I Want to Break Square-free: The $4p - 1$ Factorization Method and Its RSA Backdoor Viability

SECRYPT 2019

Vladimír Sedláček
Dušan Klinec
Marek Sýs
Petr Švenda
Vashek Matyáš

vlada.sedlacek@mail.muni.cz

Centre for Research on Cryptography and Security

Masaryk University
Brno, Czech Republic

Overview

Motivation

- Factorization of integers - an old and well studied problem
- RSA depends on its infeasibility

Motivation

- Factorization of integers - an old and well studied problem
- RSA depends on its infeasibility

Modern factorization algorithms:

- Pollard's $\rho, p-1$
- Lenstra's elliptic curve method
- Quadratic sieve
- Number field sieve
- Shor's algorithm (quantum) - first general polynomial

Motivation

- Factorization of integers - an old and well studied problem
- RSA depends on its infeasibility

Modern factorization algorithms:

- Pollard's $\rho, p-1$
- Lenstra's elliptic curve method
- Quadratic sieve
- Number field sieve
- Shor's algorithm (quantum) - first general polynomial
- The $4p-1$ method - Qi Cheng (2002)

Motivation

- Factorization of integers - an old and well studied problem
- RSA depends on its infeasibility

Modern factorization algorithms:

- Pollard's $\rho, p-1$
- Lenstra's elliptic curve method
- Quadratic sieve
- Number field sieve
- Shor's algorithm (quantum) - first general polynomial
- The $4p-1$ method - Qi Cheng (2002)
  - very fast, but special assumptions

Motivation

- Factorization of integers - an old and well studied problem
- RSA depends on its infeasibility

Modern factorization algorithms:

- Pollard's $\rho, p-1$
- Lenstra's elliptic curve method
- Quadratic sieve
- Number field sieve
- Shor's algorithm (quantum) - first general polynomial
- The $4p-1$ method - Qi Cheng (2002)
  - very fast, but special assumptions
  - interesting as a backdoor

Cheng's $4p - 1$ method - basic ideas

- $N = pq$, where $p, q > 3$ are distinct primes

Cheng's $4p - 1$ method - basic ideas

- $N = pq$, where $p, q > 3$ are distinct primes
- an elliptic curve E over $\mathbb{Z}_N$: given by

$$y^2 \equiv x^3 + ax + b \pmod{N},$$

  forms an abelian group

Cheng's $4p - 1$ method - basic ideas

- $N = pq$, where $p, q > 3$ are distinct primes
- an elliptic curve E over $\mathbb{Z}_N$: given by

$$y^2 \equiv x^3 + ax + b \pmod{N},$$

  forms an abelian group
- we have $E(\mathbb{Z}_N) \cong E(\mathbb{F}_p) \oplus E(\mathbb{F}_q)$

Cheng's $4p - 1$ method - basic ideas

- $N = pq$, where $p, q > 3$ are distinct primes
- an elliptic curve E over $\mathbb{Z}_N$: given by

$$y^2 \equiv x^3 + ax + b \pmod{N},$$

  forms an abelian group
- we have $E(\mathbb{Z}_N) \cong E(\mathbb{F}_p) \oplus E(\mathbb{F}_q)$
- if $|E(\mathbb{F}_p)| = p$, multiplication by N annihilates the first summand, which reveals p

The basic idea

For any $k \in \mathbb{Z}$, there are division polynomials $\psi_k, \phi_k, \omega_k \in \mathbb{Z}[x]$:

$$k \cdot P = \left(\frac{\phi_k(x)}{\psi_k^2(x)}, \frac{\omega_k(x,y)}{\psi_k^3(x,y)} \right)$$

for all $P = (x, y)$ on E.

The basic idea

For any $k \in \mathbb{Z}$, there are division polynomials $\psi_k, \phi_k, \omega_k \in \mathbb{Z}[x]$:

$$k \cdot P = \left(\frac{\phi_k(x)}{\psi_k^2(x)}, \frac{\omega_k(x,y)}{\psi_k^3(x,y)} \right)$$

for all $P = (x, y)$ on E.

