Public Key Cryptography

Henning Seidler

April 25, 2024

Organisation

- one lecture a week
- irregular example sheets, including programming tasks
  - Install Python, including IPython
  - at least one task needs SageMath
  - you are advised to create your own tools collection
- notes/slides, example sheets on ISIS

Organisation

- one lecture a week
- irregular example sheets, including programming tasks
  - Install Python, including IPython
  - at least one task needs SageMath
  - you are advised to create your own tools collection
- notes/slides, example sheets on ISIS

Exam

- part of "Secure Cryptography"
  cannot be examined alone
- oral exam at the end, about PKC and CryptoSec
- details will be announced towards the end of the semester
- prior to the exam no registration necessary

Organisation – Further Information

Questions?

- check course description(!)
- read announcements
- ask in the forum
- only if question contains private information, mail henning.seidler@tu-berlin.de

Organisation – Further Information

Questions?

- check course description(!)
- read announcements
- ask in the forum
- only if question contains private information, mail henning.seidler@tu-berlin.de

Literature

- Galbraith – "Mathematics of Public Key Cryptography"
- "Handbook of Applied Cryptography" (?)
- Wikipedia/Write-Ups/papers/...

Tell me, if you find a matching text book.

Goals

There are two kinds of cryptography in this world: cryptography that will stop your kid sister from reading your files, and cryptography that will stop major governments from reading your files. This [lecture] is about the latter.

(Bruce Schneier)

Goals

> *There are two kinds of cryptography in this world: cryptography that will stop your kid sister from reading your files, and cryptography that will stop major governments from reading your files. This [lecture] is about the latter.*
>
> *(Bruce Schneier)*

- give you an overview about Public Key Crypto
- typical encryption schemes
- also tell you, what can go wrong
- including practical tasks
- prepare you for CTFs

$> whoami && jobs

AG Rechnersicherheit

- We're a registered student organization. Basically a group of students interested in (IT) security topics.
- Weekly Meetups:
  - Tue, 6pm - 8pm
  - TEL 20, Auditorium 3 and via Jitsi
  - Techtalks and discussions about recent events or techniques
  - No knowledge needed - just be interested and eager to learn new things! :-)
- We participate in hacking contests (CTFs) as ENOFLAG/LEGOFAN/Last-Email!

$> cat ~/.todo

Upcoming events

this Saturday: Bambi-CTF (beginner)

- Attack-Defense CTF
- Exploit other teams while fixing our own vulnerabilities

several weekends:

play Jeopardy CTFs

- tasks with security flaws
- find secret code (Flag)

?? 2024: FaustCTF

8.6.2024: CryptoCTF

22.6.2024 ~~LNDW~~

$> exit

See/Hear you on Tuesday :-)

- Auditorium 3 @ TEL 20. floor on Tuesday 6 - 8 pm
- `https://meet.enoflag.de/erstis` on Tuesday 6 - 8 pm
- E-Mail: hi@enoflag.de / mailing list
- Links: `https://enoflag.de` and `https://www.agrs.tu-berlin.de`

Story

How this lecture was created:

- played RuCTFe, had an afterparty
- after drinks and pizza (ca. 1 AM),
  Júlia: "We should teach each other, what we know."
  me: "I could teach crypto."
  next morning, it still seemed a good idea
- Winter 18/19: course of 8 lectures and exercises in AGRS
- since summer 2020 full lecture

Story

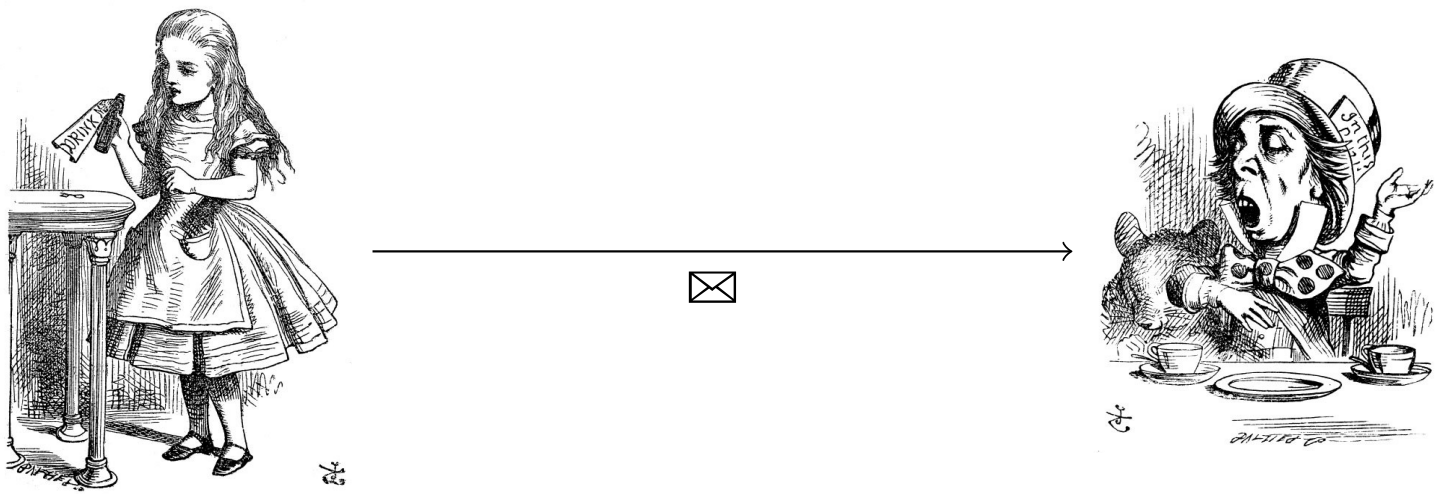
How this lecture was created:

- played RuCTFe, had an afterparty
- after drinks and pizza (ca. 1 AM),
  Júlia: "We should teach each other, what we know."
  me: "I could teach crypto."
  next morning, it still seemed a good idea
- Winter 18/19: course of 8 lectures and exercises in AGRS
- since summer 2020 full lecture

Questions so far?

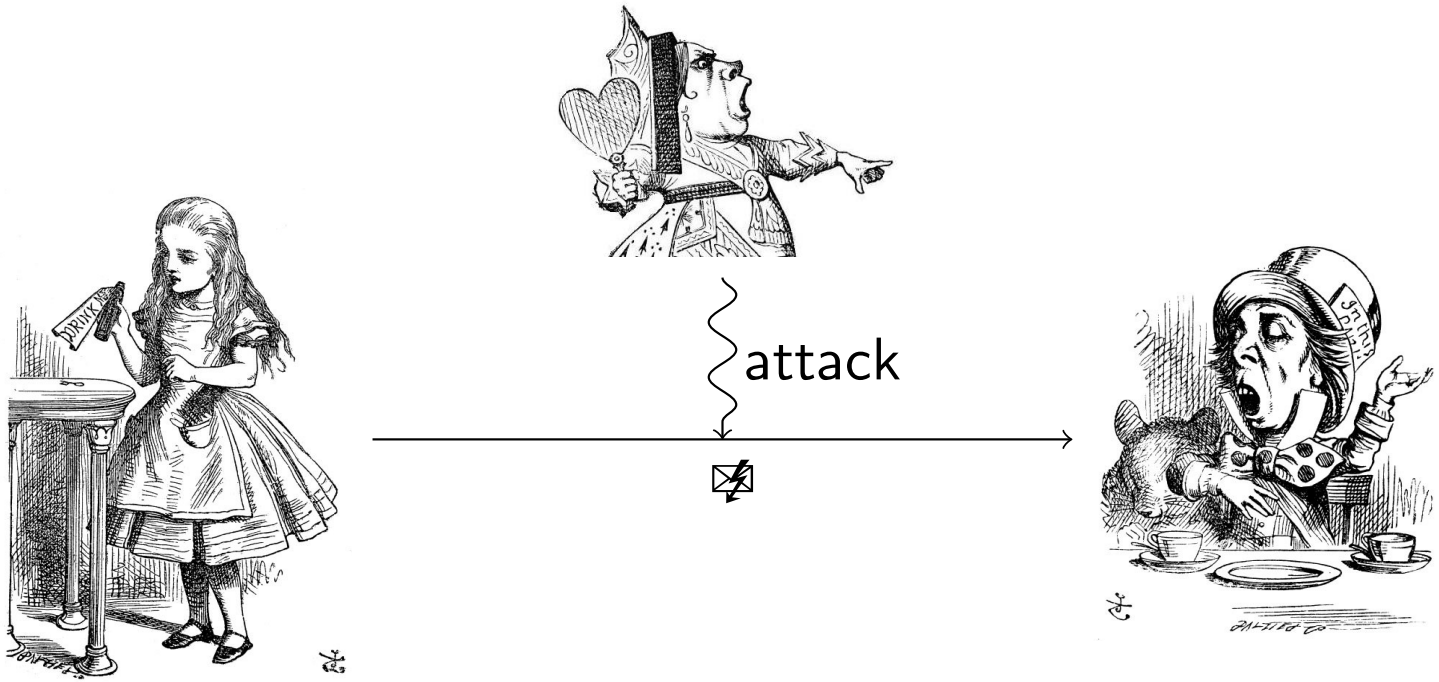
What is Cryptography?

Setting:



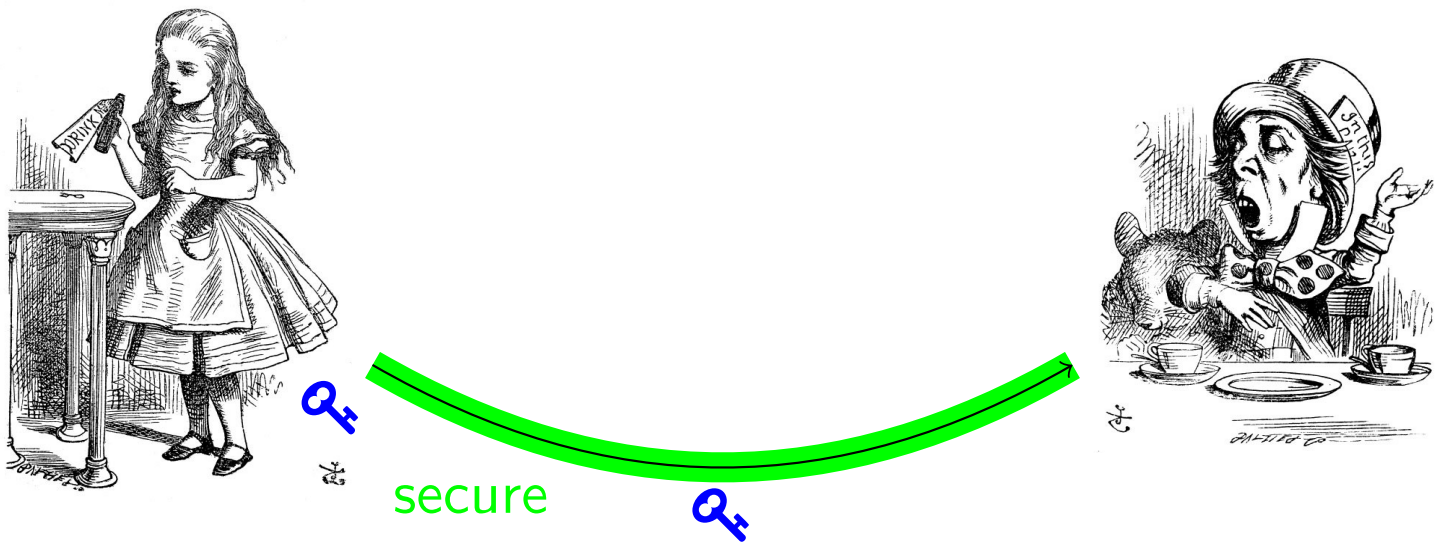
What is Cryptography?

Setting:



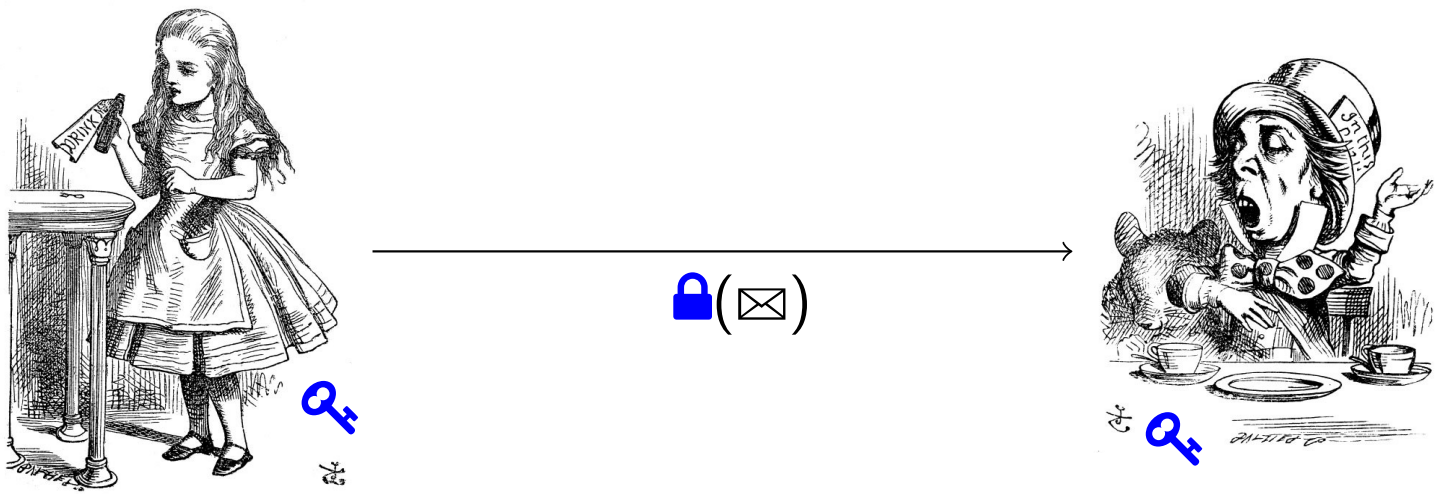
What is Cryptography?

Symmetric:



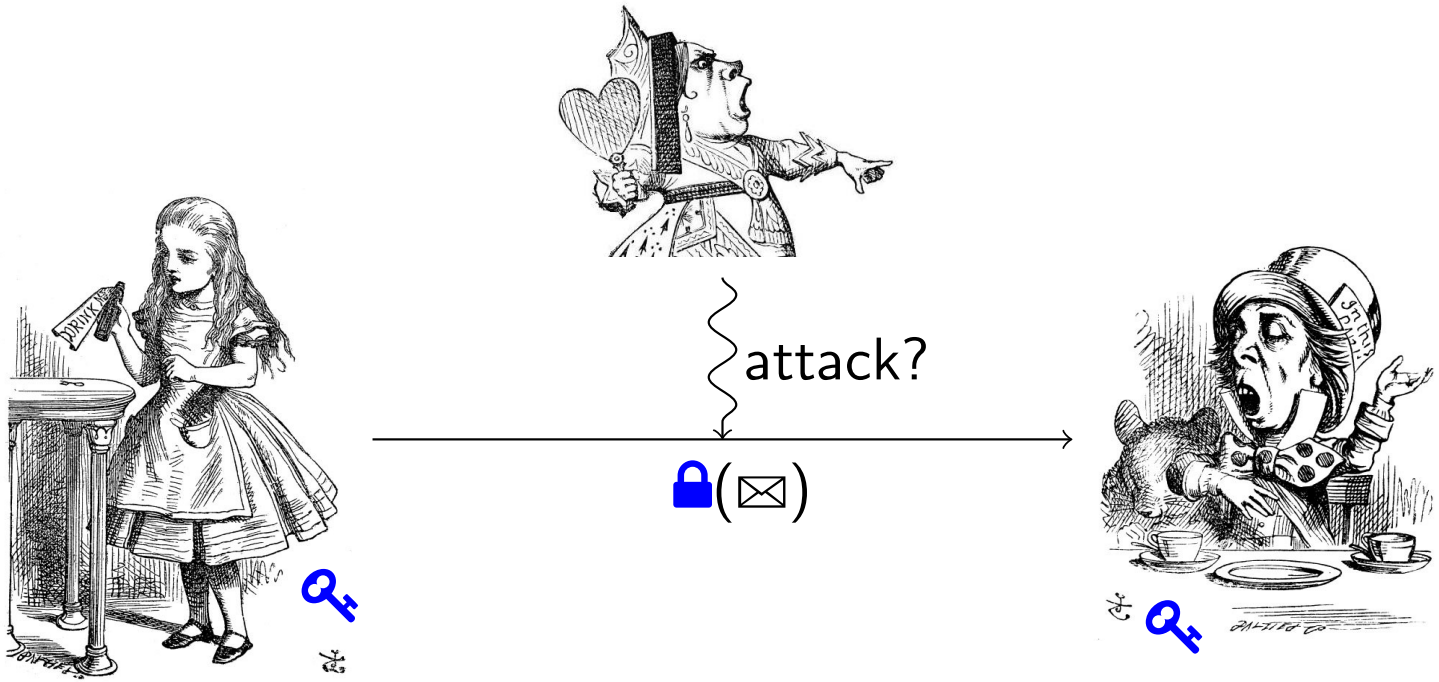
What is Cryptography?

Symmetric:



What is Cryptography?

Symmetric:



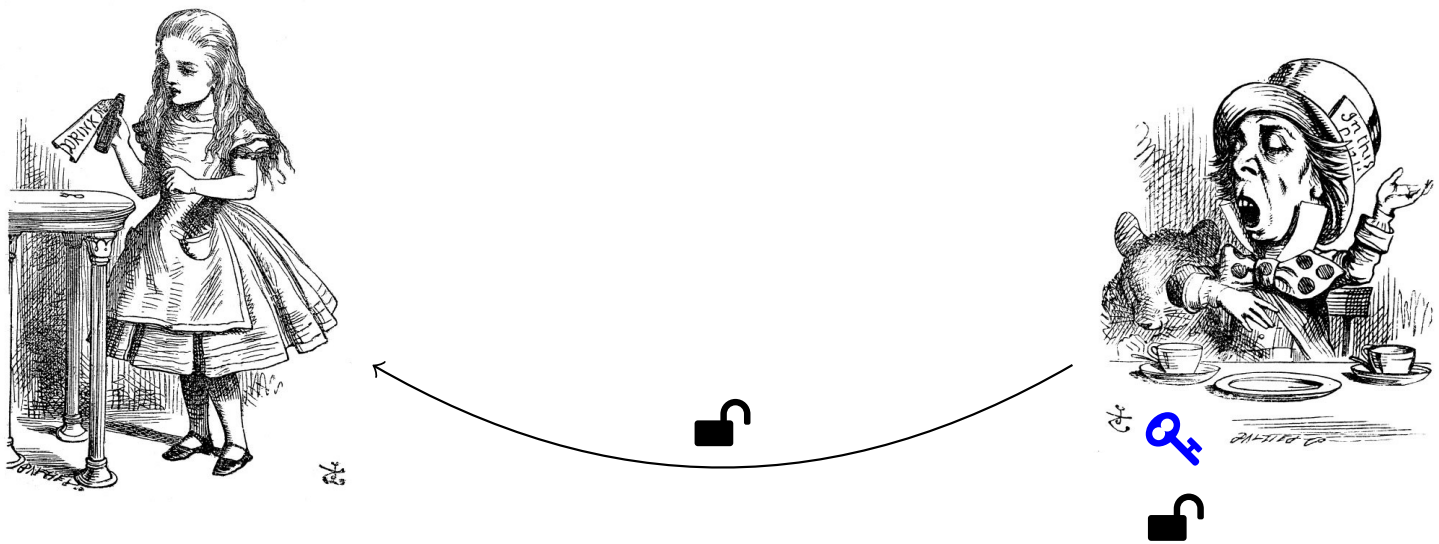
What is Cryptography?

Asymmetric:

What if we do not have a secure connection?

