Is $2^{255} - 19$ big enough?

Generate public keys on a “strong” elliptic curve $E$ over the field $\mathbb{Z}/(2^{255} - 19)$. Is that safe?

“Size does matter!”

What marketing says

56-bit crypto: Broken.
128-bit crypto: Okay.
256-bit crypto: High security!
512-bit crypto: Broken.
1024-bit crypto: Shaky.

$2^{255} - 19$ must be, um, 256 bits. Fantastic!
Best possible security level.
Is 2\(^{255} - 19\) big enough?

Generate public keys on a “strong” elliptic curve \(E\) over the field \(\mathbb{Z}_{2^{255} - 19}\).

Is that safe?

“Size does matter!”

What marketing says

56-bit crypto: Broken.
128-bit crypto: Okay.
256-bit crypto: High security!
512-bit crypto: Broken.
1024-bit crypto: Shaky.

\(2^{255} - 19\) must be, um, 256 bits.

Fantastic!

Best possible security level.

What NSA says

NSA approves products for “classified or mission critical national security information.”

NSA wants “elliptic curves over \(GF(p)\), where \(p\) is a prime number greater than \(2^{255}\).”

So \(2^{255} + 95\) is fine for national security information, but \(2^{255} - 19\) is not.
What marketing says

56-bit crypto: Broken.
128-bit crypto: Okay.
256-bit crypto: High security!
512-bit crypto: Broken.
1024-bit crypto: Shaky.

$2^{255} - 19$ must be, um, 256 bits.
Fantastic!
Best possible security level.

What NSA says

NSA approves products for “classified or mission critical national security information.”

NSA wants “elliptic curves over $\text{GF}(p)$, where $p$ is a prime number greater than $2^{255}$.”

So $2^{255} + 95$ is fine for national security information but $2^{255} - 19$ is not.
What marketing says

56-bit crypto: Broken.
128-bit crypto: Okay.
256-bit crypto: High security!
512-bit crypto: Broken.
1024-bit crypto: Shaky.

... 2^{255} + 95 must be, um, 256 bits.
Fantastic!
Best possible security level.

What NSA says

NSA approves products for “classified or mission critical national security information.”

NSA wants “elliptic curves over \( \text{GF}(p) \), where \( p \) is a prime number greater than \( 2^{255} \).”

So \( 2^{255} + 95 \) is fine for national security information but \( 2^{255} - 19 \) is not.

What NIST says

128-bit AES keys “correspond” to ECC primes with “256-383” bits: the amount of work needed to “break the algorithms” is approximately the same, namely \( 2^{128} \) operations, by best techniques known.
What NSA says

NSA approves products for “classified or mission critical national security information.”

NSA wants “elliptic curves over $\text{GF}(p)$, where $p$ is a prime number greater than $2^{255}$.”

So $2^{255} + 95$ is fine for national security information but $2^{255} - 19$ is not.

What NIST says

128-bit AES keys “correspond” to ECC primes with “256-383” bits: the amount of work needed to “break the algorithms” is approximately the same, namely $2^{128}$ operations, by best techniques known.
What NSA says

NSA approves products for “classified or mission critical national security information.”

NSA wants “elliptic curves over $\mathbb{GF}(2^n)$, where $n$ is a prime number greater than $2^{255}$.”

So $2^{255} + 95$ is fine for national security information but $2^{255} - 19$ is not.

What NIST says

128-bit AES keys “correspond” to ECC primes with “256-383” bits: the amount of work needed to “break the algorithms” is approximately the same, namely $2^{128}$ operations, by best techniques known.

What I say

Given $H(k) = \text{AES}(0)$, find using $\approx 2^{127}$ AES evaluations.

Given $H(k_1), H(k_2)$, find all $k_i$ using a total of $2^{127}$ AES evaluations.

Or find some $k_i$ using $2^{87}$ AES evaluations.

Standard algorithms have negligible communication and perfect parallelization: see, e.g., cr.yp.to/papers.html #bruteforce
What NIST says

128-bit AES keys “correspond” to ECC primes with “256-383” bits: the amount of work needed to “break the algorithms” is approximately the same, namely $2^{128}$ operations, by best techniques known.

What I say

Given $H(k) = AES_k(0)$, find $k$ using $\approx 2^{127}$ AES evaluations.

Given $H(k_1), H(k_2), \ldots, H(k_{240})$, find all $k_i$ using a total of $\approx 2^{127}$ AES evaluations.

Or find some $k_i$ using $\approx 2^{87}$ AES evaluations.

Standard algorithms have negligible communication and perfect parallelization: see, e.g., cr.yp.to/papers.html #bruteforce
What I say

Given $H(k) = AES_k(0)$, find $k$ using $\approx 2^{127}$ AES evaluations.

Given $H(k_1), H(k_2), \ldots, H(k_{2^{40}})$, find all $k_i$ using a total of $\approx 2^{127}$ AES evaluations.

Or find some $k_i$ using $\approx 2^{87}$ AES evaluations.

Standard algorithms have negligible communication and perfect parallelization: see, e.g., cr.yp.to/papers.html #bruteforce

Given public key on 255-bit elliptic curve, find secret key using $\approx 2^{127}$ additions.

Given $2^{40}$ public keys, find all secret keys using $\approx 2^{147}$ additions.

Finding some key is as hard as finding first key: $\approx 2^{127}$ additions. Easily prove by random self-reduction.

See, e.g., Kuhn and Struik, 2001.
What I say

Given $H(k) = \text{AES}_k(0)$, find $k$ using $\approx 2^{127}$ AES evaluations.

Given $H(k_1), H(k_2), \ldots, H(k_{2^{40}})$, find all $k_i$ using a total of $\approx 2^{127}$ AES evaluations.

Or find some $k_i$ using $\approx 2^{87}$ AES evaluations.

Standard algorithms have negligible communication and perfect parallelization: see, e.g., cr.yp.to/papers.html #bruteforce

Given public key on 255-bit elliptic curve $E$, find secret key using $\approx 2^{127}$ additions on $E$.

Given $2^{40}$ public keys, find all secret keys using $\approx 2^{147}$ additions on $E$.

Finding some key is as hard as finding first key: $\approx 2^{127}$ additions. Easily prove by random self-reduction.

See, e.g., Kuhn and Struik, 2001.
Given \( \text{AES}_k(0) \), find \( k \) using \( 2^{127} \) AES evaluations.

Given \( \text{AES}(k_2) \), \( \ldots \), \( \text{AES}(k_{240}) \), finding all \( k \)'s \( \approx 2^{127} \) evaluations.

Even worse for AES: Attacker can try much less computation. Success chance drops linearly.

For elliptic curves, success chance drops quadratically.

Bottom line: 128-bit AES keys are not comparable in security to 255-bit elliptic-curve keys.

Is \( 2^{255} - 19 \) big enough? Yes.

Is 128-bit AES safe? Unclear.
Given public key on 255-bit elliptic curve $E$, find secret key using $\approx 2^{127}$ additions on $E$.

Given $2^{40}$ public keys, find all secret keys using $\approx 2^{147}$ additions on $E$.

Finding some key is as hard as finding first key: $\approx 2^{127}$ additions. Easily prove by random self-reduction.

See, e.g., Kuhn and Struik, 2001.

Even worse for AES: Attacker can try much less computation. Success chance drops linearly.

For elliptic curves, success chance drops quadratically.

Bottom line: 128-bit AES keys are not comparable in security to 255-bit elliptic-curve keys.

Is $2^{255} - 19$ big enough? Yes.
Is 128-bit AES safe? Unclear.