- if $|E(\mathbb{F}_p)| = p$ and $P \in E(\mathbb{Z}_N)$, then the computation of $N \cdot P$ usually fails

The basic idea

For any $k \in \mathbb{Z}$, there are division polynomials $\psi_k, \phi_k, \omega_k \in \mathbb{Z}[x]$:

$$k \cdot P = \left(\frac{\phi_k(x)}{\psi_k^2(x)}, \frac{\omega_k(x,y)}{\psi_k^3(x,y)} \right)$$

for all $P = (x, y)$ on E.

- if $|E(\mathbb{F}_p)| = p$ and $P \in E(\mathbb{Z}_N)$, then the computation of $N \cdot P$ usually fails
- this is because $\psi_N(x)$ is invertible modulo q, but not modulo p

The basic idea

For any $k \in \mathbb{Z}$, there are division polynomials $\psi_k, \phi_k, \omega_k \in \mathbb{Z}[x]$:

$$k \cdot P = \left(\frac{\phi_k(x)}{\psi_k^2(x)}, \frac{\omega_k(x,y)}{\psi_k^3(x,y)} \right)$$

for all $P = (x, y)$ on E.

- if $|E(\mathbb{F}_p)| = p$ and $P \in E(\mathbb{Z}_N)$, then the computation of $N \cdot P$ usually fails
- this is because $\psi_N(x)$ is invertible modulo q, but not modulo p
- thus we can recover $p = \gcd(N, \psi_N(x))$

Complex multiplication (CM)

How to find E such that $|E(\mathbb{F}_p)| = p$?

Complex multiplication (CM)

How to find E such that $|E(\mathbb{F}_p)| = p$?

- ECs over $\mathbb{F}_p$ are classified by their j-invariant (up to twists)

Complex multiplication (CM)

How to find E such that $|E(\mathbb{F}_p)| = p$?

- ECs over $\mathbb{F}_p$ are classified by their j-invariant (up to twists)
- if the j-invariant of E is a root of the $-D$-th Hilbert class polynomial $H_{-D}(x)$ mod p, then $|E(\mathbb{F}_p)| = p + 1 \pm t$, where $4p = t^2 + Ds^2$

Complex multiplication (CM)

How to find E such that $|E(\mathbb{F}_p)| = p$?

- ECs over $\mathbb{F}_p$ are classified by their j-invariant (up to twists)
- if the j-invariant of E is a root of the $-D$-th Hilbert class polynomial $H_{-D}(x)$ mod p, then $|E(\mathbb{F}_p)| = p + 1 \pm t$, where $4p = t^2 + Ds^2$
- thus if $4p - 1 = Ds^2$ and $H_{-D}(j(E)) \equiv 0 \pmod{p}$, then $|E(\mathbb{F}_p)| = p$ in one half of cases

An outline of the algorithm

Suppose that we know that $4p - 1 = Ds^2$ for some $s, D \in \mathbb{Z}$, where D is known.

An outline of the algorithm

Suppose that we know that $4p - 1 = Ds^2$ for some $s, D \in \mathbb{Z}$, where D is known. Ideally, we would like to do the following:

1. construct H_{-D},

An outline of the algorithm

Suppose that we know that $4p - 1 = Ds^2$ for some $s, D \in \mathbb{Z}$, where D is known. Ideally, we would like to do the following:

1. construct H_{-D},
2. find a root j_0 of H_{-D} modulo p,

An outline of the algorithm

Suppose that we know that $4p - 1 = Ds^2$ for some $s, D \in \mathbb{Z}$, where D is known. Ideally, we would like to do the following:

1. construct H_{-D},
2. find a root j_0 of H_{-D} modulo p,
3. construct E_{j_0},

An outline of the algorithm

Suppose that we know that $4p - 1 = Ds^2$ for some $s, D \in \mathbb{Z}$, where D is known. Ideally, we would like to do the following:

1. construct H_{-D},
2. find a root j_0 of H_{-D} modulo p,
3. construct E_{j_0},
4. find a point $P = (x, y)$ on E_{j_0},

An outline of the algorithm

Suppose that we know that $4p - 1 = Ds^2$ for some $s, D \in \mathbb{Z}$, where D is known. Ideally, we would like to do the following:

1. construct H_{-D},
2. find a root j_0 of H_{-D} modulo p,
3. construct E_{j_0},
4. find a point $P = (x, y)$ on E_{j_0},
5. try to compute $N \cdot P$ (or just $\psi_N(x)$),

An outline of the algorithm

Suppose that we know that $4p - 1 = Ds^2$ for some $s, D \in \mathbb{Z}$, where D is known. Ideally, we would like to do the following:

1. construct H_{-D},
2. find a root j_0 of H_{-D} modulo p,
3. construct E_{j_0},
4. find a point $P = (x, y)$ on E_{j_0},
5. try to compute $N \cdot P$ (or just $\psi_N(x)$),
6. if the computation of $N \cdot P$ does not fail, go back to step 3) and replace E_{j_0} with its twist,

An outline of the algorithm

Suppose that we know that $4p - 1 = Ds^2$ for some $s, D \in \mathbb{Z}$, where D is known. Ideally, we would like to do the following:

1. construct H_{-D},
2. find a root j_0 of H_{-D} modulo p,
3. construct E_{j_0},
4. find a point $P = (x, y)$ on E_{j_0},
5. try to compute $N \cdot P$ (or just $\psi_N(x)$),
6. if the computation of $N \cdot P$ does not fail, go back to step 3) and replace E_{j_0} with its twist,
7. if the computation of $N \cdot P$ fails, compute a factor of N as $\gcd(\psi_N(x), N)$.

Problems

Not knowing p:

- obstacle for solving nonlinear congruences modulo p

Problems

Not knowing p:

- obstacle for solving nonlinear congruences modulo p
- solution: replace $\mathbb{Z}_N$ by $\mathbb{Z}_N[x]/H_{-D}(x)$ and compute symbolically

Problems

Not knowing p:

- obstacle for solving nonlinear congruences modulo p
- solution: replace $\mathbb{Z}_N$ by $\mathbb{Z}_N[x]/H_{-D}(x)$ and compute symbolically
- not clear how to find the point P and the correct twist

Problems

Not knowing p:

- obstacle for solving nonlinear congruences modulo p
- solution: replace $\mathbb{Z}_N$ by $\mathbb{Z}_N[x]/H_{-D}(x)$ and compute symbolically
- not clear how to find the point P and the correct twist
- solution: probabilistic guessing

Problems

Not knowing p:

- obstacle for solving nonlinear congruences modulo p
- solution: replace $\mathbb{Z}_N$ by $\mathbb{Z}_N[x]/H_{-D}(x)$ and compute symbolically
- not clear how to find the point P and the correct twist
- solution: probabilistic guessing

Computation of H_{-D}:

- complicated, roughly exponential in D

Problems

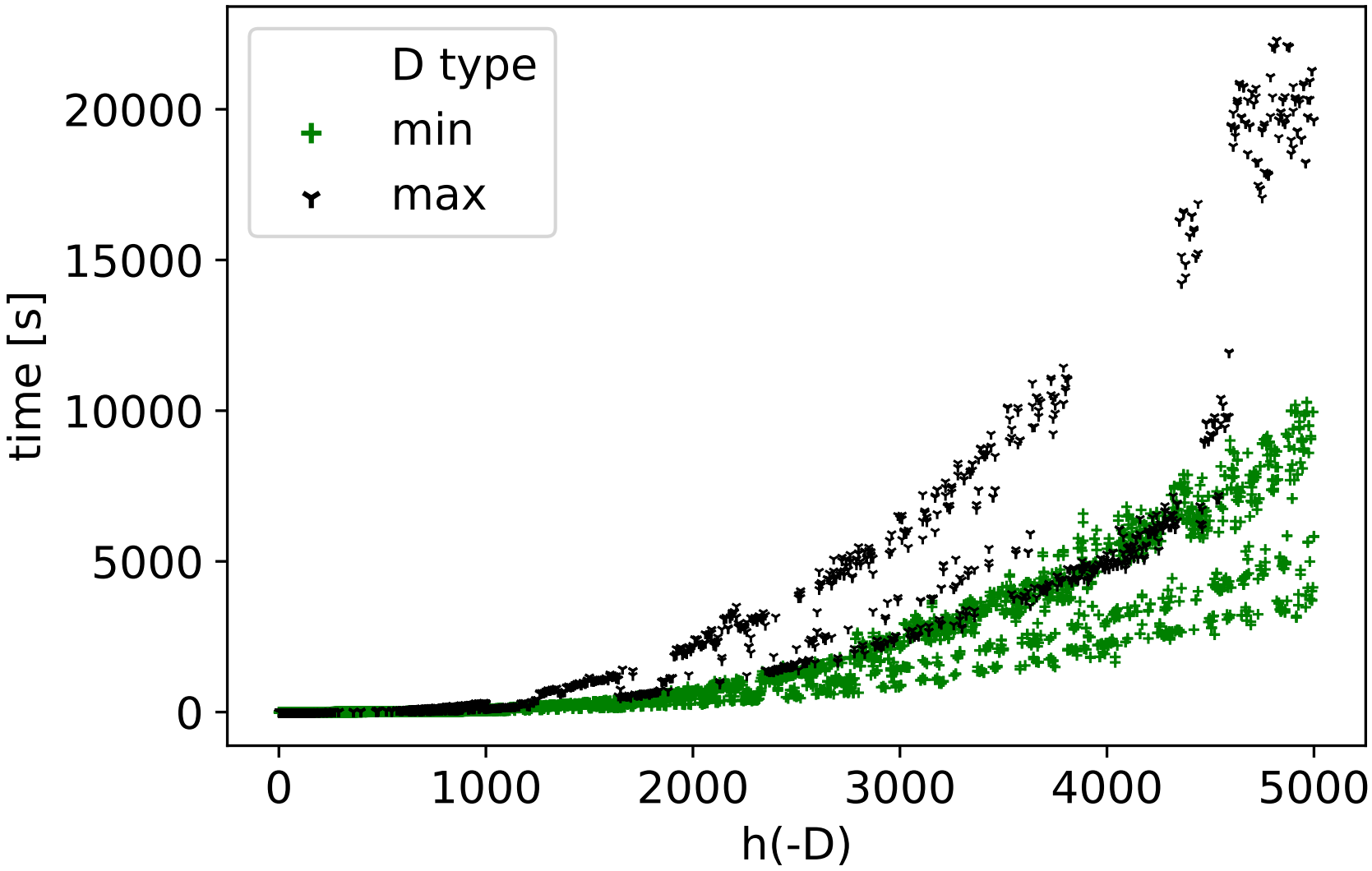
Not knowing p:

- obstacle for solving nonlinear congruences modulo p
- solution: replace $\mathbb{Z}_N$ by $\mathbb{Z}_N[x]/H_{-D}(x)$ and compute symbolically
- not clear how to find the point P and the correct twist
- solution: probabilistic guessing

Computation of H_{-D}:

- complicated, roughly exponential in D
- current record: $D \approx 2^{53}$

Problems


D type
min
max
20000
15000
10000
5000
0
time [s]
0
1000
2000
3000
4000
5000
h(-D)

Time complexity

The improved (probabilistic) algorithm is polynomial in N, but exponential in D (the squarefree part of $4p - 1$).

Time complexity

The improved (probabilistic) algorithm is polynomial in N, but exponential in D (the squarefree part of $4p - 1$).

Current record for $D = 11$ (with our custom implementation on a single core):