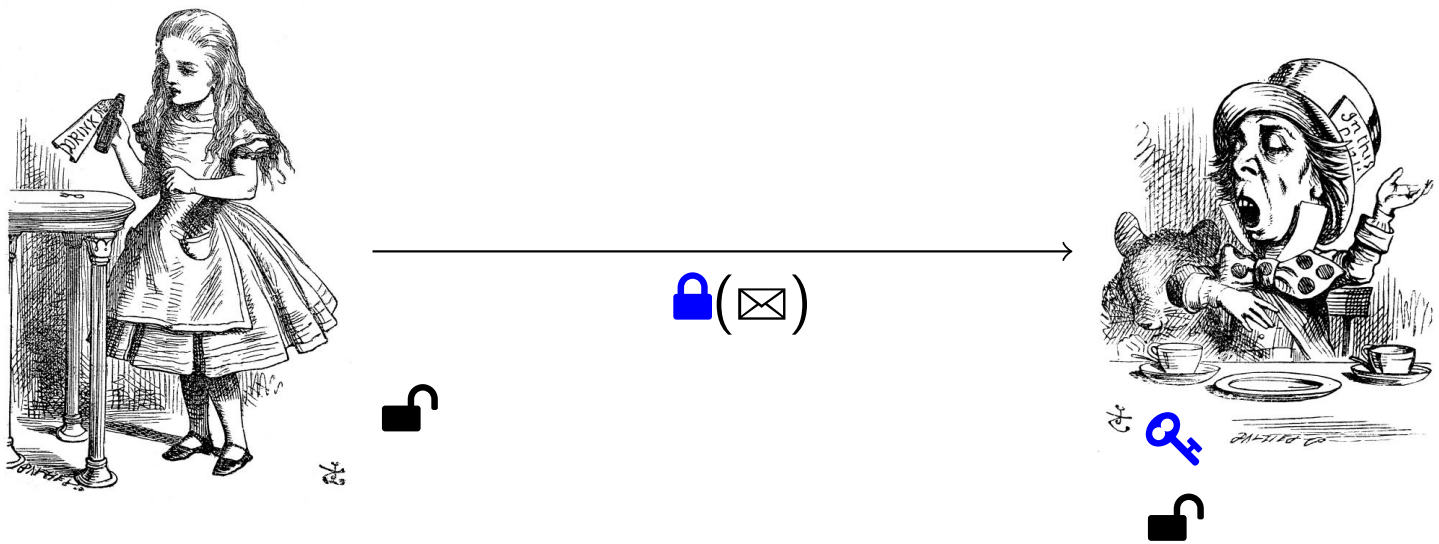
What is Cryptography?

Asymmetric:



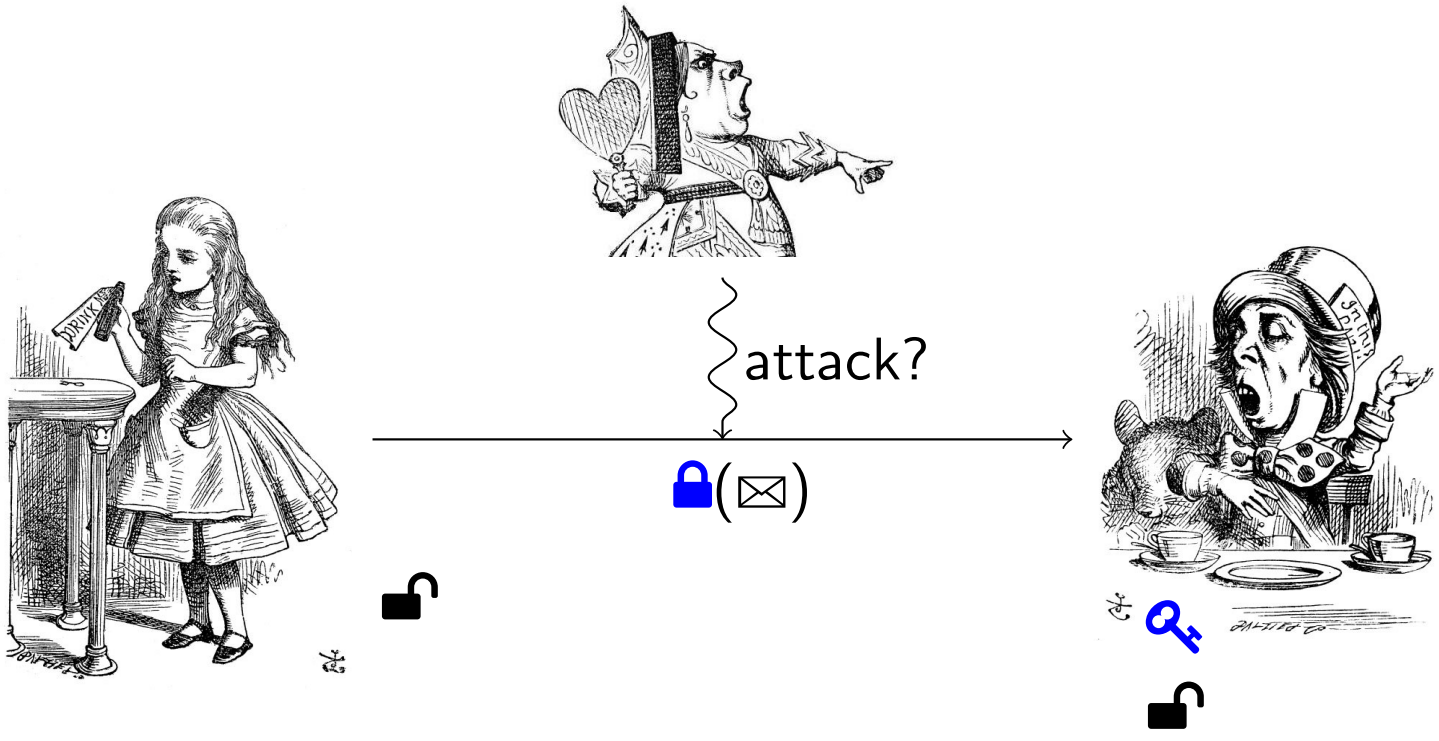
What is Cryptography?

Asymmetric:



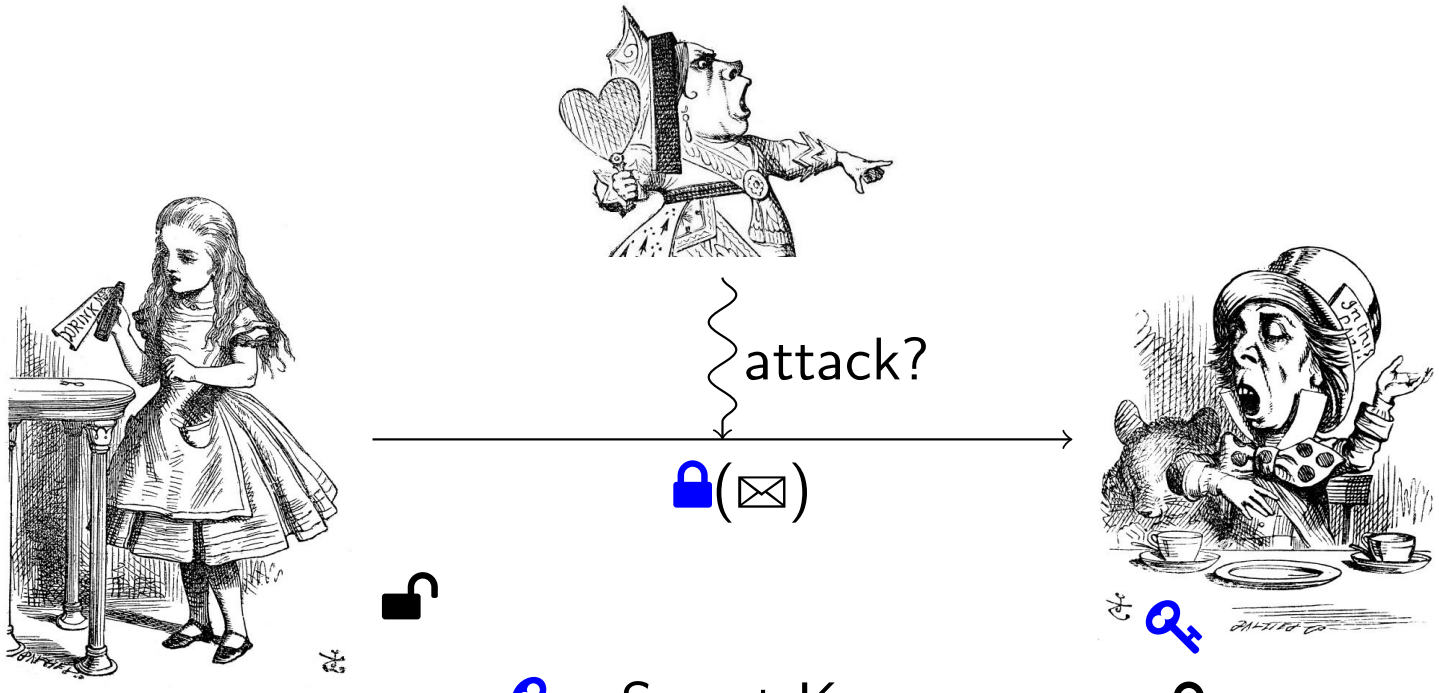
What is Cryptography?

Asymmetric:



What is Cryptography?

Asymmetric:



🔑: Secret Key

🔓: Public Key

What is Cryptography?

cryptography: (in strict sense) design of cryptosystems

cryptoanalysis: breaking encryption

cryptology: both, but often "cryptography" used instead

What is Cryptography?

cryptography: (in strict sense) design of cryptosystems

cryptoanalysis: breaking encryption

cryptology: both, but often "cryptography" used instead

- encrypt: $c \sim \mathrm{enc}(m, k_{\mathrm{enc}})$ (can be ambiguous)
- decrypt: $m = \mathrm{dec}(c, k_{\mathrm{dec}})$

What is Cryptography?

cryptography: (in strict sense) design of cryptosystems

cryptoanalysis: breaking encryption

cryptology: both, but often "cryptography" used instead

- encrypt: $c \sim \mathrm{enc}(m, k_{\mathrm{enc}})$ (can be ambiguous)
- decrypt: $m = \mathrm{dec}(c, k_{\mathrm{dec}})$
- symmetric: $k_{\mathrm{enc}} = k_{\mathrm{dec}}$
- asymmetric: $k_{\mathrm{enc}} \neq k_{\mathrm{dec}}$, but related

What is Cryptography?

cryptography: (in strict sense) design of cryptosystems

cryptoanalysis: breaking encryption

cryptology: both, but often "cryptography" used instead

- encrypt: $c \sim \mathrm{enc}(m, k_{\mathrm{enc}})$ (can be ambiguous)
- decrypt: $m = \mathrm{dec}(c, k_{\mathrm{dec}})$
- symmetric: $k_{\mathrm{enc}} = k_{\mathrm{dec}}$
- asymmetric: $k_{\mathrm{enc}} \neq k_{\mathrm{dec}}$, but related

Kerckhoff's Principle (Open Design)

enc and dec are known,
only key k_{dec} secret (and k_{enc} if both same)

Mathematical Model

Definition (Cryptosystem)

A cryptosystem is a quintuple $(P, C, K, \mathrm{enc}, \mathrm{dec})$ where

- P is the set of all plaintexts
- C is the set of all ciphers
- K is the set of all keys/key pairs
- $\mathrm{enc} : P \times K \rightsquigarrow C$ is the encryption relation (not necessarily a map)
- $\mathrm{dec} : C \times K \to P$ is the decryption function
- $\forall m \in P, k \in K \,.\, \mathrm{dec}\left(\mathrm{enc}\left(m, k\right), k\right) = m$, or $\forall m \in P, (k_{\mathrm{dec}}, k_{\mathrm{enc}}) \in K \,.\, \mathrm{dec}\left(\mathrm{enc}\left(m, k_{\mathrm{enc}}\right), k_{\mathrm{dec}}\right) = m$
- $\mathrm{enc}, \mathrm{dec}$ are efficiently computable

Mathematical Model

Definition (Cryptosystem)

A cryptosystem is a quintuple $(P, C, K, \mathrm{enc}, \mathrm{dec})$ where

- P is the set of all plaintexts
- C is the set of all ciphers
- K is the set of all keys/key pairs
- $\mathrm{enc} : P \times K \rightsquigarrow C$ is the encryption relation (not necessarily a map)
- $\mathrm{dec} : C \times K \to P$ is the decryption function
- $\forall m \in P, k \in K \,.\, \mathrm{dec}\left(\mathrm{enc}\left(m, k\right), k\right) = m$, or $\forall m \in P, (k_{\mathrm{dec}}, k_{\mathrm{enc}}) \in K \,.\, \mathrm{dec}\left(\mathrm{enc}\left(m, k_{\mathrm{enc}}\right), k_{\mathrm{dec}}\right) = m$
- $\mathrm{enc}, \mathrm{dec}$ are efficiently computable

Kerckhoff: The whole cryptosystem in known.

Attack Scenarios

What does Eve know?

CO: ciphertext only

KP: known plaintext, i.e. pairs of cipher and message

CPA: chosen plaintext attack

CCA1: chosen cipher attack, at the beginning, Eve can request decryption for chosen ciphers

CCA2: adaptive chosen cipher attack, after being given the task, Eve can request decryption for chosen ciphers

Attack Scenarios

What does Eve know?

CO: ciphertext only

KP: known plaintext, i.e. pairs of cipher and message

CPA: chosen plaintext attack

CCA1: chosen cipher attack, at the beginning, Eve can request decryption for chosen ciphers

CCA2: adaptive chosen cipher attack, after being given the task, Eve can request decryption for chosen ciphers

Example

- CPA: minimum for public key crypto
- CCA2: impersonate authentication server (ssh login)

Attack Scenarios

What is a success?

OW: one-way, decrypting cipher

NM: non-malleability, change cipher that decryption still yields a meaningful message

PA: plaintext awareness, generate a cipher, whose decryption yields a meaningful message

IND: indistinguishability, which given cipher matches given message answer must be significantly better than guessing

Attack Scenarios

What is a success?

OW: one-way, decrypting cipher

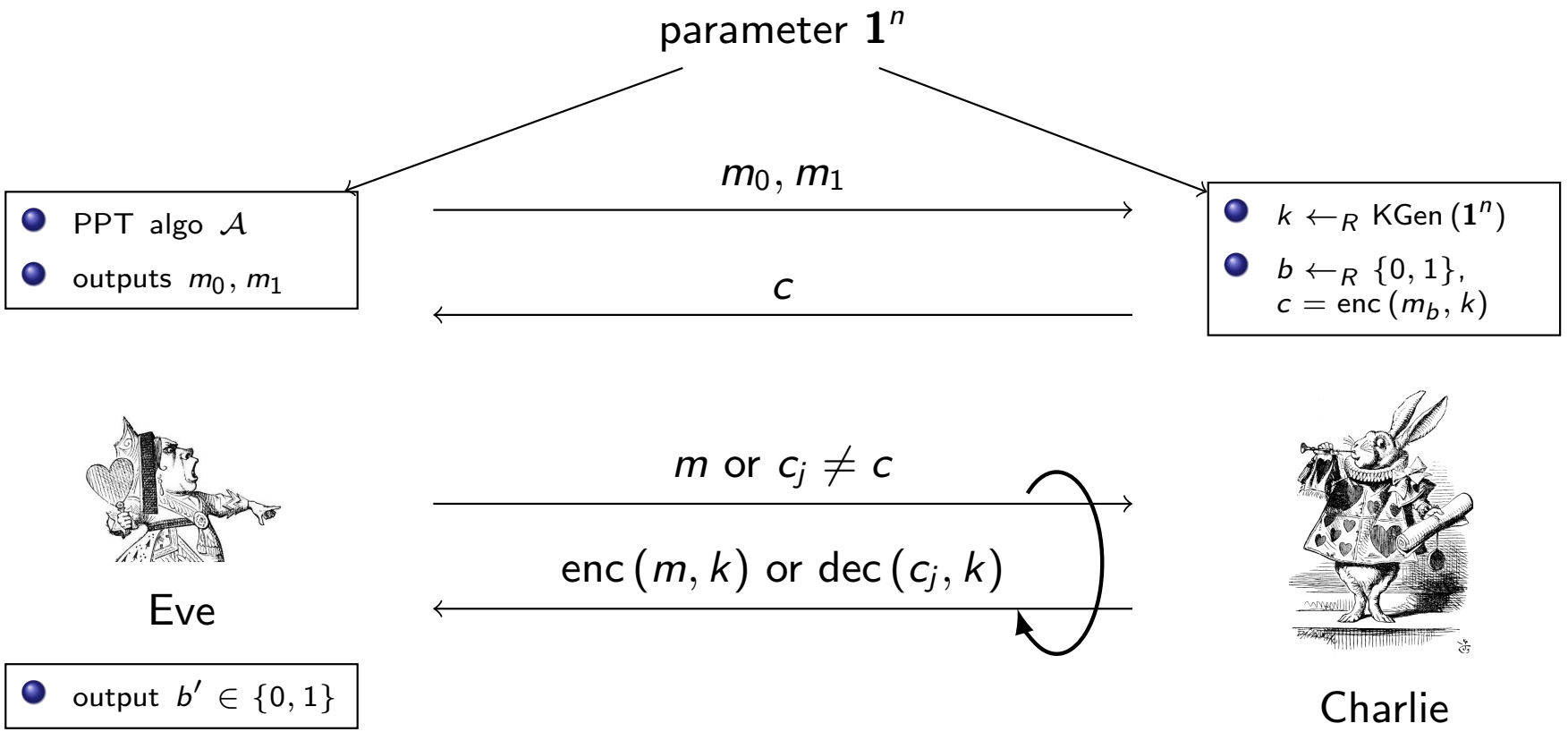
NM: non-malleability, change cipher that decryption still yields a meaningful message

PA: plaintext awareness, generate a cipher, whose decryption yields a meaningful message

IND: indistinguishability, which given cipher matches given message answer must be significantly better than guessing

- combine attacker's power and notion of success
- Strongest goal: IND-CCA2

The IND-CCA2 Game



- Eve wins if $b' = b$
- PPT = probabilistic polynomial time
- secure if for every $\mathcal{A}$ and every polynomial p, Eve's winning chance is less than $\frac{1}{2} + \frac{1}{p(n)}$

Caesar-Cipher

by Albert Uderzo,

taken from

https://asterix.fandom.com/de/wiki/Julius_C%C3%A4sar

100 BC - 44 BC

Simple substitution

- A $\rightarrow$ D
- B $\rightarrow$ E
- C $\rightarrow$ F
- ...

