 - 3x/23 + 104$ minimal choice over the OPF field $\mathbb{F}_p$, $p = 255 \cdot 2^{72} + 1$
- $E : y^2 = x^3 - 3x + 104 \cdot \sqrt{23}^3$ over $\mathbb{F}_{p^2} = \mathbb{F}_p(\sqrt{23})$
- Timings and resource consumption similar to the SECG curve case, except for signature generation itself.

Concrete attack on the "careless way"

- We successfully applied this variant of Bleichenbacher on a 160-bit GLS curve
- $E_0 : y^2 = x^3 - 3x/23 + 104$ minimal choice over the OPF field $\mathbb{F}_p$, $p = 255 \cdot 2^{72} + 1$
- $E : y^2 = x^3 - 3x + 104 \cdot \sqrt{23}^3$ over $\mathbb{F}_{p^2} = \mathbb{F}_p(\sqrt{23})$
- Timings and resource consumption similar to the SECG curve case, except for signature generation itself.

Concrete attack on the "careless way"

- We successfully applied this variant of Bleichenbacher on a 160-bit GLS curve
- $E_0 : y^2 = x^3 - 3x/23 + 104$ minimal choice over the OPF field $\mathbb{F}_p$, $p = 255 \cdot 2^{72} + 1$
- $E : y^2 = x^3 - 3x + 104 \cdot \sqrt{23}^3$ over $\mathbb{F}_{p^2} = \mathbb{F}_p(\sqrt{23})$
- Timings and resource consumption similar to the SECG curve case, except for signature generation itself.

Concrete attack on the "careless way"

- We successfully applied this variant of Bleichenbacher on a 160-bit GLS curve
- $E_0 : y^2 = x^3 - 3x/23 + 104$ minimal choice over the OPF field $\mathbb{F}_p$, $p = 255 \cdot 2^{72} + 1$
- $E : y^2 = x^3 - 3x + 104 \cdot \sqrt{23}^3$ over $\mathbb{F}_{p^2} = \mathbb{F}_p(\sqrt{23})$
- Timings and resource consumption similar to the SECG curve case, except for signature generation itself.

Outline

Attack on ECDSA with 1-bit nonce bias

HNP attacks on ECDSA

Our attack on 1-bit bias

GLV/GLS decomposition and HNP

Curves with fast endomorphisms

The recomposition approach

The decomposition approach

Conclusion

Decompositon and side-channels

- Recall that the decomposition approach consists in choosing k randomly, and computing half-size k_1, k_2 coefficients such that $k = k_1 + k_2\lambda \bmod n$ afterwards
- If k is chosen uniformly at random in $\mathbb{Z}/n\mathbb{Z}$, no mathematical problem with the distribution
- But the physical implementation of the algorithm computing (k_1, k_2) from k may leak information!
- Concretely, we considered a specific algorithm due to Park et al. for decomposition, and showed that an unprotected implementation of it leaks the LSBs of k
- Template attack on an 8-bit AVR smartcard: possible to recover the least significant byte of k. Then breaking ECDSA is easy with lattices!

Decompositon and side-channels

- Recall that the decomposition approach consists in choosing k randomly, and computing half-size k_1, k_2 coefficients such that $k = k_1 + k_2\lambda \bmod n$ afterwards
- If k is chosen uniformly at random in $\mathbb{Z}/n\mathbb{Z}$, no mathematical problem with the distribution
- But the physical implementation of the algorithm computing (k_1, k_2) from k may leak information!
- Concretely, we considered a specific algorithm due to Park et al. for decomposition, and showed that an unprotected implementation of it leaks the LSBs of k
- Template attack on an 8-bit AVR smartcard: possible to recover the least significant byte of k. Then breaking ECDSA is easy with lattices!

Decompositon and side-channels

- Recall that the decomposition approach consists in choosing k randomly, and computing half-size k_1, k_2 coefficients such that $k = k_1 + k_2\lambda \bmod n$ afterwards
- If k is chosen uniformly at random in $\mathbb{Z}/n\mathbb{Z}$, no mathematical problem with the distribution
- But the physical implementation of the algorithm computing (k_1, k_2) from k may leak information!
- Concretely, we considered a specific algorithm due to Park et al. for decomposition, and showed that an unprotected implementation of it leaks the LSBs of k
- Template attack on an 8-bit AVR smartcard: possible to recover the least significant byte of k. Then breaking ECDSA is easy with lattices!

Decompositon and side-channels

- Recall that the decomposition approach consists in choosing k randomly, and computing half-size k_1, k_2 coefficients such that $k = k_1 + k_2\lambda \bmod n$ afterwards
- If k is chosen uniformly at random in $\mathbb{Z}/n\mathbb{Z}$, no mathematical problem with the distribution
- But the physical implementation of the algorithm computing (k_1, k_2) from k may leak information!
- Concretely, we considered a specific algorithm due to Park et al. for decomposition, and showed that an unprotected implementation of it leaks the LSBs of k
- Template attack on an 8-bit AVR smartcard: possible to recover the least significant byte of k. Then breaking ECDSA is easy with lattices!

Decompositon and side-channels

- Recall that the decomposition approach consists in choosing k randomly, and computing half-size k_1, k_2 coefficients such that $k = k_1 + k_2\lambda \bmod n$ afterwards
- If k is chosen uniformly at random in $\mathbb{Z}/n\mathbb{Z}$, no mathematical problem with the distribution
- But the physical implementation of the algorithm computing (k_1, k_2) from k may leak information!
- Concretely, we considered a specific algorithm due to Park et al. for decomposition, and showed that an unprotected implementation of it leaks the LSBs of k
- Template attack on an 8-bit AVR smartcard: possible to recover the least significant byte of k. Then breaking ECDSA is easy with lattices!

Outline

Conclusion

- Record of the smallest amount of nonce bias needed to break ECDSA
  - 1-bit bias on 160-bit elliptic curves!
  - impossible with lattices, but doable with Bleichenbacher
  - significant computational effort, but a resourceful attacker could go much further (256-bit curves?)
- One should be careful about GLV/GLS decomposition
  - GLV/GLS curves are not safe against nonce attacks
  - When properly carried out, the recomposition technique can be secure
  - When using decomposition, pay attention to side-channels

Conclusion

- Record of the smallest amount of nonce bias needed to break ECDSA
  - 1-bit bias on 160-bit elliptic curves!
  - impossible with lattices, but doable with Bleichenbacher
  - significant computational effort, but a resourceful attacker could go much further (256-bit curves?)
- One should be careful about GLV/GLS decomposition
  - GLV/GLS curves are not safe against nonce attacks
  - When properly carried out, the recomposition technique can be secure
  - When using decomposition, pay attention to side-channels

謝謝！

Thank you for your attention