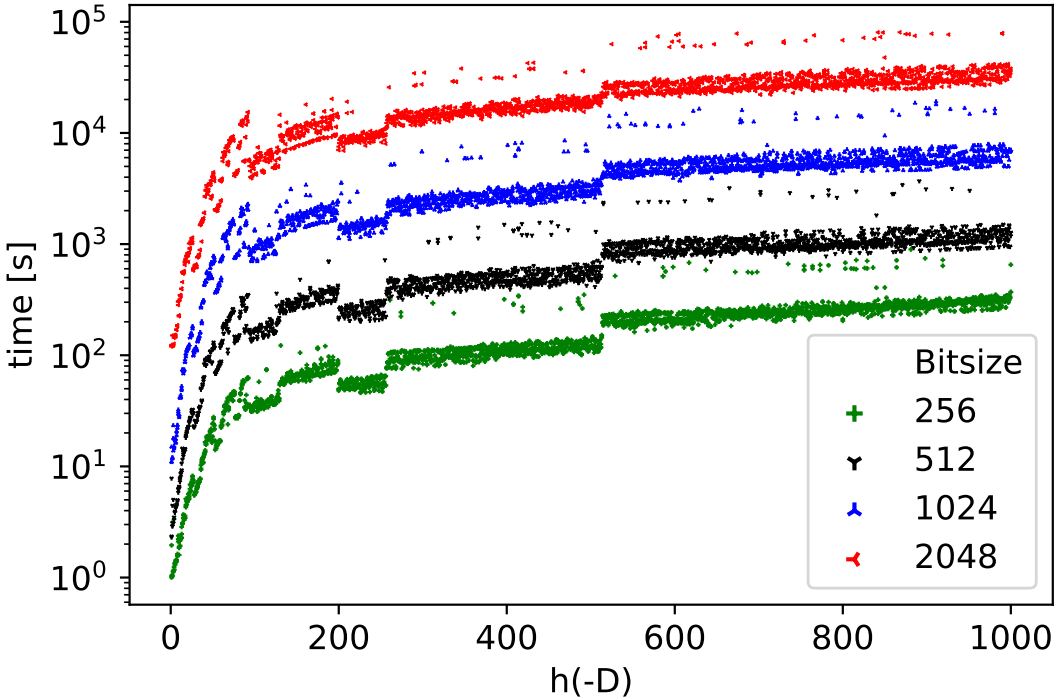
- RSA 4096-bit moduli factored in around 150 seconds
- RSA 2048-bit moduli factored in under 10 seconds

Time complexity

The improved (probabilistic) algorithm is polynomial in N, but exponential in D (the squarefree part of $4p - 1$).

Current record for $D = 11$ (with our custom implementation on a single core):

- RSA 4096-bit moduli factored in around 150 seconds
- RSA 2048-bit moduli factored in under 10 seconds



Applicability

- for random primes p, the condition $4p - 1 = Ds^2$ with small D is extermely rare

Applicability

- for random primes p, the condition $4p - 1 = Ds^2$ with small D is extermely rare
- only $\frac{1}{\sqrt{X}}$ of primes $p < X$ satisfy it

Applicability

- for random primes p, the condition $4p - 1 = Ds^2$ with small D is extermely rare
- only $\frac{1}{\sqrt{X}}$ of primes $p < X$ satisfy it
- still could serve as an interesting backdoor (e.g., on black-box devices)

Applicability

- for random primes p, the condition $4p - 1 = Ds^2$ with small D is extermely rare
- only $\frac{1}{\sqrt{X}}$ of primes $p < X$ satisfy it
- still could serve as an interesting backdoor (e.g., on black-box devices)
- generation of vulnerable primes for given D is easy

Backdoor properties

Advantages:

- works for all key lengths

Backdoor properties

Advantages:

- works for all key lengths
- no observable bias for backdoored keys

Backdoor properties

Advantages:

- works for all key lengths
- no observable bias for backdoored keys
- adjustable factorization difficulty, favorable ratio

Backdoor properties

Advantages:

- works for all key lengths
- no observable bias for backdoored keys
- adjustable factorization difficulty, favorable ratio
- possible parallelizability

Backdoor properties

Advantages:

- works for all key lengths
- no observable bias for backdoored keys
- adjustable factorization difficulty, favorable ratio
- possible parallelizability

Disdvantages:

- easy to detect from private keys if the same D is reused or for short keys (< 1280 bits)

Backdoor properties

Advantages:

- works for all key lengths
- no observable bias for backdoored keys
- adjustable factorization difficulty, favorable ratio
- possible parallelizability

Disdvantages:

- easy to detect from private keys if the same D is reused or for short keys (< 1280 bits)
- an unpredictable unique D for each keypair can be problematic

Backdoor properties

Advantages:

- works for all key lengths
- no observable bias for backdoored keys
- adjustable factorization difficulty, favorable ratio
- possible parallelizability

Disdvantages:

- easy to detect from private keys if the same D is reused or for short keys (< 1280 bits)
- an unpredictable unique D for each keypair can be problematic
- if D is leaked, anyone can perform the factorization

Backdoor detection

Inquirer scenarios:

1. public keys only - need to guess

Backdoor detection

Inquirer scenarios:

1. public keys only - need to guess
2. short private keys (< 768 bits) - a direct factorization reveals the backdoor

Backdoor detection

Inquirer scenarios:

1. public keys only - need to guess
2. short private keys (< 768 bits) - a direct factorization reveals the backdoor
3. many private keys - batch GCD reveals the backdoor if D is not unique per keypair

Audit of keys

- 44.7 million RSA keypairs generated by 15 smartcards and 3 HSMs

Audit of keys

- 44.7 million RSA keypairs generated by 15 smartcards and 3 HSMs
- access to private keys, keylengths 512,1024,2048 bits

Audit of keys

- 44.7 million RSA keypairs generated by 15 smartcards and 3 HSMs
- access to private keys, keylengths 512,1024,2048 bits
- Scenario 2:
  - random selection of 5000 512-bit keys and 100 1024-bit keys

Audit of keys

- 44.7 million RSA keypairs generated by 15 smartcards and 3 HSMs
- access to private keys, keylengths 512,1024,2048 bits
- Scenario 2:
  - random selection of 5000 512-bit keys and 100 1024-bit keys
  - square-free parts of $4p - 1$ and $4q - 1$ computed, all large enough

Audit of keys

- 44.7 million RSA keypairs generated by 15 smartcards and 3 HSMs
- access to private keys, keylengths 512,1024,2048 bits
- Scenario 2:
  - random selection of 5000 512-bit keys and 100 1024-bit keys
  - square-free parts of $4p - 1$ and $4q - 1$ computed, all large enough
- Scenario 3:
  - all 44.7M keys (including 2048-bit) used

Audit of keys

- 44.7 million RSA keypairs generated by 15 smartcards and 3 HSMs
- access to private keys, keylengths 512,1024,2048 bits
- Scenario 2:
  - random selection of 5000 512-bit keys and 100 1024-bit keys
  - square-free parts of $4p - 1$ and $4q - 1$ computed, all large enough
- Scenario 3:
  - all 44.7M keys (including 2048-bit) used
  - batch GCD used for all $4p - 1$ and $4q - 1$, as well as the product of "small" D's

Audit of keys

- 44.7 million RSA keypairs generated by 15 smartcards and 3 HSMs
- access to private keys, keylengths 512,1024,2048 bits
- Scenario 2:
  - random selection of 5000 512-bit keys and 100 1024-bit keys
  - square-free parts of $4p - 1$ and $4q - 1$ computed, all large enough
- Scenario 3:
  - all 44.7M keys (including 2048-bit) used
  - batch GCD used for all $4p - 1$ and $4q - 1$, as well as the product of "small" D's
  - no small square-free parts found

Conclusion

Main contributions:

- method simplified and better analyzed, faster than claimed and asymptotically determinisitic

Conclusion

Main contributions:

- method simplified and better analyzed, faster than claimed and asymptotically determinisitic
- public implementation, many experimental evaluations

Conclusion

Main contributions:

- method simplified and better analyzed, faster than claimed and asymptotically determinisitic
- public implementation, many experimental evaluations
- discussion of backdoor viability and possible scenarios

Conclusion

Main contributions:

- method simplified and better analyzed, faster than claimed and asymptotically determinisitic
- public implementation, many experimental evaluations
- discussion of backdoor viability and possible scenarios
- 44.7M keys analyzed, no backdoors found

Conclusion

Main contributions:

- method simplified and better analyzed, faster than claimed and asymptotically determinisitic
- public implementation, many experimental evaluations
- discussion of backdoor viability and possible scenarios
- 44.7M keys analyzed, no backdoors found
- main result: an attacker would need unique D's, but the backdoor presence cannot be ruled out for longer keys (such as 2048 bits)

Conclusion

Main contributions:

- method simplified and better analyzed, faster than claimed and asymptotically determinisitic
- public implementation, many experimental evaluations
- discussion of backdoor viability and possible scenarios
- 44.7M keys analyzed, no backdoors found
- main result: an attacker would need unique D's, but the backdoor presence cannot be ruled out for longer keys (such as 2048 bits)

Thank you for your attention.

All data and implementation are publicly available at `https://crocs.fi.muni.cz/public/papers/Secrypt2019`.