(much) later with arbitrary shift

Polyalphabetic Ciphers

Renaissance

- Johannes Trithemius
- Giovan Battista Bellaso
- Leon Battista Alberti
- **Blaise de Vigenére**

Different Ceaser-ciphers for different letters, depending on keyword

Broken by

- Charles Babbage (1854)
- **Friedrich Wilhelm Kasiski** (1863)

find length of keyword

- search for blocks that occur multiple times
- greatest common divisor of differences of their occurrences $\leadsto$ keylength
- then break separate indices by frequency

Rotor Machines

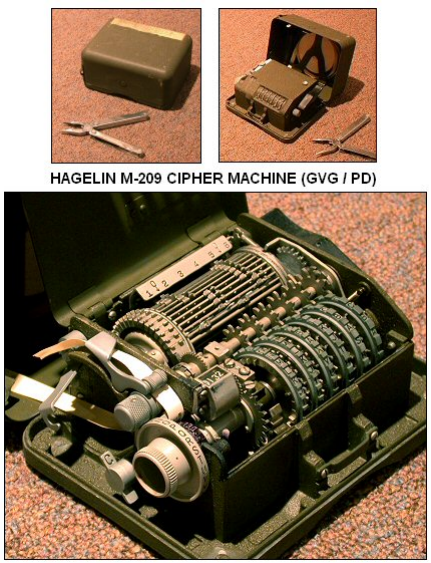
Starting during and after World War I

Enigma by Arthur Scherbius, broken by project "Ultra" (Alan Turing)

M-209 by Boris Hagelin, used by USA,
broken by German cryptoanalysts, from 1943 on

several others

Enigma



M-209

Public Key Cryptography

DH key exchange: Whitfield Diffie, Martin Hellman, 1976

- works in a group
- nowadays mostly elliptic curves over a finite field

RSA: Ron Rivest, Adi Shamir, Leonard Adleman, 1977

- works in the ring of integers modulo n

Public Key Cryptography

DH key exchange: Whitfield Diffie, Martin Hellman, 1976

- works in a group
- nowadays mostly elliptic curves over a finite field

RSA: Ron Rivest, Adi Shamir, Leonard Adleman, 1977

- works in the ring of integers modulo n

Enter Mathematics

Modular Arithmetic

- ring is essentially abstraction of $\mathbb{Z}$
- ring of integers modulo n: notation $\mathbb{Z}_n$
  just append " mod n" to every operation

Modular Arithmetic

- ring is essentially abstraction of $\mathbb{Z}$
- ring of integers modulo n: notation $\mathbb{Z}_n$
  just append " mod n" to every operation

addition: as usual

multiplication: as usual

Modular Arithmetic

- ring is essentially abstraction of $\mathbb{Z}$
- ring of integers modulo n: notation $\mathbb{Z}_n$
  just append " mod n" to every operation

addition: as usual

multiplication: as usual

negation: $-a$ is the number with $a + (-a) = 0$
here $-0 = 0$ or $-a = n - a$ for $a > 0$

Modular Arithmetic

- ring is essentially abstraction of $\mathbb{Z}$
- ring of integers modulo n: notation $\mathbb{Z}_n$
  just append " mod n" to every operation

addition: as usual

multiplication: as usual

negation: $-a$ is the number with $a + (-a) = 0$
here $-0 = 0$ or $-a = n - a$ for $a > 0$

multiplicative inverse: a^{-1} is the number with $a \cdot a^{-1} = 1$

- not always possible
- works iff $\gcd(a, n) = 1$
- if n prime, works for all $0 < a < n$

Example Ring – CPU/ALU

- modern CPU uses 64 Bit $\rightsquigarrow$ can save 2^{64} numbers
- all computations run modulo 2^{64}
- $1 \ldots 1_2 = 2^{64} - 1 = -1$
- for arithmetic, ALU does not care about signed/unsigned

Example Ring – CPU/ALU

- modern CPU uses 64 Bit $\rightsquigarrow$ can save 2^{64} numbers
- all computations run modulo 2^{64}
- $1 \ldots 1_2 = 2^{64} - 1 = -1$
- for arithmetic, ALU does not care about signed/unsigned

"Negative" Numbers

- $-a$ is the number that satisfies $a + (-a) = 0$
- say $\overline{a}$ is a with all bits flipped,
- $a + \overline{a} = 1 \ldots 1$ (in every bit add 0 and 1)
- $a + \overline{a} + 1 = (1)0 \ldots 0 = 0$ (overflow)
- hence $-a = \overline{a} + 1$

Algorithms

Need algorithms for the following: (b bits input)

- addition, subtraction,
- efficient multiplication
- division with remainder in $\mathbb{Z}$, in particular modulo-operator
- division/multiplicative inverse in $\mathbb{Z}_p$ (or $\mathbb{Z}_n$, if possible)

Algorithms

Need algorithms for the following: (b bits input)

- ✓ addition, subtraction,
  efficient multiplication
  division with remainder in $\mathbb{Z}$, in particular modulo-operator
  division/multiplicative inverse in $\mathbb{Z}_p$ (or $\mathbb{Z}_n$, if possible)

Addition/Subtraction

naive approach: digit-wise, with carry bit $\rightsquigarrow \mathcal{O}(b)$

Algorithms

Need algorithms for the following: (b bits input)

- ✓ addition, subtraction,
- ✓ efficient multiplication

  division with remainder in $\mathbb{Z}$, in particular modulo-operator

  division/multiplicative inverse in $\mathbb{Z}_p$ (or $\mathbb{Z}_n$, if possible)

Addition/Subtraction

naive approach: digit-wise, with carry bit $\leadsto \mathcal{O}(b)$

Multiplication

- naive/school-method: $\mathcal{O}(b^2)$
- Karatsuba: divide-and-conquer, $\mathcal{O}\left(b^{\log_2 3}\right)$
- Fast-Fourier-Transformation: $\mathcal{O}(b \log b)$

Algorithms

Need algorithms for the following: (b bits input)

- ✓ addition, subtraction,
- ✓ efficient multiplication
- ✓ division with remainder in $\mathbb{Z}$, in particular modulo-operator
  division/multiplicative inverse in $\mathbb{Z}_p$ (or $\mathbb{Z}_n$, if possible)

Addition/Subtraction

naive approach: digit-wise, with carry bit $\rightsquigarrow \mathcal{O}(b)$

Multiplication

- naive/school-method: $\mathcal{O}(b^2)$
- Karatsuba: divide-and-conquer, $\mathcal{O}\left(b^{\log_2 3}\right)$
- Fast-Fourier-Transformation: $\mathcal{O}(b \log b)$

Division

reduce to multiplication, same complexity

Algorithms

Theorem (Extended Euclidian Algorithm)

For all $a, b \in \mathbb{Z}$ there are $s, t \in \mathbb{Z}$ such that

$$s \cdot a + t \cdot b = \gcd(a, b)$$

Algorithms

Theorem (Extended Euclidian Algorithm)

For all $a, b \in \mathbb{Z}$ there are $s, t \in \mathbb{Z}$ such that

$$s \cdot a + t \cdot b = \gcd(a, b)$$

```
def EEA(a,b):
  if b == 0: return (a,1,0)
  d,s,t = EEA(b, a % b)
  return (d, t, s - (a//b) * t)
```

Algorithms

Theorem (Extended Euclidian Algorithm)

For all $a, b \in \mathbb{Z}$ there are $s, t \in \mathbb{Z}$ such that

$$s \cdot a + t \cdot b = \gcd(a, b)$$

```
def EEA(a,b):
  if b == 0: return (a,1,0)
  d,s,t = EEA(b, a % b)
  return (d, t, s - (a//b) * t)
```

Modular Inverse in $\mathbb{Z}_n$

- assume $\gcd(a, n) = 1$, (always works if n prime and $0 < a < n$)
- compute $d, s, t = \mathrm{EEA}(a, n)$, clearly $d = 1$

Algorithms

Theorem (Extended Euclidian Algorithm)

For all $a, b \in \mathbb{Z}$ there are $s, t \in \mathbb{Z}$ such that

$$s \cdot a + t \cdot b = \gcd(a, b)$$

```
def EEA(a,b):
  if b == 0: return (a,1,0)
  d,s,t = EEA(b, a % b)
  return (d, t, s - (a//b) * t)
```

Modular Inverse in $\mathbb{Z}_n$

- assume $\gcd(a, n) = 1$, (always works if n prime and $0 < a < n$)
- compute $d, s, t = \mathrm{EEA}(a, n)$, clearly $d = 1$
- $1 = s \cdot a + t \cdot n$ means $s \cdot a \equiv 1 \mod n$
- so $s = a^{-1}$ in $\mathbb{Z}_n$

Chinese Remainder Theorem

Theorem (Chinese Remainder Theorem (CRT))

Let $n_i \in \mathbb{Z}$ (pairwise) coprime, $a_i \in \mathbb{Z}$ arbitrary for $i = 1, \ldots, k$. Then the system

$$a_i \equiv x \mod n_i \qquad i = 1, \ldots, k$$

has a unique solution $0 \leq x < \prod n_i$.

Chinese Remainder Theorem

Theorem (Chinese Remainder Theorem (CRT))

Let $n_i \in \mathbb{Z}$ (pairwise) coprime, $a_i \in \mathbb{Z}$ arbitrary for $i = 1, \ldots, k$. Then the system

$$a_i \equiv x \mod n_i \qquad i = 1, \ldots, k$$

has a unique solution $0 \leq x < \prod n_i$.

Algorithm for 2 congruences:

$$1, s, t = \text{EEA}(n_1, n_2) \rightsquigarrow \qquad 1 = s \cdot n_1 + t \cdot n_2$$
$$\text{solution} \qquad x := a_2 \cdot s \cdot n_1 + a_1 \cdot t \cdot n_2$$

continue recursively with: $a' = x$ and $n' = n_1 \cdot n_2$

Example (CRT)

- Consider the system

$$x \equiv 1 \mod 3$$
$$x \equiv 2 \mod 5$$
$$x \equiv 3 \mod 7$$

Example (CRT)

- Consider the system

$$\begin{aligned} x &\equiv 1 \mod 3 \\ x &\equiv 2 \mod 5 \\ x &\equiv 3 \mod 7 \end{aligned}$$

- combine first two:

$$\mathrm{EEA}(3,5) = (1,2,-1) \qquad \text{as } 1 = 2 \cdot 3 + (-1) \cdot 5$$
$$\rightsquigarrow a' = 2 \cdot 2 \cdot 3 + 1 \cdot (-1) \cdot 5 = 7$$

Example (CRT)

- Consider the system

$$\begin{aligned} x &\equiv 1 \mod 3 \\ x &\equiv 2 \mod 5 \\ x &\equiv 3 \mod 7 \end{aligned}$$

- combine first two:

$$\begin{aligned} \mathrm{EEA}(3,5) &= (1,2,-1) && \text{as } 1 = 2 \cdot 3 + (-1) \cdot 5 \\ &\rightsquigarrow a' = 2 \cdot 2 \cdot 3 + 1 \cdot (-1) \cdot 5 = 7 \end{aligned}$$

- reduced system, continue recursively

$$\begin{aligned} x &\equiv 7 \mod 15 \\ x &\equiv 3 \mod 7 \end{aligned}$$

Fermat's Little Theorem

Theorem

Let p prime, a arbitrary, then $a^p \equiv a \mod p$.

Fermat's Little Theorem

Theorem

Let p prime, a arbitrary, then $a^p \equiv a \mod p$.

Proof by induction on a:

Base: $a = 0$ ✓

Fermat's Little Theorem

Theorem

Let p prime, a arbitrary, then $a^p \equiv a \mod p$.

Proof by induction on a:

Base: $a = 0$ ✓

Step:

$$(a+1)^p = \sum_{k=0}^{p} \binom{p}{k} a^k = a^p + 1 + \sum_{k=1}^{p-1} \underbrace{\frac{p(p-1)\dots(p-k+1)}{1 \cdot 2 \cdot \dots \cdot k}}_{p \text{ divides this}} a^k$$

$$\equiv a^p + 1 \overset{IH}{\equiv} a + 1 \mod p$$

□

Fermat's Little Theorem

Theorem

Let p prime, a arbitrary, then $a^p \equiv a \mod p$.

Proof by induction on a:

Base: $a = 0$ ✓

Step:

$$(a+1)^p = \sum_{k=0}^{p} \binom{p}{k} a^k = a^p + 1 + \sum_{k=1}^{p-1} \underbrace{\frac{p(p-1)\dots(p-k+1)}{1 \cdot 2 \cdot \dots \cdot k}}_{p \text{ divides this}} a^k$$

$$\equiv a^p + 1 \overset{IH}{\equiv} a + 1 \mod p$$

□

Corollary (Alternative formulation)

p prime, a coprime to p (i.e. no multiple), then $a^{p-1} \equiv 1 \mod p$.

Euler's Theorem

Generalise Fermat's Little Theorem:

Definition (Euler's Phi-Function)

$$\varphi(n) := |\mathbb{Z}_n^*| = |\{a \in \mathbb{Z}_n : \gcd(a, n) = 1\}|$$

Euler's Theorem

Generalise Fermat's Little Theorem:

Definition (Euler's Phi-Function)

$$\varphi(n) := |\mathbb{Z}_n^*| = |\{a \in \mathbb{Z}_n : \gcd(a, n) = 1\}|$$

Lemma (How to compute $\varphi(n)$?)

Let $n = \prod p_i^{e_i}$ factorisation. Then $\varphi(n) = \prod (p_i - 1) \cdot p_i^{e_i - 1}$.

Euler's Theorem

Generalise Fermat's Little Theorem:

Definition (Euler's Phi-Function)

$$\varphi(n) := |\mathbb{Z}_n^*| = |\{a \in \mathbb{Z}_n : \gcd(a, n) = 1\}|$$

Lemma (How to compute $\varphi(n)$?)

Let $n = \prod p_i^{e_i}$ factorisation. Then $\varphi(n) = \prod (p_i - 1) \cdot p_i^{e_i - 1}$.

Theorem (Euler)

Let $n \geq 2$ and $a \in \mathbb{Z}_n^*$. Then $a^{\varphi(n)} \equiv 1 \mod n$.

Euler's Theorem

Generalise Fermat's Little Theorem:

Definition (Euler's Phi-Function)

$$\varphi(n) := |\mathbb{Z}_n^*| = |\{a \in \mathbb{Z}_n : \gcd(a, n) = 1\}|$$

Lemma (How to compute $\varphi(n)$?)

Let $n = \prod p_i^{e_i}$ factorisation. Then $\varphi(n) = \prod (p_i - 1) \cdot p_i^{e_i - 1}$.

Theorem (Euler)

Let $n \geq 2$ and $a \in \mathbb{Z}_n^*$. Then $a^{\varphi(n)} \equiv 1 \mod n$.

Special case: $n = p$ prime, $\varphi(p) = p - 1$, exactly Fermat
Proof e.g. via group theory (Lagrange's Theorem).

Prime Number Theorem

Theorem (Prime-Number-Theorem)

Let $\pi(n)$ denote the number of primes up to n. Then $\pi(n) \sim \frac{n}{\ln(n)}$.

Prime Number Theorem

Theorem (Prime-Number-Theorem)

Let $\pi(n)$ denote the number of primes up to n. Then $\pi(n) \sim \frac{n}{\ln(n)}$.

There are more primes than you would think.

Example (Make a guess)

$$\pi(100) =$$
$$\pi(10\ 000) =$$

Prime Number Theorem

Theorem (Prime-Number-Theorem)

Let $\pi(n)$ denote the number of primes up to n. Then $\pi(n) \sim \frac{n}{\ln(n)}$.

There are more primes than you would think.

Example (Make a guess)

$$\pi(100) = 25 \qquad \sim 1/4 \text{ of numbers}$$
$$\pi(10\ 000) = 1229 \qquad \sim 1/8 \text{ of numbers}$$

RSA

RSA

- THE classic in Public Key Cryptography
  besides Diffie-Hellman key-exchange ($\rightsquigarrow$ later section)
- published April, 1977
- simple design, scheme yet unbroken

RSA

- THE classic in Public Key Cryptography
  besides Diffie-Hellman key-exchange ($\rightsquigarrow$ later section)
- published April, 1977
- simple design, scheme yet unbroken
- named after
  - Ron **R**ivest
  - Adi **S**hamir
  - Leonard **A**dleman
- featured in Martin Gardner's "Mathematical Games", Aug 1977;
  including the first RSA-challenge (129 decimal digits, 100$),

RSA

- THE classic in Public Key Cryptography
  besides Diffie-Hellman key-exchange ($\rightsquigarrow$ later section)
- published April, 1977
- simple design, scheme yet unbroken
- named after
  - Ron **R**ivest
  - Adi **S**hamir
  - Leonard **A**dleman
- featured in Martin Gardner's "Mathematical Games", Aug 1977;
  including the first RSA-challenge (129 decimal digits, 100$),
  solved in 1994,
- already included idea of signature via RSA

RSA

Setup:

- p, q primes
- $n := p \cdot q \implies \varphi(n) = (p-1)(q-1)$
- choose e coprime to $\varphi(n)$
- $d := e^{-1} \bmod \varphi(n)$ (extended Euclidean Algorithm)

RSA

Setup:

- p, q primes
- $n := p \cdot q \implies \varphi(n) = (p-1)(q-1)$
- choose e coprime to $\varphi(n)$
- $d := e^{-1} \bmod \varphi(n)$ (extended Euclidean Algorithm)

Keys:

- public key (n, e)
- private key (n, d), possibly $p, q, \varphi(n)$

RSA

Setup:

- p, q primes
- $n := p \cdot q \implies \varphi(n) = (p-1)(q-1)$
- choose e coprime to $\varphi(n)$
- $d := e^{-1} \bmod \varphi(n)$ (extended Euclidean Algorithm)

Keys:

- public key (n, e)
- private key (n, d), possibly $p, q, \varphi(n)$

Usage:

- encrypt $c = m^e \bmod n$
- decrypt $m = c^d \bmod n$

Example (Key Generation)

- choose primes $p = 47$ and $q = 97$, yields $n = 4559$
- choose $e = 17$
- $\varphi(n) = (p-1)(q-1) = 4416$
- `_,d,_ = EEA(e, phi)`, yields $d = 3377$

Example (Key Generation)

- choose primes $p = 47$ and $q = 97$, yields $n = 4559$
- choose $e = 17$
- $\varphi(n) = (p-1)(q-1) = 4416$
- `_,d,_ = EEA(e, phi)`, yields $d = 3377$

Example (En-/Decryption)

- message $m = 102$ (first letter of `flag...`)
- encrypt: cipher

$$c = m^e \bmod n = 102^{17} \bmod 4559 = 2993$$

- decrypt: get back message

$$m = c^d \bmod n = 2993^{3377} \bmod 4559 = 102$$

Correctness of RSA

Theorem

For every message $0 \leq m < n$ *we have* $m = (m^e)^d \bmod n$.

Correctness of RSA

Theorem

For every message $0 \leq m < n$ *we have* $m = (m^e)^d \bmod n$.

slightly wrong "proof".

$$m^{ed} \equiv m^{1+k\varphi(n)} \equiv m \cdot \left(m^{\varphi(n)}\right)^k \equiv m \cdot 1^k \equiv m \mod n$$

by Euler.

Correctness of RSA

Theorem

For every message $0 \leq m < n$ *we have* $m = (m^e)^d \bmod n$.

slightly wrong "proof".

$$m^{ed} \equiv m^{1+k\varphi(n)} \equiv m \cdot \left(m^{\varphi(n)}\right)^k \equiv m \cdot 1^k \equiv m \mod n$$

by Euler. But this only works for m, n coprime. □

Correctness of RSA

Theorem

For every message $0 \leq m < n$ *we have* $m = (m^e)^d \bmod n$.

slightly wrong "proof".

$$m^{ed} \equiv m^{1+k\varphi(n)} \equiv m \cdot \left(m^{\varphi(n)}\right)^k \equiv m \cdot 1^k \equiv m \mod n$$

by Euler. But this only works for m, n coprime. □

Proof.

If $p \mid m$, then $m^{ed} \equiv 0 \equiv m \mod p$.

Correctness of RSA

Theorem

For every message $0 \leq m < n$ *we have* $m = (m^e)^d \bmod n$.

slightly wrong "proof".

$$m^{ed} \equiv m^{1+k\varphi(n)} \equiv m \cdot \left(m^{\varphi(n)}\right)^k \equiv m \cdot 1^k \equiv m \mod n$$

by Euler. But this only works for m, n coprime. □

Proof.

If $p \mid m$, then $m^{ed} \equiv 0 \equiv m \mod p$. Else

$$m^{ed} \equiv m^{1+k(q-1)(p-1)} \equiv m \cdot \left(m^{p-1}\right)^{k(q-1)} \equiv m \cdot 1^{k(q-1)} \equiv m \bmod p$$

So $m^{ed} \equiv m \mod p$.

Correctness of RSA

Theorem

For every message $0 \leq m < n$ *we have* $m = (m^e)^d \bmod n$.

slightly wrong "proof".

$$m^{ed} \equiv m^{1+k\varphi(n)} \equiv m \cdot \left(m^{\varphi(n)}\right)^k \equiv m \cdot 1^k \equiv m \mod n$$

by Euler. But this only works for m, n coprime. □

Proof.

If $p \mid m$, then $m^{ed} \equiv 0 \equiv m \mod p$. Else

$$m^{ed} \equiv m^{1+k(q-1)(p-1)} \equiv m \cdot \left(m^{p-1}\right)^{k(q-1)} \equiv m \cdot 1^{k(q-1)} \equiv m \bmod p$$

So $m^{ed} \equiv m \mod p$. Analogue for q.
Hence $m^{ed} \equiv m \mod n$ by CRT. □

RSA Details

Generating primes of B bit

- generate random bit sequence $p = p_{B-1} \dots p_1 1$ (last bit 1)
  (Random Number Generators $\rightsquigarrow$ "Cryptography for Security")

RSA Details

Generating primes of B bit

- generate random bit sequence $p = p_{B-1} \dots p_1 1$ (last bit 1)
  (Random Number Generators $\rightsquigarrow$ "Cryptography for Security")
- test, whether p is (probably) prime ($\rightsquigarrow$ later section)

RSA Details

Generating primes of B bit

- generate random bit sequence $p = p_{B-1} \dots p_1 1$ (last bit 1)
  (Random Number Generators $\rightsquigarrow$ "Cryptography for Security")
- test, whether p is (probably) prime ($\rightsquigarrow$ later section)
- chance: $\sim \frac{\frac{2^B}{\ln 2^B}}{2^B} = \frac{1}{\ln 2^B} \sim \frac{1}{B}$ (Prime Number Theorem)

RSA Details

Generating primes of B bit

- generate random bit sequence $p = p_{B-1} \dots p_1 1$ (last bit 1)
  (Random Number Generators $\rightsquigarrow$ "Cryptography for Security")
- test, whether p is (probably) prime ($\rightsquigarrow$ later section)
- chance: $\sim \frac{\frac{2^B}{\ln 2^B}}{2^B} = \frac{1}{\ln 2^B} \sim \frac{1}{B}$ (Prime Number Theorem)

Choosing e

- pick $e = 65537 = 2^{16} + 1 = 0x10001$
- does not work $\implies$ new primes

RSA Details

Generating primes of B bit

- generate random bit sequence $p = p_{B-1} \dots p_1 1$ (last bit 1)
  (Random Number Generators $\rightsquigarrow$ "Cryptography for Security")
- test, whether p is (probably) prime ($\rightsquigarrow$ later section)
- chance: $\sim \frac{\frac{2^B}{\ln 2^B}}{2^B} = \frac{1}{\ln 2^B} \sim \frac{1}{B}$ (Prime Number Theorem)

Choosing e

- pick $e = 65537 = 2^{16} + 1 = 0x10001$
- does not work $\implies$ new primes
- alternatively:
  $e \in \{3, 5, 17, 257, 65537\} = \{2^{2^k} + 1 : k = 0, \dots, 4\}$

RSA Details

Generating primes of B bit

- generate random bit sequence $p = p_{B-1} \dots p_1 1$ (last bit 1)
  (Random Number Generators $\rightsquigarrow$ "Cryptography for Security")
- test, whether p is (probably) prime ($\rightsquigarrow$ later section)
- chance: $\sim \frac{\frac{2^B}{\ln 2^B}}{2^B} = \frac{1}{\ln 2^B} \sim \frac{1}{B}$ (Prime Number Theorem)

Choosing e

- pick $e = 65537 = 2^{16} + 1 = 0x10001$
- does not work $\implies$ new primes
- alternatively:
  $e \in \{3, 5, 17, 257, 65537\} = \{2^{2^k} + 1 : k = 0, \dots, 4\}$
  Fermat primes: coprime iff $e \nmid \varphi(n)$,
  $e = 10 \dots 01_2$, only $2^k + 1 \leq 17$ multiplications $\rightsquigarrow$ fast

Modular Exponentiation

Need to compute $a^b \mod n$

Modular Exponentiation

Need to compute $a^b \mod n$

Naive approach 

- $(b-1)$ multiplications, one modulo
- huge intermediate results, size $b \cdot \log a$ instead of $\log n$

Modular Exponentiation

Need to compute $a^b \mod n$

Naive approach 

- $(b-1)$ multiplications, one modulo
- huge intermediate results, size $b \cdot \log a$ instead of $\log n$

Square-and-Multiply

1: **function** $\textsc{Pow}(a, b, n)$
2: $\quad c \leftarrow 1$
3: $\quad$ **for** $i = \log b, \ldots, 0$ **do**
4: $\quad\quad c \leftarrow c^2 \mod n$
5: $\quad\quad$ **if** $b_i = 1$ **then** $\quad \triangleright$ `if(b & (1 << i))`
6: $\quad\quad\quad c \leftarrow c \cdot a \mod n$
7: $\quad$ **return** c

Modular Exponentiation

Need to compute $a^b \mod n$

Naive approach 

- $(b-1)$ multiplications, one modulo
- huge intermediate results, size $b \cdot \log a$ instead of $\log n$

Square-and-Multiply

1: **function** $\textsc{Pow}(a, b, n)$
2: $\quad c \leftarrow 1$
3: $\quad$ **for** $i = \log b, \ldots, 0$ **do**
4: $\quad\quad c \leftarrow c^2 \mod n$
5: $\quad\quad$ **if** $b_i = 1$ **then** $\quad \triangleright$ `if(b & (1 << i))`
6: $\quad\quad\quad c \leftarrow c \cdot a \mod n$
7: $\quad$ **return** c

total: $\leq 2 \log b$ mult. and mod of size $\log n$

Example

- modulus $n = 4559$
- public exponent $e = 17 = 2^4 + 1$
- message $m = 102$

Further computations in $\mathbb{Z}_n$:

$$
\begin{aligned}
c &:= 102 && && && m^1 \bmod n \\
c &:= 102^2 && = 1286 && && m^2 \bmod n \\
c &:= 1286^2 && = 3438 && && m^4 \bmod n \\
c &:= 3438^2 && = 2916 && && m^8 \bmod n \\
c &:= 2916^2 && = 521 && && m^{16} \bmod n \\
c &:= 521 \cdot 102 && = 2993 && && m^{17} \bmod n
\end{aligned}
$$

In total: 5 multiplications

Further Optimisation

During setup also compute (once)

$$d_p = d \bmod p - 1 \qquad d_q = d \bmod q - 1 \qquad q_{\text{inv}} = q^{-1} \bmod p$$

Further Optimisation

During setup also compute (once)

$$d_p = d \bmod p - 1 \qquad d_q = d \bmod q - 1 \qquad q_{\text{inv}} = q^{-1} \bmod p$$

decryption

$$c_p = c \bmod p \qquad c_q = c \bmod q$$

$$m_p = c_p^{d_p} \bmod p \qquad m_q = c_q^{d_q} \bmod q$$

$$h = q_{\text{inv}}(m_p - m_q) \bmod p$$

$$m = m_q + hq \bmod n$$

Further Optimisation

During setup also compute (once)

$$d_p = d \bmod p-1 \qquad d_q = d \bmod q-1 \qquad q_{\text{inv}} = q^{-1} \bmod p$$

decryption

$$c_p = c \bmod p \qquad c_q = c \bmod q$$

$$m_p = c_p^{d_p} \bmod p \qquad m_q = c_q^{d_q} \bmod q$$

$$h = q_{\text{inv}}(m_p - m_q) \bmod p$$

$$m = m_q + hq \bmod n$$

Proof of correctness.

$$c^d \equiv c_q^{k\cdot(q-1)+d_q} \equiv c_q^{d_q} \equiv m_q \equiv m_q + hq \equiv m \pmod q$$

$$c^d \equiv c_p^{d_p} \equiv m_p \equiv m_q + qq^{-1}(m_p - m_q) \equiv m_q + hq \equiv m \pmod p$$

Further Optimisation

During setup also compute (once)

$$d_p = d \bmod p-1 \qquad d_q = d \bmod q-1 \qquad q_{\text{inv}} = q^{-1} \bmod p$$

decryption

$$c_p = c \bmod p \qquad c_q = c \bmod q$$
$$m_p = c_p^{d_p} \bmod p \qquad m_q = c_q^{d_q} \bmod q$$

$$h = q_{\text{inv}}(m_p - m_q) \bmod p$$
$$m = m_q + hq \bmod n$$

Proof of correctness.

$$c^d \equiv c_q^{k\cdot(q-1)+d_q} \equiv c_q^{d_q} \equiv m_q \equiv m_q + hq \equiv m \pmod q$$
$$c^d \equiv c_p^{d_p} \equiv m_p \equiv m_q + qq^{-1}(m_p - m_q) \equiv m_q + hq \equiv m \pmod p$$

(Special cases $p \mid c$ and $q \mid c$.) Hence, $m = c^d \bmod n$ by CRT. □

Complexity Analysis

$$c_p = c \bmod p \qquad c_q = c \bmod q$$

$$m_p = c_p^{d_p} \bmod p \qquad m_q = c_q^{d_q} \bmod q$$

$$h = q_{\text{inv}}(m_p - m_q) \bmod p$$

$$m = m_q + hq \bmod n$$

Assume $\log d = \log n = B$, and $\log p = \log q = \frac{B}{2}$,
d, d_p, d_q equally many 0s and 1s

Complexity Analysis

$$c_p = c \bmod p \qquad c_q = c \bmod q$$
$$m_p = c_p^{d_p} \bmod p \qquad m_q = c_q^{d_q} \bmod q$$

$$h = q_{\text{inv}}(m_p - m_q) \bmod p$$
$$m = m_q + hq \bmod n$$

Assume $\log d = \log n = B$, and $\log p = \log q = \frac{B}{2}$,
d, d_p, d_q equally many 0s and 1s

normal $\sim \frac{3}{2}B$ mult. of size B

Complexity Analysis

$$c_p = c \bmod p \qquad c_q = c \bmod q$$
$$m_p = c_p^{d_p} \bmod p \qquad m_q = c_q^{d_q} \bmod q$$

$$h = q_{\text{inv}}(m_p - m_q) \bmod p$$
$$m = m_q + hq \bmod n$$

Assume $\log d = \log n = B$, and $\log p = \log q = \frac{B}{2}$,
d, d_p, d_q equally many 0s and 1s

normal $\sim \frac{3}{2}B$ mult. of size B

via CRT $3 + 2 \cdot \frac{3}{2} \cdot \frac{B}{2}$ op.s of size $\frac{B}{2}$, 1 mod of size B
$\sim \frac{3}{2}B$ op.s of size $\frac{B}{2}$
factor 2-4, depending on multiplication method
can be run in parallel $\rightsquigarrow$ another factor 2

Example

- private key: $n = 4559$, $d = 3377$, $p = 47$, $q = 97$
- compute once: $d_p = 19$, $d_q = 17$, $q_{inv} = 16$

Example

- private key: $n = 4559$, $d = 3377$, $p = 47$, $q = 97$
- compute once: $d_p = 19$, $d_q = 17$, $q_{inv} = 16$
- decrypt $c = 2993$

$$c_p = 32 \qquad c_q = 83$$

$$m_p = c_p^{d_p} \bmod p = 8 \qquad m_q = c_q^{d_q} \bmod q = 5$$

$$h = q_{\text{inv}}(m_p - m_q) \bmod p = 1$$

$$m = m_q + hq \bmod n = 102$$

Note: These computations are nearly possible by hand.

Definition (Addition Chain, D. Knuth, TAOCP Vol. 2)

An addition chain for integer n of length l is a sequence

$$1 = a_0, a_1, \ldots, a_l = n$$

such that every entry is a sum of 2 previous ones.

Definition (Addition Chain, D. Knuth, TAOCP Vol. 2)

An addition chain for integer n of length l is a sequence

$$1 = a_0, a_1, \ldots, a_l = n$$

such that every entry is a sum of 2 previous ones.

Example

For 15 we have $1, 2, 3, 6, 12, 15$ of length 5.

Definition (Addition Chain, D. Knuth, TAOCP Vol. 2)

An addition chain for integer n of length l is a sequence

$$1 = a_0, a_1, \ldots, a_l = n$$

such that every entry is a sum of 2 previous ones.

Example

For 15 we have $1, 2, 3, 6, 12, 15$ of length 5.
Application: faster modular exponentiation,
square-and-multiply: $15 = 1111_2 \rightsquigarrow 7$ multiplications

Definition (Addition Chain, D. Knuth, TAOCP Vol. 2)

An addition chain for integer n of length l is a sequence

$$1 = a_0, a_1, \ldots, a_l = n$$

such that every entry is a sum of 2 previous ones.

Example

For 15 we have $1, 2, 3, 6, 12, 15$ of length 5.
Application: faster modular exponentiation,
square-and-multiply: $15 = 1111_2 \rightsquigarrow$ 7 multiplications

$$x^2 = x \cdot x \mod n \qquad x^{12} = x^6 \cdot x^6 \mod n$$
$$x^3 = x^2 \cdot x \mod n \qquad x^{15} = x^{12} \cdot x^3 \mod n$$
$$x^6 = x^3 \cdot x^3 \mod n$$

Problem with Addition Chains

Let $l(n)$ denote length of smallest addition chain for n.

Finding $l(n)$ is hard

Let $|n|_1$ denote number of 1s. Known bounds are:

$$\log n + \log |n|_1 - 2.13 \leq l(n) \leq \log n + |n|_1 - 1$$

$$l(n) \in \log n + (1 + o(1)) \cdot \frac{\log n}{\log \log n}$$

Problem with Addition Chains

Let $l(n)$ denote length of smallest addition chain for n.

Finding $l(n)$ is hard

Let $|n|_1$ denote number of 1s. Known bounds are:

$$\log n + \log |n|_1 - 2.13 \leq l(n) \leq \log n + |n|_1 - 1$$

$$l(n) \in \log n + (1 + o(1)) \cdot \frac{\log n}{\log \log n}$$

Theorem (Downey, Leong, Seth, 1981)

Given $a_1, \ldots, a_k$, find smallest chain containing them all is NP-complete.

Problem with Addition Chains

Let $l(n)$ denote length of smallest addition chain for n.

Finding $l(n)$ is hard

Let $|n|_1$ denote number of 1s. Known bounds are:

$$\log n + \log |n|_1 - 2.13 \leq l(n) \leq \log n + |n|_1 - 1$$

$$l(n) \in \log n + (1 + o(1)) \cdot \frac{\log n}{\log \log n}$$

Theorem (Downey, Leong, Seth, 1981)

Given $a_1, \ldots, a_k$, find smallest chain containing them all is NP-complete.

Exercise (Challenge)

Find a small addition chain for $2^{127} - 3$.

Parameter Size

- strength of key given in bit size of n
- `ssh-keygen` currently has default 3072
- secure key should have 4096; more threatened by Quantum Computers, than classical factoring
- p, q should have same bitlength

Parameter Size

- strength of key given in bit size of n
- `ssh-keygen` currently has default 3072
- secure key should have 4096; more threatened by Quantum Computers, than classical factoring
- p, q should have same bitlength

Standard-setting

Assume $n \sim 4096$ Bit, $e = 65537$.

- Encryption: 17 op.s of size 4096
- Decryption: $d \sim 4096$ Bit
  - Square-and-Multiply: ~ 6000 op.s of size 4096
  - optimised: $\sim 2 \times 2100$-3000 op.s of size 2048

Parameter Size

- strength of key given in bit size of n
- `ssh-keygen` currently has default 3072
- secure key should have 4096; more threatened by Quantum Computers, than classical factoring
- p, q should have same bitlength

Standard-setting

Assume $n \sim 4096$ Bit, $e = 65537$.

- Encryption: 17 op.s of size 4096
- Decryption: $d \sim 4096$ Bit
  - Square-and-Multiply: ~ 6000 op.s of size 4096
  - optimised: $\sim 2 \times 2100$-3000 op.s of size 2048

Can we swap the effort? NO! (see later)

Theorem of Secret Parameters

Theorem (Theorem of Secret Parameters)

Given one entry of the private key $(p, q, \varphi(n), d)$ and the public key, we can efficiently compute the full private key.

Theorem of Secret Parameters

Theorem (Theorem of Secret Parameters)

Given one entry of the private key $(p, q, \varphi(n), d)$ and the public key, we can efficiently compute the full private key.

Corollary

If d is known, it is not sufficient to just replace e and d.
need new primes (computational effort)

Exercise

We can also break the key, if d_p or d_q is given beside the public key.

Theorem of Secret Parameters

Theorem (Theorem of Secret Parameters)

Given one entry of the private key $(p, q, \varphi(n), d)$ and the public key, we can efficiently compute the full private key.

p, q **known** see key generation

$\varphi(n)$ **known** solve quadratic equation:

$$a := n - \varphi(n) = p + q - 1 \qquad \text{known}$$
$$n = p \cdot q = p \cdot (a + 1 - p)$$

Equation $x^2 - (a+1)x + n = 0$ has two solution: p, q

Theorem of Secret Parameters – d known

Assume d is known.

e **small:** Test $ed - 1 = * \cdot \varphi(n)$ for all $* \leq 2e$; likely $* = \left\lceil \frac{ed-1}{n} \right\rceil$

Theorem of Secret Parameters – d known

Assume d is known.

e **small:** Test $ed - 1 = * \cdot \varphi(n)$ for all $* \leq 2e$; likely $* = \left\lceil \frac{ed-1}{n} \right\rceil$

else: Note: $\gcd(*, n) \in \{1, p, q, n\}$

Theorem of Secret Parameters – d known

Assume d is known.

e **small:** Test $ed - 1 = * \cdot \varphi(n)$ for all $* \leq 2e$; likely $* = \left\lceil \frac{ed-1}{n} \right\rceil$

else: Note: $\gcd(*, n) \in \{1, p, q, n\}$

1: **function** FACTOR(d, e, n)
2: $s \leftarrow \mathsf{M}_2(ed - 1)$ ▷ multiplicity of 2
3: $k \leftarrow \frac{ed-1}{2^s}$ ▷ "odd part"

Theorem of Secret Parameters – d known

Assume d is known.

e **small:** Test $ed - 1 = * \cdot \varphi(n)$ for all $* \leq 2e$; likely $* = \left\lceil \frac{ed-1}{n} \right\rceil$

else: Note: $\gcd(*, n) \in \{1, p, q, n\}$

function FACTOR(d, e, n)
  $s \leftarrow \mathrm{M}_2(ed - 1)$ ▷ multiplicity of 2
  $k \leftarrow \frac{ed-1}{2^s}$ ▷ "odd part"
  while True **do**
    pick random $0 < a < n$
    if $\gcd(a, n) > 1$ **then** ▷ very low chance
      return gcd
    for $i = 0, \ldots, s - 1$ **do**
      if $\gcd\left((a^k)^{2^i} - 1, n\right) \notin \{1, n\}$ **then**
        return gcd

Theorem of Secret Parameters – d known

Assume d is known.

e **small:** Test $ed - 1 = * \cdot \varphi(n)$ for all $* \leq 2e$; likely $* = \left\lceil \frac{ed-1}{n} \right\rceil$

else: Note: $\gcd(*, n) \in \{1, p, q, n\}$

1: **function** FACTOR(d, e, n)
2: $s \leftarrow \mathrm{M}_2(ed - 1)$ ▷ multiplicity of 2
3: $k \leftarrow \frac{ed-1}{2^s}$ ▷ "odd part"
4: **while** True **do**
5: pick random $0 < a < n$
6: **if** $\gcd(a, n) > 1$ **then** ▷ very low chance
7: **return** gcd
8: **for** $i = 0, \ldots, s - 1$ **do**
9: **if** $\gcd\left((a^k)^{2^i} - 1, n\right) \notin \{1, n\}$ **then**
10: **return** gcd

Chance of success $\geq \frac{1}{2}$ per loop,
but the "why" is more complicatded

Notation: $ed - 1 = * \cdot \varphi(n) = k \cdot 2^s$, k odd

Interesting Code part:

for $i = 0, \ldots, s-1$ **do**

  if $\gcd\left((a^k)^{2^i} - 1, n\right) \notin \{1, n\}$ **then**

    return gcd

Do not really need d, but just some multiple of $\varphi(n)$

Notation: $ed - 1 = * \cdot \varphi(n) = k \cdot 2^s$, k odd

Interesting Code part:

for $i = 0, \ldots, s-1$ **do**
  if $\gcd\left((a^k)^{2^i} - 1, n\right) \notin \{1, n\}$ **then**
    return gcd

Do not really need d, but just some multiple of $\varphi(n)$

Proof (beginning).

$$\gcd(a, n) = 1 \implies (a^k)^{2^s} = a^{*\cdot\varphi(n)} = \left(a^{\varphi(n)}\right)^* \equiv 1 \mod n$$

Notation: $ed - 1 = * \cdot \varphi(n) = k \cdot 2^s$, k odd

Interesting Code part:

for $i = 0, \ldots, s-1$ **do**
  if $\gcd\left((a^k)^{2^i} - 1, n\right) \notin \{1, n\}$ **then**
    return gcd

Do not really need d, but just some multiple of $\varphi(n)$

Proof (beginning).

$$\gcd(a,n) = 1 \implies (a^k)^{2^s} = a^{*\cdot\varphi(n)} = \left(a^{\varphi(n)}\right)^* \equiv 1 \mod n$$

$$\implies (a^k)^{2^s} - 1 \equiv 0 \mod n$$

$$\implies \gcd\left((a^k)^{2^s} - 1, n\right) = n$$

Notation: $ed - 1 = * \cdot \varphi(n) = k \cdot 2^s$, k odd

Interesting Code part:

for $i = 0, \ldots, s-1$ **do**

  if $\gcd\left((a^k)^{2^i} - 1, n\right) \notin \{1, n\}$ **then**

    return gcd

Do not really need d, but just some multiple of $\varphi(n)$

Proof (beginning).

$$\gcd(a,n) = 1 \implies (a^k)^{2^s} = a^{*\cdot\varphi(n)} = \left(a^{\varphi(n)}\right)^* \equiv 1 \mod n$$

$$\implies (a^k)^{2^s} - 1 \equiv 0 \mod n$$

$$\implies \gcd\left((a^k)^{2^s} - 1, n\right) = n$$

Look for first step i with $\gcd\left((a^k)^{2^i} - 1, n\right) > 1$, (could be n)
but if not, the gcd is p or $q \rightsquigarrow$ know everything

Proof Idea.

- started with observation

$$(a^k)^{2^s} - 1 \equiv 0 \mod n$$

- congruence also holds modulo p, q

Proof Idea.

- started with observation

$$(a^k)^{2^s} - 1 \equiv 0 \mod n$$

- congruence also holds modulo p, q
- also possibly for smaller exponents x, y (pick smallest)

$$(a^k)^{2^x} - 1 \equiv 0 \mod p \qquad (a^k)^{2^y} - 1 \equiv 0 \mod q$$

- assume x, y differ (chance $\geq 50\%$), wlog $x < y$

$$(a^k)^{2^x} - 1 \equiv 0 \mod p \qquad (a^k)^{2^x} - 1 \not\equiv 0 \mod q$$

$$\implies \gcd\left((a^k)^{2^x} - 1, n\right) = p$$

- try out all $x \rightsquigarrow$ success

Groups

Definition

A group is a structure $\mathcal{G} = (G, \circ, *^{-1}, 1)$ such that

- $\circ$ is associative
- $\forall g \in G \,.\, 1 \circ g = g = g \circ 1$ (neutral element)
- $\forall g \in G \,.\, g \circ g^{-1} = 1 = g^{-1} \circ g$ (inverse element)

If in addition, $\circ$ is commutative, we call $\mathcal{G}$ abelian group.

Groups

Definition

A group is a structure $\mathcal{G} = (G, \circ, *^{-1}, 1)$ such that

- $\circ$ is associative
- $\forall g \in G \,.\, 1 \circ g = g = g \circ 1$ (neutral element)
- $\forall g \in G \,.\, g \circ g^{-1} = 1 = g^{-1} \circ g$ (inverse element)

If in addition, $\circ$ is commutative, we call $\mathcal{G}$ abelian group.

Often refer to G as the group, or just define $\circ$ explicitly.

Groups

Definition

A group is a structure $\mathcal{G} = (G, \circ, *^{-1}, 1)$ such that

- $\circ$ is associative
- $\forall g \in G \,.\, 1 \circ g = g = g \circ 1$ (neutral element)
- $\forall g \in G \,.\, g \circ g^{-1} = 1 = g^{-1} \circ g$ (inverse element)

If in addition, $\circ$ is commutative, we call $\mathcal{G}$ abelian group.

Often refer to G as the group, or just define $\circ$ explicitly.

Example

- $(\mathbb{Z}, +)$, $(\mathbb{Z}_n, +)$
- $(\mathbb{Z}_n^*, \cdot)$ all numbers coprime to n

Groups

Definition

A group is a structure $\mathcal{G} = (G, \circ, *^{-1}, 1)$ such that

- $\circ$ is associative
- $\forall g \in G \,.\, 1 \circ g = g = g \circ 1$ (neutral element)
- $\forall g \in G \,.\, g \circ g^{-1} = 1 = g^{-1} \circ g$ (inverse element)

If in addition, $\circ$ is commutative, we call $\mathcal{G}$ abelian group.

Often refer to G as the group, or just define $\circ$ explicitly.

Example

- $(\mathbb{Z}, +)$, $(\mathbb{Z}_n, +)$
- $(\mathbb{Z}_n^*, \cdot)$ all numbers coprime to n
- S_n: the group of permutations of n elements
- point addition on elliptic curves ($\rightsquigarrow$ later section)

Some more Algebra

Definition

Let G be a group, $g \in G$. The order of g, $o_G(g)$ or just $o(g)$, is the smallest number $k > 0$ with $g^k = 1$.

Some more Algebra

Definition

Let G be a group, $g \in G$. The order of g, $o_G(g)$ or just $o(g)$, is the smallest number $k > 0$ with $g^k = 1$.

Example

- $G = (\mathbb{Z}_7^*, \cdot)$, $g = 2$, then $2^3 = 8 = 1$, but $2^1, 2^2 \neq 1$ so $o(2) = 3$

Some more Algebra

Definition

Let G be a group, $g \in G$. The order of g, $o_G(g)$ or just $o(g)$, is the smallest number $k > 0$ with $g^k = 1$.

Example

- $G = (\mathbb{Z}_7^*, \cdot)$, $g = 2$, then $2^3 = 8 = 1$, but $2^1, 2^2 \neq 1$ so $o(2) = 3$
- $G = (\mathbb{Z}_7, +)$, $g = 2$, then we take multiples of 2 and look for 0 (neutral element), multiples are $2, 4, 6, 1, 3, 5, 0$, so $o(2) = 7$

Some more Algebra

Definition

Let G be a group, $g \in G$. The order of g, $o_G(g)$ or just $o(g)$, is the smallest number $k > 0$ with $g^k = 1$.

Example

- $G = (\mathbb{Z}_7^*, \cdot)$, $g = 2$, then $2^3 = 8 = 1$, but $2^1, 2^2 \neq 1$ so $o(2) = 3$
- $G = (\mathbb{Z}_7, +)$, $g = 2$, then we take multiples of 2 and look for 0 (neutral element), multiples are $2, 4, 6, 1, 3, 5, 0$, so $o(2) = 7$

Lemma (Properties of order)

Let G be a group, $g \in G$

- $o(g) \mid |G|$ *(element order divides group order),* $g^{|G|} = 1$
- *If* $g^n = 1$, *then* $o(g) \mid n$.

Definition

Let G be a group, $g \in G$. If $o(g) = |G|$, then G is called cyclic, and g is called generator. Equivalently: $G = \langle g \rangle = \{g^n : n \in \mathbb{Z}\}$.

Definition

Let G be a group, $g \in G$. If $o(g) = |G|$, then G is called cyclic, and g is called generator. Equivalently: $G = \langle g \rangle = \{g^n : n \in \mathbb{Z}\}$.

Example

- $\mathbb{Z} = (\mathbb{Z}, +)$: $\mathbb{Z} = \langle 1 \rangle$
- $\mathbb{G} = (\mathbb{Z}_7, +)$, $g = 2$: multiples are $2, 4, 6, 1, 3, 5, 0$, so $\mathbb{Z}_7 = \langle 2 \rangle$
- $\mathbb{G} = (\mathbb{Z}_7^*, \cdot)$: $g = 3$, powers are $3, 2, 6, 4, 5, 1$, so $\mathbb{Z}_7^* = \langle 3 \rangle$

Definition

Let G be a group, $g \in G$. If $o(g) = |G|$, then G is called cyclic, and g is called generator. Equivalently: $G = \langle g \rangle = \{g^n : n \in \mathbb{Z}\}$.

Example

- $\mathbb{Z} = (\mathbb{Z}, +)$: $\mathbb{Z} = \langle 1 \rangle$
- $\mathbb{G} = (\mathbb{Z}_7, +)$, $g = 2$: multiples are $2, 4, 6, 1, 3, 5, 0$, so $\mathbb{Z}_7 = \langle 2 \rangle$
- $\mathbb{G} = (\mathbb{Z}_7^*, \cdot)$: $g = 3$, powers are $3, 2, 6, 4, 5, 1$, so $\mathbb{Z}_7^* = \langle 3 \rangle$

Lemma

The multiplicative group of every finite field is cyclic.

In particular for prime p there is some $g < p$ such that $\mathbb{Z}_p^* = \langle g \rangle$.

Notation: $ed - 1 = x \cdot \varphi(n) = k \cdot 2^s$

Proof (cont.)

$$\mathbb{Z}_n \cong \mathbb{Z}_p \times \mathbb{Z}_q \qquad \text{CRT}$$

Notation: $ed - 1 = x \cdot \varphi(n) = k \cdot 2^s$

Proof (cont.)

$$\mathbb{Z}_n{}^* \cong \mathbb{Z}_p{}^* \times \mathbb{Z}_q{}^* \qquad \text{CRT}$$

Notation: $ed - 1 = x \cdot \varphi(n) = k \cdot 2^s$

Proof (cont.)

$$\mathbb{Z}_n{}^* \cong \mathbb{Z}_p{}^* \times \mathbb{Z}_q{}^* \qquad \text{CRT}$$

order of a^k: $\left(a^k\right)^{2^s} = 1$ in $\mathbb{Z}_n^*$,

Notation: $ed - 1 = x \cdot \varphi(n) = k \cdot 2^s$

Proof (cont.)

$$\mathbb{Z}_n{}^* \cong \mathbb{Z}_p{}^* \times \mathbb{Z}_q{}^* \qquad \text{CRT}$$

order of a^k: $\left(a^k\right)^{2^s} = 1$ in $\mathbb{Z}_n^*$, also in $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$, so $o\left(a^k\right) \mid 2^s$

Notation: $ed - 1 = x \cdot \varphi(n) = k \cdot 2^s$

Proof (cont.)

$$\mathbb{Z}_n{}^* \cong \mathbb{Z}_p{}^* \times \mathbb{Z}_q{}^* \qquad \text{CRT}$$

order of a^k: $\left(a^k\right)^{2^s} = 1$ in $\mathbb{Z}_n^*$, also in $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$, so $o\left(a^k\right) \mid 2^s$

$$\Longrightarrow \quad o_{\mathbb{Z}_p^*}\left(a^k\right) = 2^{l_1} \qquad o_{\mathbb{Z}_q^*}\left(a^k\right) = 2^{l_2} \qquad \text{for some } l_1, l_2 \leq s$$

Notation: $ed - 1 = x \cdot \varphi(n) = k \cdot 2^s$

Proof (cont.)

$$\mathbb{Z}_n{}^* \cong \mathbb{Z}_p{}^* \times \mathbb{Z}_q{}^* \qquad \text{CRT}$$

order of a^k: $\left(a^k\right)^{2^s} = 1$ in $\mathbb{Z}_n^*$, also in $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$, so $o\left(a^k\right) \mid 2^s$

$$\Longrightarrow \quad o_{\mathbb{Z}_p^*}\left(a^k\right) = 2^{l_1} \qquad o_{\mathbb{Z}_q^*}\left(a^k\right) = 2^{l_2} \qquad \text{for some } l_1, l_2 \leq s$$

Let g, h be generators of $\mathbb{Z}_p^*, \mathbb{Z}_q^*$ (i.e. $\mathbb{Z}_p^* = \langle g \rangle = \{g^n : n \in \mathbb{N}\}$)
random $a \rightsquigarrow a^k \cong (g^y, h^z)$ (a bit random) in $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$ for some y, z

Notation: $ed - 1 = x \cdot \varphi(n) = k \cdot 2^s$

Proof (cont.)

$$\mathbb{Z}_n{}^* \cong \mathbb{Z}_p{}^* \times \mathbb{Z}_q{}^* \qquad \text{CRT}$$

order of a^k: $\left(a^k\right)^{2^s} = 1$ in $\mathbb{Z}_n^*$, also in $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$, so $o\left(a^k\right) \mid 2^s$

$$\Longrightarrow \quad o_{\mathbb{Z}_p^*}\left(a^k\right) = 2^{l_1} \qquad o_{\mathbb{Z}_q^*}\left(a^k\right) = 2^{l_2} \qquad \text{for some } l_1, l_2 \leq s$$

Let g, h be generators of $\mathbb{Z}_p^*, \mathbb{Z}_q^*$ (i.e. $\mathbb{Z}_p^* = \langle g \rangle = \{g^n : n \in \mathbb{N}\}$)
random $a \rightsquigarrow a^k \cong (g^y, h^z)$ (a bit random) in $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$ for some y, z
$o\left(g^y\right) = 2^{l_1}, o\left(h^z\right) = 2^{l_2}$ are 2-powers; if different $\rightsquigarrow$ success

Notation: $ed - 1 = x \cdot \varphi(n) = k \cdot 2^s$

Proof (cont.)

$$\mathbb{Z}_n{}^* \cong \mathbb{Z}_p{}^* \times \mathbb{Z}_q{}^* \qquad \text{CRT}$$

order of a^k: $\left(a^k\right)^{2^s} = 1$ in $\mathbb{Z}_n^*$, also in $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$, so $o\left(a^k\right) \mid 2^s$

$$\implies o_{\mathbb{Z}_p^*}\left(a^k\right) = 2^{l_1} \qquad o_{\mathbb{Z}_q^*}\left(a^k\right) = 2^{l_2} \qquad \text{for some } l_1, l_2 \leq s$$

Let g, h be generators of $\mathbb{Z}_p^*, \mathbb{Z}_q^*$ (i.e. $\mathbb{Z}_p^* = \langle g \rangle = \{g^n : n \in \mathbb{N}\}$)
random $a \rightsquigarrow a^k \cong (g^y, h^z)$ (a bit random) in $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$ for some y, z
$o\left(g^y\right) = 2^{l_1}, o\left(h^z\right) = 2^{l_2}$ are 2-powers; if different $\rightsquigarrow$ success
Assume wlog $l_1 < l_2$, then

$$\left(a^k\right)^{2^{l_1}} \equiv 1 \mod p \qquad \left(a^k\right)^{2^{l_1}} \not\equiv 1 \mod q$$

Notation: $ed - 1 = x \cdot \varphi(n) = k \cdot 2^s$

Proof (cont.)

$$\mathbb{Z}_n{}^* \cong \mathbb{Z}_p{}^* \times \mathbb{Z}_q{}^* \qquad \text{CRT}$$

order of a^k: $\left(a^k\right)^{2^s} = 1$ in $\mathbb{Z}_n^*$, also in $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$, so $o\left(a^k\right) \mid 2^s$

$$\implies o_{\mathbb{Z}_p^*}\left(a^k\right) = 2^{l_1} \qquad o_{\mathbb{Z}_q^*}\left(a^k\right) = 2^{l_2} \qquad \text{for some } l_1, l_2 \leq s$$

Let g, h be generators of $\mathbb{Z}_p^*, \mathbb{Z}_q^*$ (i.e. $\mathbb{Z}_p^* = \langle g \rangle = \{g^n : n \in \mathbb{N}\}$)
random $a \rightsquigarrow a^k \cong (g^y, h^z)$ (a bit random) in $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$ for some y, z
$o\left(g^y\right) = 2^{l_1}, o\left(h^z\right) = 2^{l_2}$ are 2-powers; if different $\rightsquigarrow$ success
Assume wlog $l_1 < l_2$, then

$$\left(a^k\right)^{2^{l_1}} \equiv 1 \mod p \qquad \left(a^k\right)^{2^{l_1}} \not\equiv 1 \mod q$$

$$p \mid \left(a^k\right)^{2^{l_1}} - 1 \qquad q \nmid \left(a^k\right)^{2^{l_1}} - 1 \qquad \implies \gcd\left(\left(a^k\right)^{2^{l_1}} - 1, n\right) = p$$

Proof (cont.)

Recall $a^k \cong (g^y, h^z) \in \mathbb{Z}_p^* \times \mathbb{Z}_q^*$

$\mathrm{M}_2\left(|\mathbb{Z}_p^*|\right)$: How much squaring is irreversible in $\mathbb{Z}_p^*$?

$\mathrm{M}_2(y)$: How much squaring did we already do?

l_1: How much squaring do we still have to do?

Proof (cont.)

Recall $a^k \cong (g^y, h^z) \in \mathbb{Z}_p^* \times \mathbb{Z}_q^*$

$\mathrm{M}_2\left(|\mathbb{Z}_p^*|\right)$: How much squaring is irreversible in $\mathbb{Z}_p^*$?

$\mathrm{M}_2(y)$: How much squaring did we already do?

l_1: How much squaring do we still have to do?

$$\mathrm{M}_2\left(|\mathbb{Z}_p^*|\right) = \mathrm{M}_2(p-1) = \mathrm{M}_2(y) + l_1$$

Proof (cont.)

Recall $a^k \cong (g^y, h^z) \in \mathbb{Z}_p^* \times \mathbb{Z}_q^*$

$\mathrm{M}_2\left(|\mathbb{Z}_p^*|\right)$: How much squaring is irreversible in $\mathbb{Z}_p^*$?

$\mathrm{M}_2(y)$: How much squaring did we already do?

l_1: How much squaring do we still have to do?

$$\mathrm{M}_2\left(|\mathbb{Z}_p^*|\right) = \mathrm{M}_2(p-1) = \mathrm{M}_2(y) + l_1$$

Case distinction:

$\mathrm{M}_2(p-1) < \mathrm{M}_2(q-1)$: if z odd, then $\mathrm{M}_2(z) = 0$, so

$$l_1 \leq \mathrm{M}_2(p-1) < \mathrm{M}_2(q-1) = l_2$$

Chance $\geq 50\%$ (works at least if z odd)

Proof (cont.)

Recall $a^k \cong (g^y, h^z) \in \mathbb{Z}_p^* \times \mathbb{Z}_q^*$

$\mathrm{M}_2\left(|\mathbb{Z}_p^*|\right)$: How much squaring is irreversible in $\mathbb{Z}_p^*$?

$\mathrm{M}_2(y)$: How much squaring did we already do?

l_1: How much squaring do we still have to do?

$$\mathrm{M}_2\left(|\mathbb{Z}_p^*|\right) = \mathrm{M}_2(p-1) = \mathrm{M}_2(y) + l_1$$

Case distinction:

$\mathrm{M}_2(p-1) < \mathrm{M}_2(q-1)$: if z odd, then $\mathrm{M}_2(z) = 0$, so

$$l_1 \leq \mathrm{M}_2(p-1) < \mathrm{M}_2(q-1) = l_2$$

Chance $\geq 50\%$ (works at least if z odd)

$\mathrm{M}_2(p-1) = \mathrm{M}_2(q-1)$: if y, z different parity (odd/even), then $l_1 \neq l_2$; again 50% chance

Calm Down

Recap

What we did so far:

- public key (n, e)
- private key (n, d)
- every entry $p, q, d, \varphi(n)$ allows to compute all others

Recap

What we did so far:

- public key (n, e)
- private key (n, d)
- every entry $p, q, d, \varphi(n)$ allows to compute all others

Next Steps:

- goal: find original plaintext in other words: focus on OW-CPA or OW-CCA
- exploit properties of RSA

RSA-Problem

Definition (RSA-Problem)

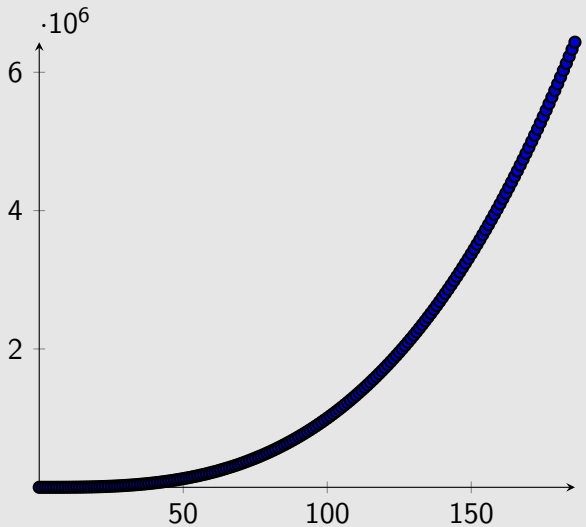
Given $n, e, m^e \bmod n$, find m (i.e. the e-th modular root).

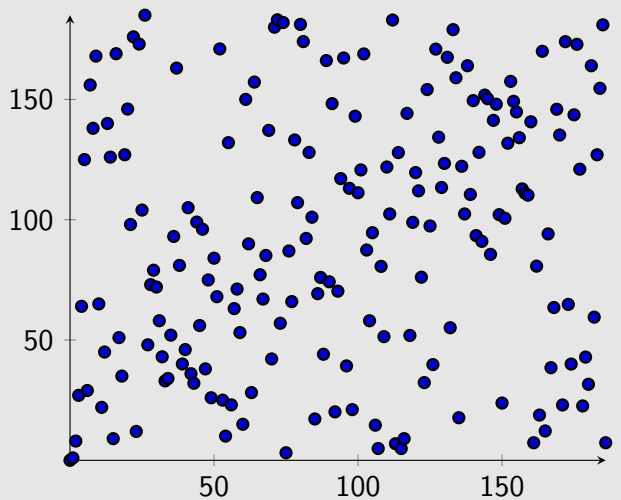
RSA-Problem

Definition (RSA-Problem)

Given $n, e, m^e \bmod n$, find m (i.e. the e-th modular root).

Example (x^3 without and with modulo 187)



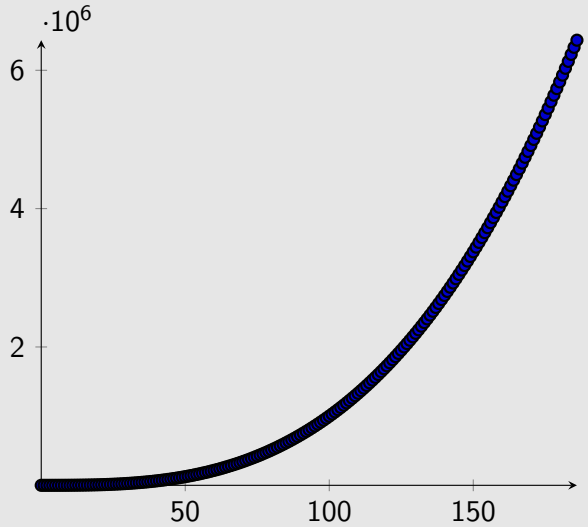


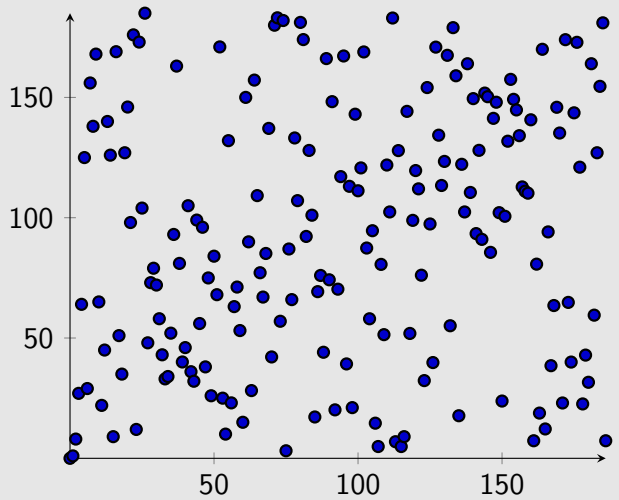
RSA-Problem

Definition (RSA-Problem)

Given $n, e, m^e \bmod n$, find m (i.e. the e-th modular root).

Example (x^3 without and with modulo 187)





Difference

In $\mathbb{Z}$ we have an order and monotonicity, allows e.g. bisection.

Connection of Problems

Clearly, we have the reduction

$$\text{RSA-problem} \leq_p \text{finding } d \leq_p \text{Factoring}$$

Connection of Problems

Clearly, we have the reduction

$$\text{RSA-problem} \leq_p \text{finding } d \leq_p \text{Factoring}$$

But we just showed:

$$\text{find } d \equiv_{BPP} \text{Factoring}$$

i.e. equivalent in "bounded error probability polynomial time".

Connection of Problems

Clearly, we have the reduction

$$\text{RSA-problem} \leq_p \text{finding } d \leq_p \text{Factoring}$$

But we just showed:

$$\text{find } d \equiv_{BPP} \text{Factoring}$$

i.e. equivalent in "bounded error probability polynomial time".

Theorem (Coron, May, 2004)

Using Coppersmith (see later): finding $d \equiv_p$ Factoring

i.e. they are equivalent in deterministic polynomial time.

Connection of Problems

Clearly, we have the reduction

$$\text{RSA-problem} \leq_p \text{finding } d \leq_p \text{Factoring}$$

But we just showed:

$$\text{find } d \equiv_{BPP} \text{Factoring}$$

i.e. equivalent in "bounded error probability polynomial time".

Theorem (Coron, May, 2004)

Using Coppersmith (see later): finding $d \equiv_p$ Factoring

i.e. they are equivalent in deterministic polynomial time. For the first reduction, the converse is open.

$$\text{finding } d \overset{?}{\leq_p} \text{RSA-problem}$$

Insecure Special Cases

What can go wrong?

In general, RSA is secure but take care if:

e **is small:** several attacks, find m

same n**, but different** e**:** find m

m **is very small:** find m

d **is small:** find d, thus everything

fault in prime generation: factor n, thus everything

Insecure Special Cases

What can go wrong?

In general, RSA is secure but take care if:

e **is small:** several attacks, find m

same n**, but different** e**:** find m

m **is very small:** find m

d **is small:** find d, thus everything

fault in prime generation: factor n, thus everything

There is a standard, avoiding all/most of these.

Public-Key Cryptography Standard (PKCS)

PKCS #1 covers RSA, currently in version 2.2

`https://tools.ietf.org/html/rfc8017a`

Small Public Exponent e

Scenario: Hybrid Encryption

Use asymmetric crypto to exchange key, then use (faster) symmetric encryption
send AES key (128/256 Bit) via RSA (4096 Bit)

Small Public Exponent e

Scenario: Hybrid Encryption

Use asymmetric crypto to exchange key, then use (faster) symmetric encryption
send AES key (128/256 Bit) via RSA (4096 Bit)

Toy Example 0

- $m < \sqrt[e]{n}$, so $m^e < n \implies m^e \bmod n = m^e$
- don't compute modulo
- Decrypt: $m = \sqrt[e]{c}$ in $\mathbb{Z}$ (e.g. bisection, no floating point!)

Small Public Exponent e

Scenario: Hybrid Encryption

Use asymmetric crypto to exchange key, then use (faster) symmetric encryption
send AES key (128/256 Bit) via RSA (4096 Bit)

Toy Example 0

- $m < \sqrt[e]{n}$, so $m^e < n \implies m^e \bmod n = m^e$
- don't compute modulo
- Decrypt: $m = \sqrt[e]{c}$ in $\mathbb{Z}$ (e.g. bisection, no floating point!)

Example

Key $(10\,720\,441, 3)$, i.e. n has 24 Bit, message $m = 102$

$$m^3 \bmod n = 1\,061\,208 \bmod 10\,720\,441 = 1\,061\,208$$

Padding

Padding

Artificially enlarge message to $m' = \mathrm{Pad}(m)$, such that $m' \approx n$.

Padding

Padding

Artificially enlarge message to $m' = \text{Pad}(m)$, such that $m' \approx n$.

Toy Example 1: multiply with fixed, known number

Encrypt: $m' := m \cdot r$ for fixed $r \in \mathbb{Z}_n^*$; $c = (m')^e \mod n$
Decrypt: $m = r^{-1} \cdot (c)^d \mod n$

Padding

Padding

Artificially enlarge message to $m' = \mathrm{Pad}(m)$, such that $m' \approx n$.

Toy Example 1: multiply with fixed, known number

Encrypt: $m' := m \cdot r$ for fixed $r \in \mathbb{Z}_n^*$; $c = (m')^e \mod n$
Decrypt: $m = r^{-1} \cdot (c)^d \mod n$

Break

RSA is multiplicative

If we know $\mathrm{enc}(m)$, then we also know $\mathrm{enc}(x \cdot m)$ for every x.

Padding

Padding

Artificially enlarge message to $m' = \mathrm{Pad}(m)$, such that $m' \approx n$.

Toy Example 1: multiply with fixed, known number

Encrypt: $m' := m \cdot r$ for fixed $r \in \mathbb{Z}_n^*$; $c = (m')^e \mod n$
Decrypt: $m = r^{-1} \cdot (c)^d \mod n$

Break

RSA is multiplicative

If we know $\mathrm{enc}(m)$, then we also know $\mathrm{enc}(x \cdot m)$ for every x.
Reduce to previous case, put

$$c' := \left(r^{-1}\right)^e \cdot c = \left(r^{-1}\right)^e \cdot (m')^e = \left(r^{-1} m'\right)^e = \left(r^{-1} r m\right)^e = m^e$$

So we know m^e, if m small, decrypt as before

Padding

Toy Example 2: Fill with 0s

Putting $m' = m \,||\, 0 \ldots 0$

Padding

Toy Example 2: Fill with 0s

Putting $m' = m \,||\, 0 \ldots 0$ is just $m' = m \cdot 2^k$, so same as version 1.

Padding

Toy Example 2: Fill with 0s

Putting $m' = m \,||\, 0 \ldots 0$ is just $m' = m \cdot 2^k$, so same as version 1.

Toy Example 3: Concatenate

concatenate m with itself: $m' = m \,||\, \ldots \,||\, m$

Padding

Toy Example 2: Fill with 0s

Putting $m' = m \,||\, 0 \ldots 0$ is just $m' = m \cdot 2^k$, so same as version 1.

Toy Example 3: Concatenate

concatenate m with itself: $m' = m \,||\, \ldots \,||\, m$
But mathematically, that is just

$$m' = m \cdot 1 \underbrace{0 \ldots 01}_{\lceil \log m \rceil} 0 \ldots 01 \ldots 01_2$$

- guess length of m: $\log m < \log n$, i.e. small, we can test all
- we know $10 \ldots 010 \ldots 01 \ldots 01_2$ (i.e. in binary)
- break like Version 1

Coppersmith

Theorem (Coppersmith, 1996)

Let $f \in \mathbb{Z}[x]$ normalised, $e = \deg f$. Then we can compute all $x_0 \in \mathbb{Z}$ with $f(x_0) \equiv 0 \mod n$ and $|x_0| \leq \sqrt[e]{n}$ in polynomial time.

Coppersmith

Theorem (Coppersmith, 1996)

Let $f \in \mathbb{Z}[x]$ normalised, $e = \deg f$. Then we can compute all $x_0 \in \mathbb{Z}$ with $f(x_0) \equiv 0 \mod n$ and $|x_0| \leq \sqrt[e]{n}$ in polynomial time.

polynomial time $\neq$ efficient

Coppersmith

Theorem (Coppersmith, 1996)

Let $f \in \mathbb{Z}[x]$ normalised, $e = \deg f$. Then we can compute all $x_0 \in \mathbb{Z}$ with $f(x_0) \equiv 0 \mod n$ and $|x_0| \leq \sqrt[e]{n}$ in polynomial time.

polynomial time $\neq$ efficient

Theorem (from Nina Jekel, Bsc-thesis, 2017)

Let $f \in \mathbb{Z}[x]$ normalised, $\deg f = e$. Assume we have an upper bound for our roots

$$X \leq \frac{1}{2} n^{\frac{1}{e} - \varepsilon}$$

for some $\varepsilon > 0$. Then the running time of Coppersmith is in

$$\mathcal{O}\left(\frac{e^9}{\varepsilon^5} \log n\right)$$

Coppersmith

Proof idea.

Transform into lattice problem, apply LLL-algorithm to reduce base
Way(!) too involved for this course. □

Coppersmith

Proof idea.

Transform into lattice problem, apply LLL-algorithm to reduce base
Way(!) too involved for this course. □

Corollary (Application in RSA)

If m has (significantly) fewer than $\log(n)/e$ bits, and we have any fixed padding, we can compute m.

Coppersmith

Proof idea.

Transform into lattice problem, apply LLL-algorithm to reduce base
Way(!) too involved for this course. □

Corollary (Application in RSA)

If m has (significantly) fewer than $\log(n)/e$ bits, and we have any fixed padding, we can compute m.

In Sagemath implemented as `f.small_roots()`, (but has issues)
Alternatively: CTF-writeup from github

> *If you want an implementation of a crypto algorithm, write a crypto CTF challenge that needs it and read writeups.*
>
> *(ubuntor)*

Coppersmith – Application

Example (PWN-CTF 2018, Whistle)

Padding PKCS#1 v1.5 (RFC 2313, Nov 1993),
but applied padding for private-key-operation (i.e. for $m^d \bmod n$):

$$m' = 00 \,||\, 01 \,||\, FF \ldots FF \,||\, 00 \,||\, m$$

Coppersmith – Application

Example (PWN-CTF 2018, Whistle)

Padding PKCS#1 v1.5 (RFC 2313, Nov 1993),
but applied padding for private-key-operation (i.e. for m^d mod n):

$$m' = 00 \,||\, 01 \,||\, FF \ldots FF \,||\, 00 \,||\, m$$

$n \sim 4096$ bit, $e = 3$, $m \sim 128$ bit AES key, so padding is

$$\underbrace{0 \ldots 0}_{15} \underbrace{1 \ldots\ldots 1}_{\text{rest}} \underbrace{0 \ldots 0}_{8} \underbrace{0 \ldots 0}_{128}$$

Coppersmith – Application

Example (PWN-CTF 2018, Whistle)

Padding PKCS#1 v1.5 (RFC 2313, Nov 1993),
but applied padding for private-key-operation (i.e. for m^d mod n):

$$m' = 00 \,||\, 01 \,||\, FF \ldots FF \,||\, 00 \,||\, m$$

$n \sim 4096$ bit, $e = 3$, $m \sim 128$ bit AES key, so padding is

$$\underbrace{0\ldots0}_{15}\underbrace{1\ldots\ldots1}_{\text{rest}}\underbrace{0\ldots0}_{8}\underbrace{0\ldots0}_{128}$$

Counting and adding correct 2-power:

$$f(x) = (2^{4081} - 2^{8+128} + x)^3 - c$$

Coppersmith – Application

Example (PWN-CTF 2018, Whistle)

Padding PKCS#1 v1.5 (RFC 2313, Nov 1993),
but applied padding for private-key-operation (i.e. for $m^d \bmod n$):

$$m' = 00 \,||\, 01 \,||\, FF \ldots FF \,||\, 00 \,||\, m$$

$n \sim 4096$ bit, $e = 3$, $m \sim 128$ bit AES key, so padding is

$$\underbrace{0\ldots0}_{15}\underbrace{1\ldots\ldots1}_{\text{rest}}\underbrace{0\ldots0}_{8}\underbrace{0\ldots0}_{128}$$

Counting and adding correct 2-power:

$$f(x) = (2^{4081} - 2^{8+128} + x)^3 - c$$

m has much fewer than $\log(n)/e \approx 1365$ bits $\rightsquigarrow$ Coppersmith finds m

Coppersmith Failure

Example (NSUCrypto 2019, Problem 3)

We know p, q have 500 bits. Given $n = pq$ and

$$h = 3^{2019}p^2 + 5^{2019}q^2 \mod \underbrace{n^2 + 8 \cdot 2019}_{=:N}$$

Find p, q.

Coppersmith Failure

Example (NSUCrypto 2019, Problem 3)

We know p, q have 500 bits. Given $n = pq$ and

$$h = 3^{2019}p^2 + 5^{2019}q^2 \mod \underbrace{n^2 + 8 \cdot 2019}_{=:N}$$

Find p, q.
Multiply with p^2, use $n^2 = p^2q^2$ and rewrite into

$$0 \equiv p^4 - \left(h \cdot \left(3^{2019}\right)^{-1}\right) p^2 + 5^{2019} \cdot \left(3^{2019}\right)^{-1} n^2 \mod N$$

If we assume $p < q$, then $p^2 < n$, so $p^4 < n^2 < N$.

Coppersmith Failure

Example (NSUCrypto 2019, Problem 3)

We know p, q have 500 bits. Given $n = pq$ and

$$h = 3^{2019}p^2 + 5^{2019}q^2 \mod \underbrace{n^2 + 8 \cdot 2019}_{=:N}$$

Find p, q.
Multiply with p^2, use $n^2 = p^2q^2$ and rewrite into

$$0 \equiv p^4 - \left(h \cdot \left(3^{2019}\right)^{-1}\right) p^2 + 5^{2019} \cdot \left(3^{2019}\right)^{-1} n^2 \mod N$$

If we assume $p < q$, then $p^2 < n$, so $p^4 < n^2 < N$.
But ε too small $\rightsquigarrow$ takes too long.
Likewise if $q < p$.

Håstad Broadcast

Lemma

If a message is encrypted with the same exponent e but e different moduli n_i, we can recover the message.

Håstad Broadcast

Lemma

If a message is encrypted with the same exponent e but e different moduli n_i, we can recover the message.

Scenario: Send invitation to an event.

Håstad Broadcast

Lemma

If a message is encrypted with the same exponent e but e different moduli n_i, we can recover the message.

Scenario: Send invitation to an event.

Proof.

If some $\gcd(n_i, n_j) \neq 1$, we found a prime factor. ✓
So wlog system of congruences with coprime n_i

$$c_i \equiv m^e \mod n_i \qquad i = 1, \ldots, e$$

Håstad Broadcast

Lemma

If a message is encrypted with the same exponent e but e different moduli n_i, we can recover the message.

Scenario: Send invitation to an event.

Proof.

If some $\gcd(n_i, n_j) \neq 1$, we found a prime factor. ✓
So wlog system of congruences with coprime n_i

$$c_i \equiv m^e \mod n_i \qquad\qquad i = 1, \ldots, e$$

Put $x = m^e$ and solve via CRT. Unique solution $m^e \bmod \prod n_i$

$$m < n_i \implies m^e < \prod n_i \implies m^e \bmod \prod n_i = m^e$$

Håstad Broadcast

Lemma

If a message is encrypted with the same exponent e but e different moduli n_i, we can recover the message.

Scenario: Send invitation to an event.

Proof.

If some $\gcd(n_i, n_j) \neq 1$, we found a prime factor. ✓
So wlog system of congruences with coprime n_i

$$c_i \equiv m^e \mod n_i \qquad\qquad i = 1, \ldots, e$$

Put $x = m^e$ and solve via CRT. Unique solution $m^e \bmod \prod n_i$

$$m < n_i \implies m^e < \prod n_i \implies m^e \bmod \prod n_i = m^e$$

then just compute root in $\mathbb{Z}$ (as before). □

Example

Same message m, $e = 3$, $c_i = m^e \bmod n_i$:

$$c_1 = 533 \qquad n_1 = 551$$
$$c_2 = 333 \qquad n_2 = 943$$
$$c_3 = 357 \qquad n_3 = 527$$

Example

Same message m, $e = 3$, $c_i = m^e \bmod n_i$:

$$c_1 = 533 \qquad n_1 = 551$$
$$c_2 = 333 \qquad n_2 = 943$$
$$c_3 = 357 \qquad n_3 = 527$$

CRT yields

$$\begin{aligned} m^3 &\equiv 1061208 \mod 273825511 \\ \implies m^3 &= 1061208 \\ \implies m &= 102 \end{aligned}$$

General Håstad Broadcast

Theorem

Let n_i be coprime. Assume we modify some base message via $m_i = f_i(m)$ for $i = 1, \ldots, k$ for known polynomials f_i. If

$$k \geq e \cdot \max\{\deg f_i : i = 1, \ldots, k\},$$

then we can recover m from the f_i and $c_i = m_i^e \bmod n_i$.

General Håstad Broadcast

Theorem

Let n_i be coprime. Assume we modify some base message via $m_i = f_i(m)$ for $i = 1, \ldots, k$ for known polynomials f_i. If

$$k \geq e \cdot \max\{\deg f_i : i = 1, \ldots, k\},$$

then we can recover m from the f_i and $c_i = m_i^e \bmod n_i$.

- Special case: $f_i = \mathrm{id}$ is original Håstad.
- Typical padding $f_i(m) = 2^* \cdot m + *$ or $f(m) = 2^* \cdot * + m$ so often $\deg f_i = 1$

General Håstad Broadcast

Theorem

Let n_i be coprime. Assume we modify some base message via $m_i = f_i(m)$ for $i = 1, \ldots, k$ for known polynomials f_i. If

$$k \geq e \cdot \max\{\deg f_i : i = 1, \ldots, k\},$$

then we can recover m from the f_i and $c_i = m_i^e \bmod n_i$.

- Special case: $f_i = \mathrm{id}$ is original Håstad.
- Typical padding $f_i(m) = 2^* \cdot m + *$ or $f(m) = 2^* \cdot * + m$ so often $\deg f_i = 1$

Corollary

Any fixed padding scheme becomes dangerous, given enough messages. Use randomised padding.

Proof.

- put $g_i(x) = f_i(x)^e - c_i$, so all $g_i(m) \equiv 0 \mod n_i$,
  Note: $\deg g_i = e \cdot \deg f_i \leq k$

Proof.

- put $g_i(x) = f_i(x)^e - c_i$, so all $g_i(m) \equiv 0 \mod n_i$,
  Note: $\deg g_i = e \cdot \deg f_i \leq k$
- with CRT compute T_i with $T_i \equiv 1 \mod n_i$ and $T_i \equiv 0 \mod n_j$ for $i \neq j$ and put

$$g(x) := \sum_{i=1}^{k} T_i \cdot g_i(x)$$

  adding degree $\leq k$, so $\deg(g) \leq k$

Proof.

- put $g_i(x) = f_i(x)^e - c_i$, so all $g_i(m) \equiv 0 \mod n_i$,
  Note: $\deg g_i = e \cdot \deg f_i \leq k$
- with CRT compute T_i with $T_i \equiv 1 \mod n_i$ and $T_i \equiv 0 \mod n_j$ for $i \neq j$ and put

$$g(x) := \sum_{i=1}^{k} T_i \cdot g_i(x)$$

  adding degree $\leq k$, so $\deg(g) \leq k$
- $g(m) \equiv 0 \mod n_i$ for all i:
  - summands $j \neq i$ vanish because of T_j
  - summand i because of definition of g_i

Proof.

- put $g_i(x) = f_i(x)^e - c_i$, so all $g_i(m) \equiv 0 \mod n_i$,
  Note: $\deg g_i = e \cdot \deg f_i \leq k$
- with CRT compute T_i with $T_i \equiv 1 \mod n_i$ and $T_i \equiv 0 \mod n_j$ for $i \neq j$ and put

$$g(x) := \sum_{i=1}^{k} T_i \cdot g_i(x)$$

  adding degree $\leq k$, so $\deg(g) \leq k$
- $g(m) \equiv 0 \mod n_i$ for all i:
  - summands $j \neq i$ vanish because of T_j
  - summand i because of definition of g_i
- by CRT $g(m) \equiv 0 \mod \prod n_i$

$$m < \min_i n_i < \left(\prod n_i\right)^{\frac{1}{k}} \leq \left(\prod n_i\right)^{\frac{1}{\deg g}}$$

- so we find m via Coppersmith

Polynomial Rings

Polynomials

A (univariate) polynomial is an expression of the form

$$f = \sum_{k=0}^{D} a_k x^k$$

We can add/subtract/multiply (as long as can do so with the a_k).
$\rightsquigarrow$ polynomials form a ring.
Coefficients $a_k \in R$, the ring of polynomials (in x) is denoted $R[x]$.

Polynomial Rings

Polynomials

A (univariate) polynomial is an expression of the form

$$f = \sum_{k=0}^{D} a_k x^k$$

We can add/subtract/multiply (as long as can do so with the a_k).
$\rightsquigarrow$ polynomials form a ring.
Coefficients $a_k \in R$, the ring of polynomials (in x) is denoted $R[x]$.

Special Properties of $\mathbb{C}[x]$

- We can do polynomial division (with remainder).
- Every polynomial splits into linear factors (Vieta/Viète).

Polynomial Rings

Polynomials

A (univariate) polynomial is an expression of the form

$$f = \sum_{k=0}^{D} a_k x^k$$

We can add/subtract/multiply (as long as can do so with the a_k).
$\rightsquigarrow$ polynomials form a ring.
Coefficients $a_k \in R$, the ring of polynomials (in x) is denoted $R[x]$.

Special Properties of $\mathbb{C}[x]$

- We can do polynomial division (with remainder).
- Every polynomial splits into linear factors (Vieta/Viète).

Don't want Complex Numbers $\rightsquigarrow$ What holds over $\mathbb{Z}_n$?

Polynomial Division

Let $g, h \in \mathbb{Z}_n[x]$ given as

$$g = \sum_{k=0}^{D_1} g_k x^k \qquad h = \sum_{k=0}^{D_2} h_k x^k$$

with $D_1 \geq D_2$.

Polynomial Division

Let $g, h \in \mathbb{Z}_n[x]$ given as

$$g = \sum_{k=0}^{D_1} g_k x^k \qquad h = \sum_{k=0}^{D_2} h_k x^k$$

with $D_1 \geq D_2$.
Then first step of division is

$$g = \left(g_{D_1} \cdot h_{D_2}^{-1} \cdot x^{D_1 - D_2} \right) \cdot h + \text{Rem}_1$$

where $\deg(\text{Rem}_1) < \deg g$.

Polynomial Division

Let $g, h \in \mathbb{Z}_n[x]$ given as

$$g = \sum_{k=0}^{D_1} g_k x^k \qquad\qquad h = \sum_{k=0}^{D_2} h_k x^k$$

with $D_1 \geq D_2$.
Then first step of division is

$$g = \left(g_{D_1} \cdot h_{D_2}^{-1} \cdot x^{D_1 - D_2} \right) \cdot h + \mathrm{Rem}_1$$

where $\deg(\mathrm{Rem}_1) < \deg g$. Continue with Rem_1 and h.

Polynomial Division

Let $g, h \in \mathbb{Z}_n[x]$ given as

$$g = \sum_{k=0}^{D_1} g_k x^k \qquad h = \sum_{k=0}^{D_2} h_k x^k$$

with $D_1 \geq D_2$.
Then first step of division is

$$g = \left(g_{D_1} \cdot h_{D_2}^{-1} \cdot x^{D_1 - D_2} \right) \cdot h + \text{Rem}_1$$

where $\deg(\text{Rem}_1) < \deg g$. Continue with Rem_1 and h.

We always just multiply with inverse of leading coefficient of h.

As long as this exists, we can perform polynomial division.

In $\mathbb{Z}_n[x]$ we mostly can do division with remainder
(behaves like Euclidean Ring)

- for polynomial division, we must divide by coefficients
- i.e. must be able to invert elements
- if it fails, we found a divisor of n
- have solved the problem otherwise

In $\mathbb{Z}_n[x]$ we mostly can do division with remainder
(behaves like Euclidean Ring)

- for polynomial division, we must divide by coefficients
- i.e. must be able to invert elements
- if it fails, we found a divisor of n
- have solved the problem otherwise

Euclidean Algorithm for polynomials

- (Extended) Euclidean Algorithm works for polynomials
- we can compute the gcd of two polynomials

In $\mathbb{Z}_n[x]$ we mostly can do division with remainder
(behaves like Euclidean Ring)

- for polynomial division, we must divide by coefficients
- i.e. must be able to invert elements
- if it fails, we found a divisor of n
- have solved the problem otherwise

Euclidean Algorithm for polynomials

- (Extended) Euclidean Algorithm works for polynomials
- we can compute the gcd of two polynomials

Linear Factors

Let $f \in \mathbb{Z}_n[x]$ and $f(x_0) = 0$. Then there is $g \in \mathbb{Z}_n[x]$ with $f = (x - x_0)g$, which we can compute via polynomial division.

Franklin-Reiter-Related-Message-Attack

Theorem

If two messages are related via $m_2 = f(m_1)$ for some known polynomial f, we often can recover them from $c_i = m_i^e \bmod n$. The time is $\mathcal{O}\left((e \cdot \deg f)^2\right)$ arithmetic operations.
If f is linear and $e = 3$ the attack is guaranteed to work.

Franklin-Reiter-Related-Message-Attack

Theorem

If two messages are related via $m_2 = f(m_1)$ for some known polynomial f, we often can recover them from $c_i = m_i^e \bmod n$. The time is $\mathcal{O}\left((e \cdot \deg f)^2\right)$ arithmetic operations.
If f is linear and $e = 3$ the attack is guaranteed to work.

Proof.

Define polynomials

$$g(x) = x^e - c_1 \qquad h(x) = f(x)^e - c_2$$
$$\implies g(m_1) = h(m_1) = 0$$
$$\implies (x - m_1) \mid \gcd(g, h)$$

Mostly gcd is linear, if $e = 3$ and $\deg f = 1$, this is guaranteed. □

Example

- message: "Diary entry ??: Today I investigated [secret stuff]."
- ?? are consecutive numbers,
- assume only last digit changed: $f(x) = x + 2^{37 \cdot 8}$ (count bytes)

$$g(x) = x^3 - c_1 \qquad h(x) = (x + 2^{296})^3 - c_2$$

Example

- message: "Diary entry ??: Today I investigated [secret stuff]."
- ?? are consecutive numbers,
- assume only last digit changed: $f(x) = x + 2^{37 \cdot 8}$ (count bytes)

$$g(x) = x^3 - c_1 \qquad h(x) = (x + 2^{296})^3 - c_2$$

Calling Euclidean Algo:

$$r_1 = h - g = 3 \cdot 2^{592} x^2 + 3 \cdot 2^{296} x + c_1 - c_2 \qquad \text{cancel } x^3$$

$$r_2 = g - \left(3 \cdot 2^{592}\right)^{-1} x r_1 - * \cdot r_1 = k(x - m_1) \qquad \text{cancel } x^3, x^2$$

for some $k \in \mathbb{Z}_n$.

Example

- message: "Diary entry ??: Today I investigated [secret stuff]."
- ?? are consecutive numbers,
- assume only last digit changed: $f(x) = x + 2^{37 \cdot 8}$ (count bytes)

$$g(x) = x^3 - c_1 \qquad h(x) = (x + 2^{296})^3 - c_2$$

Calling Euclidean Algo:

$$r_1 = h - g = 3 \cdot 2^{592} x^2 + 3 \cdot 2^{296} x + c_1 - c_2 \qquad \text{cancel } x^3$$

$$r_2 = g - \left(3 \cdot 2^{592}\right)^{-1} x r_1 - * \cdot r_1 = k(x - m_1) \qquad \text{cancel } x^3, x^2$$

for some $k \in \mathbb{Z}_n$.

If inverting $3 \cdot 2^{592}$ or k fails, then $\gcd(*, n) \in \{p, q\}$.
So we get m_1 (and also m_2).

But there's an even less artificial scenario ...

Coppersmith Short Pad

"Why 256 Bit padding is not enough for $e = 3$."

Theorem

Let $R \leq \log(n)/e^2$, and $m_i = m \cdot 2^R + r_i$ for $i = 1, 2$. Then we can (probably) recover the message from the cipher $c_i = m_i^e \bmod n$. This always works for $e = 3$.

Coppersmith Short Pad

"Why 256 Bit padding is not enough for $e = 3$."

Theorem

Let $R \leq \log(n)/e^2$, and $m_i = m \cdot 2^R + r_i$ for $i = 1, 2$. Then we can (probably) recover the message from the cipher $c_i = m_i^e \bmod n$. This always works for $e = 3$.

Scenario

- intercept handshake
- receiver won't send ACK
- handshake is sent again, but with different random padding

Starting to break it

Define polynomials

$$g(x, y) = x^e - c_1 \qquad h(x, y) = (x + y)^e - c_2$$

- See y as parameter and x as actual variable.
- If $y = r_2 - r_1$, then common root
  $g(2^R m + r_1, y) = h(2^R m + r_1, y) = 0$.

Starting to break it

Define polynomials

$$g(x, y) = x^e - c_1 \qquad h(x, y) = (x + y)^e - c_2$$

- See y as parameter and x as actual variable.
- If $y = r_2 - r_1$, then common root
  $g(2^R m + r_1, y) = h(2^R m + r_1, y) = 0$.

Algebra knows something about this:
Search Engine "polynomials common root" $\rightsquigarrow$ resultants

Lemma

Let R comm. ring with 1. If $g, h \in R[x]$, then the resultant $\mathrm{res}(g, h) = 0$ *iff they have a common root.*

Resultants

For $e = 3$ we have

$$\operatorname{res}(g, h) = \det \begin{pmatrix} 1 & 0 & 0 & -c_1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -c_1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -c_1 \\ 1 & 3y & 3y^2 & y^3 - c_2 & 0 & 0 \\ 0 & 1 & 3y & 3y^2 & y^3 - c_2 & 0 \\ 0 & 0 & 1 & 3y & 3y^2 & y^3 - c_2 \end{pmatrix}$$

Resultants

For $e = 3$ we have

$$\operatorname{res}(g, h) = \det \begin{pmatrix} 1 & 0 & 0 & -c_1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -c_1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -c_1 \\ 1 & 3y & 3y^2 & y^3 - c_2 & 0 & 0 \\ 0 & 1 & 3y & 3y^2 & y^3 - c_2 & 0 \\ 0 & 0 & 1 & 3y & 3y^2 & y^3 - c_2 \end{pmatrix}$$

- only last e rows contain parameter y
- maximal power y^e in each of them
- in total maximal power $(y^e)^e = y^{e^2}$
- Resultant is polynomial in y of degree $e^2 (= 9)$

Coppersmith + Franklin-Reiter

- $\text{res}(g, h) \in \mathbb{Z}_n[y]$ of degree e^2
- Assumptions: $y = r_2 - r_1 < 2^R$, with $R \leq \log(n)/e^2$
- hence $y < \sqrt[e^2]{n}$, so find via Coppersmith

Coppersmith + Franklin-Reiter

- $\text{res}(g, h) \in \mathbb{Z}_n[y]$ of degree e^2
- Assumptions: $y = r_2 - r_1 < 2^R$, with $R \leq \log(n)/e^2$
- hence $y < \sqrt[e^2]{n}$, so find via Coppersmith

But now we have Franklin Reiter with linear f:

- Relation: $m_2 = m_1 + y =: f(m_1)$
- (Try to) Recover via

$$x - m_1 = \gcd\left(x^e - c_1, (x + y)^e - c_2\right)$$

- original message $m = m_1 // 2^R = (m_1 >> R)$

Coppersmith + Franklin-Reiter

- $\text{res}(g, h) \in \mathbb{Z}_n[y]$ of degree e^2
- Assumptions: $y = r_2 - r_1 < 2^R$, with $R \leq \log(n)/e^2$
- hence $y < \sqrt[e^2]{n}$, so find via Coppersmith

But now we have Franklin Reiter with linear f:

- Relation: $m_2 = m_1 + y =: f(m_1)$
- (Try to) Recover via

$$x - m_1 = \gcd\left(x^e - c_1, (x + y)^e - c_2\right)$$

- original message $m = m_1 // 2^R = (m_1 >> R)$

Formally, R is not given, but for $n \sim 4096$ bits, we have $R \leq 455$, so just bruteforce.

Very Small Message m

Brute Force via Meet-in-the-Middle:
Assume $m = m_1 m_2$ in $\mathbb{Z}$ where $m_i \leq 2^{b_i}$
Rewrite:

$$c = m^e \mod n \implies c m_1^{-e} \equiv m_2^e \mod n$$

Very Small Message m

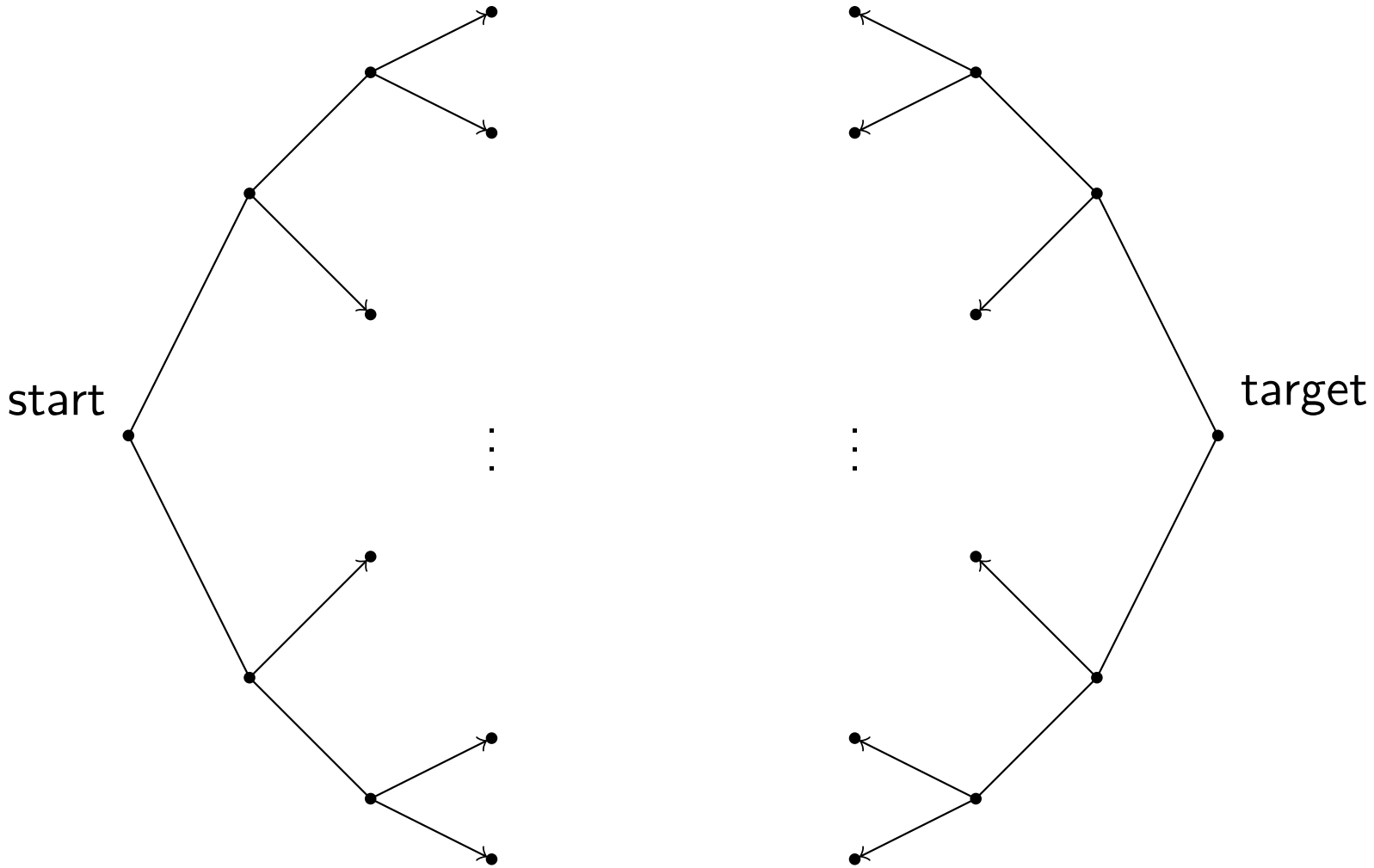
Brute Force via Meet-in-the-Middle:
Assume $m = m_1 m_2$ in $\mathbb{Z}$ where $m_i \leq 2^{b_i}$
Rewrite:

$$c = m^e \mod n \implies cm_1^{-e} \equiv m_2^e \mod n$$

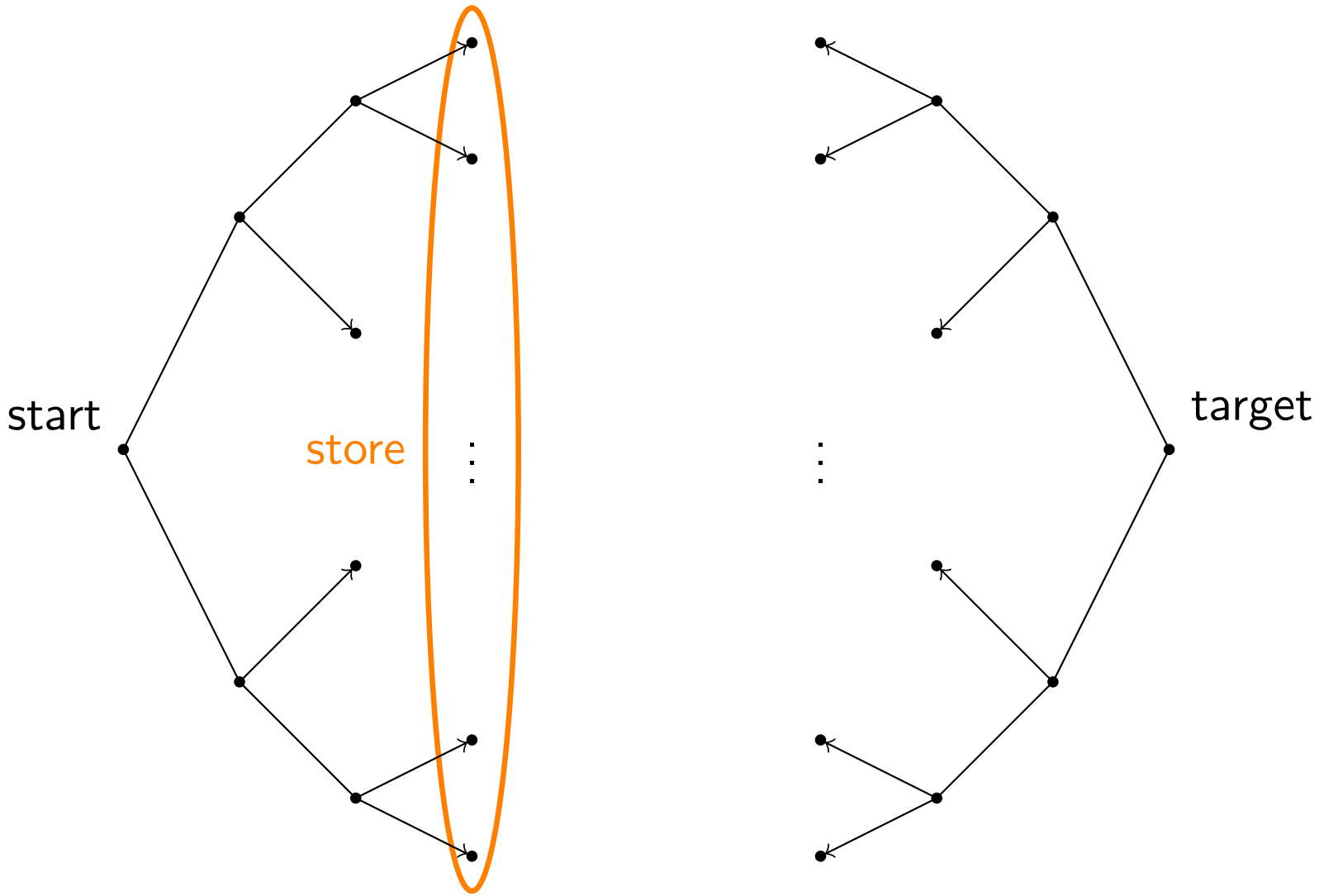
Strategy:

- List $cm_1^{-e} \mod n$ for all $m_1 \leq 2^{b_1} \rightsquigarrow$ (parallel write, not trivial)
- Look up $m_2^e \mod n$ for all $m_2 \leq 2^{b_2} \rightsquigarrow$ (parallel, only read)
- Search for collision

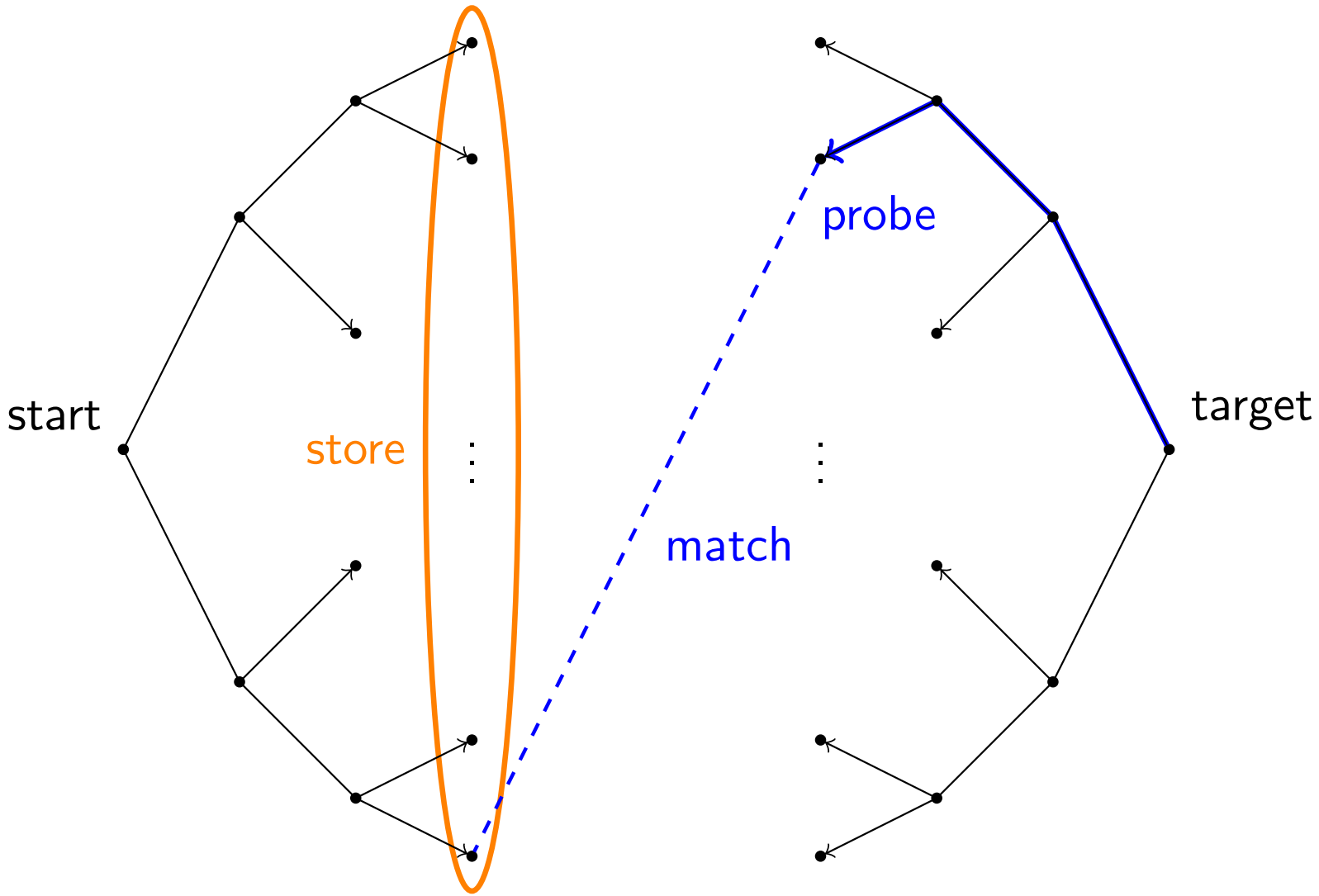
Meet in the Middle



Meet in the Middle



Meet in the Middle



Very Small Message m

- List $cm_1^{-e} \mod n$ for all $m_1 \leq 2^{b_1} \rightsquigarrow$ store
- Look up $m_2^e \mod n$ for all $m_2 \leq 2^{b_2} \rightsquigarrow$ don't store
- Search for collision should be $\mathcal{O}(1)$

Very Small Message m

- List $cm_1^{-e} \mod n$ for all $m_1 \leq 2^{b_1} \rightsquigarrow$ store
- Look up $m_2^e \mod n$ for all $m_2 \leq 2^{b_2} \rightsquigarrow$ don't store
- Search for collision should be $\mathcal{O}(1)$

Analysis (assume $b_1 < b_2$):

Time: $\mathcal{O}(2^{b_1} + 2^{b_2}) = \mathcal{O}\left(2^{b_2}\right)$ exp. and $\mathcal{O}(2^{b_1})$ inv.

Memory: $\mathcal{O}\left(2^{b_1} \log n\right)$

Total: if all goes well $b_1 = b_2$, so $2^{b_1} \approx \sqrt{m}$
$\mathcal{O}(\sqrt{m} \cdot \text{poly})$ time and space

Compare: Brute Force $\mathcal{O}(m \cdot \text{poly})$ time, but $\mathcal{O}(\log n)$ space
$\rightsquigarrow$ Space-Time-Tradeoff

Very Small Message m

- List $cm_1^{-e} \mod n$ for all $m_1 \leq 2^{b_1} \rightsquigarrow$ store
- Look up $m_2^e \mod n$ for all $m_2 \leq 2^{b_2} \rightsquigarrow$ don't store
- Search for collision should be $\mathcal{O}(1)$

Analysis (assume $b_1 < b_2$):

Time: $\mathcal{O}(2^{b_1} + 2^{b_2}) = \mathcal{O}\left(2^{b_2}\right)$ exp. and $\mathcal{O}(2^{b_1})$ inv.

Memory: $\mathcal{O}\left(2^{b_1} \log n\right)$

Total: if all goes well $b_1 = b_2$, so $2^{b_1} \approx \sqrt{m}$
$\mathcal{O}(\sqrt{m} \cdot \text{poly})$ time and space

Compare: Brute Force $\mathcal{O}(m \cdot \text{poly})$ time, but $\mathcal{O}(\log n)$ space
$\rightsquigarrow$ Space-Time-Tradeoff

The probability that a 64 bit number splits into two equally large parts lies around 18%.

Common Modulus

"What if n_i are not coprime, but same?"

Theorem

If we have keys (n, e_1) *and* (n, e_2) *with* $\gcd(e_1, e_2) = 1$*, then we can read every message sent to both.*

Common Modulus

"What if n_i are not coprime, but same?"

Theorem

If we have keys (n, e_1) *and* (n, e_2) *with* $\gcd(e_1, e_2) = 1$*, then we can read every message sent to both.*

Corollary

Every key needs its own modulus n, i.e. its own primes.

Common Modulus

"What if n_i are not coprime, but same?"

Theorem

If we have keys (n, e_1) and (n, e_2) with $\gcd(e_1, e_2) = 1$, then we can read every message sent to both.

Corollary

Every key needs its own modulus n, i.e. its own primes.

- know ciphers $c_i = m^{e_i} \bmod n$ for $i = 1, 2$
- Extended Euclid: $se_1 + te_2 = 1$, with $s < 0$ and $t > 0$
- compute $c_1^{-1} \bmod n$ (if it fails, we found a factor of n)

Common Modulus

"What if n_i are not coprime, but same?"

Theorem

If we have keys (n, e_1) and (n, e_2) with $\gcd(e_1, e_2) = 1$, then we can read every message sent to both.

Corollary

Every key needs its own modulus n, i.e. its own primes.

- know ciphers $c_i = m^{e_i} \bmod n$ for $i = 1, 2$
- Extended Euclid: $se_1 + te_2 = 1$, with $s < 0$ and $t > 0$
- compute $c_1^{-1} \bmod n$ (if it fails, we found a factor of n)
- Compute m via

$$\left(c_1^{-1}\right)^{|s|} c_2^t \equiv \left(\left(m^{e_1}\right)^{-1}\right)^{|s|} \left(m^{e_2}\right)^t \equiv m^{se_1+te_2} \equiv m \mod n$$

Recap – What can go wrong?

In general, RSA is secure but take care if:

- ✓ e **is small:** several attacks, find m
- ✓ m **is very small:** find m
- ✓ **same** n**, but different** e**:** find m

Recap – What can go wrong?

In general, RSA is secure but take care if:

- ✓ e **is small:** several attacks, find m
- ✓ m **is very small:** find m
- ✓ **same** n**, but different** e**:** find m
- d **is small:** find d, thus everything
- **fault in prime generation:** factor n, thus everything

Recap – What can go wrong?

In general, RSA is secure but take care if:

- ✓ e **is small:** several attacks, find m
- ✓ m **is very small:** find m
- ✓ **same** n**, but different** e**:** find m
- d **is small:** find d, thus everything
- **fault in prime generation:** factor n, thus everything

- previous attacks allowed to read messages

Recap – What can go wrong?

In general, RSA is secure but take care if:

- ✓ e **is small:** several attacks, find m
- ✓ m **is very small:** find m
- ✓ **same** n**, but different** e**:** find m
- d **is small:** find d, thus everything
- **fault in prime generation:** factor n, thus everything

- previous attacks allowed to read messages
- first 2 groups had assumptions on the message
  Fix: randomised padding, sufficiently long

Recap – What can go wrong?

In general, RSA is secure but take care if:

- ✓ e **is small:** several attacks, find m
- ✓ m **is very small:** find m
- ✓ **same** n**, but different** e**:** find m
- d **is small:** find d, thus everything
- **fault in prime generation:** factor n, thus everything

- previous attacks allowed to read messages
- first 2 groups had assumptions on the message
  Fix: randomised padding, sufficiently long
- same modulus allowed reading every message
  Fix: delete second key, just use first (both receivers can read each others messages anyway)

Recap – What can go wrong?

In general, RSA is secure but take care if:

- ✓ e **is small:** several attacks, find m
- ✓ m **is very small:** find m
- ✓ **same** n**, but different** e**:** find m
- d **is small:** find d, thus everything
- **fault in prime generation:** factor n, thus everything

- previous attacks allowed to read messages
- first 2 groups had assumptions on the message
  Fix: randomised padding, sufficiently long
- same modulus allowed reading every message
  Fix: delete second key, just use first (both receivers can read each others messages anyway)

other attacks completely break key

Small Private Exponent d

Theorem (Wiener, 1989)

Assume $q < p < 2q$, $e < \varphi(n)$ and $d < \frac{1}{3}n^{\frac{1}{4}}$. Then we can compute d from (n, e) in $\mathcal{O}\left(\log(n)^2\right)$ arithmetic steps.

Small Private Exponent d

Theorem (Wiener, 1989)

Assume $q < p < 2q$, $e < \varphi(n)$ and $d < \frac{1}{3}n^{\frac{1}{4}}$. Then we can compute d from (n, e) in $\mathcal{O}\left(\log(n)^2\right)$ arithmetic steps.

- Recall that finding d allows factoring n.
- Attack is based solely on the key.

Small Private Exponent d

Theorem (Wiener, 1989)

Assume $q < p < 2q$, $e < \varphi(n)$ and $d < \frac{1}{3}n^{\frac{1}{4}}$. Then we can compute d from (n, e) in $\mathcal{O}\left(\log(n)^2\right)$ arithmetic steps.

- Recall that finding d allows factoring n.
- Attack is based solely on the key.

Example

Let $n \sim 4096$ bit, choose $d \sim 1000$ bit and put $e = d^{-1} \bmod \varphi(n)$.
$\rightsquigarrow$ already unsafe

Consequences

- Decrypting "always" takes rather long

Small Private Exponent d

Theorem (Wiener, 1989)

Assume $q < p < 2q$, $e < \varphi(n)$ and $d < \frac{1}{3}n^{\frac{1}{4}}$. Then we can compute d from (n, e) in $\mathcal{O}\left(\log(n)^2\right)$ arithmetic steps.

- Recall that finding d allows factoring n.
- Attack is based solely on the key.

Example

Let $n \sim 4096$ bit, choose $d \sim 1000$ bit and put $e = d^{-1} \bmod \varphi(n)$.
$\rightsquigarrow$ already unsafe

Consequences

- Decrypting "always" takes rather long

proof uses continued fractions

Continued Fractions

Approximate large fractions by short fractions (in terms of bit size)

Continued Fractions

Approximate large fractions by short fractions (in terms of bit size)

Example (Euclidean Algo gcd(67, 24))

$$67 = 2 \cdot 24 + 19 \qquad 5 = 1 \cdot 4 + 1$$
$$24 = 1 \cdot 19 + 5 \qquad 4 = 4 \cdot 1 + 0$$
$$19 = 3 \cdot 5 + 4$$

Continued Fractions

Approximate large fractions by short fractions (in terms of bit size)

Example (Euclidean Algo gcd(67, 24))

$$67 = 2 \cdot 24 + 19 \qquad 5 = 1 \cdot 4 + 1$$
$$24 = 1 \cdot 19 + 5 \qquad 4 = 4 \cdot 1 + 0$$
$$19 = 3 \cdot 5 + 4$$

yields representation

$$\frac{67}{24} = 2 + \frac{19}{24} = 2 + \frac{1}{\frac{24}{19}} = 2 + \frac{1}{1 + \frac{5}{19}} = 2 + \frac{1}{1 + \frac{1}{3 + \frac{4}{5}}}$$
$$= 2 + \frac{1}{1 + \frac{1}{3 + \frac{1}{1 + \frac{1}{4}}}} =: [2; 1, 3, 1, 4]$$

divide – swap – repeat

Continued Fractions

Approximate large fractions by short fractions (in terms of bit size)

Example (Euclidean Algo gcd(67, 24))

$$67 = 2 \cdot 24 + 19 \qquad 5 = 1 \cdot 4 + 1$$
$$24 = 1 \cdot 19 + 5 \qquad 4 = 4 \cdot 1 + 0$$
$$19 = 3 \cdot 5 + 4$$

yields representation

$$\frac{67}{24} = 2 + \frac{19}{24} = 2 + \frac{1}{\frac{24}{19}} = 2 + \frac{1}{1 + \frac{5}{19}} = 2 + \frac{1}{1 + \frac{1}{3 + \frac{4}{5}}}$$
$$= 2 + \frac{1}{1 + \frac{1}{3 + \frac{1}{1 + \frac{1}{4}}}} =: [2; 1, 3, 1, 4]$$

divide – swap – repeat

Example (cont.)

What if we stop at some intermediate step?

Example (cont.)

What if we stop at some intermediate step?

$$[2;1,3,1,4] = 2 + \cfrac{1}{1+\cfrac{1}{3+\cfrac{1}{1+\frac{1}{4}}}} = \frac{67}{24} \quad \Big| \quad \Delta = 0$$

Example (cont.)

What if we stop at some intermediate step?

$$[2;1,3,1,4] = 2+\cfrac{1}{1+\cfrac{1}{3+\cfrac{1}{1+\frac{1}{4}}}} = \frac{67}{24} \quad \Big| \quad \Delta = 0$$

$$[2;1,3,1] = 2+\cfrac{1}{1+\cfrac{1}{3+\frac{1}{1}}} = \frac{14}{5} \quad \Big| \quad \Delta = \frac{1}{120}$$

Example (cont.)

What if we stop at some intermediate step?

$$
\begin{array}{lcccc|lcl}
[2;1,3,1,4] & = & 2+\cfrac{1}{1+\cfrac{1}{3+\cfrac{1}{1+\frac{1}{4}}}} & = & \frac{67}{24} & \Delta & = & 0 \\
[2;1,3,1] & = & 2+\cfrac{1}{1+\cfrac{1}{3+\frac{1}{1}}} & = & \frac{14}{5} & \Delta & = & \frac{1}{120} \\
[2;1,3] & = & 2+\cfrac{1}{1+\frac{1}{3}} & = & \frac{11}{4} & \Delta & = & -\frac{1}{24}
\end{array}
$$

Example (cont.)

What if we stop at some intermediate step?

$$
\begin{array}{lcccc|lcl}
[2;1,3,1,4] & = & 2+\cfrac{1}{1+\cfrac{1}{3+\cfrac{1}{1+\frac{1}{4}}}} & = & \frac{67}{24} & \Delta & = & 0 \\
[2;1,3,1] & = & 2+\cfrac{1}{1+\cfrac{1}{3+\frac{1}{1}}} & = & \frac{14}{5} & \Delta & = & \frac{1}{120} \\
[2;1,3] & = & 2+\cfrac{1}{1+\frac{1}{3}} & = & \frac{11}{4} & \Delta & = & -\frac{1}{24} \\
[2;1] & = & 2+\frac{1}{1} & = & 3 & \Delta & = & \frac{5}{24}
\end{array}
$$

Example (cont.)

What if we stop at some intermediate step?

$$
\begin{array}{lcll|lcl}
[2;1,3,1,4] & = & 2+\cfrac{1}{1+\cfrac{1}{3+\cfrac{1}{1+\frac{1}{4}}}} & = \frac{67}{24} & \Delta & = & 0 \\
[2;1,3,1] & = & 2+\cfrac{1}{1+\cfrac{1}{3+\frac{1}{1}}} & = \frac{14}{5} & \Delta & = & \frac{1}{120} \\
[2;1,3] & = & 2+\cfrac{1}{1+\frac{1}{3}} & = \frac{11}{4} & \Delta & = & -\frac{1}{24} \\
[2;1] & = & 2+\frac{1}{1} & = 3 & \Delta & = & \frac{5}{24} \\
[2] & & & = 2 & \Delta & = & -\frac{19}{24}
\end{array}
$$

Example (cont.)

What if we stop at some intermediate step?

$$
\begin{array}{lcccc|lcl}
[2;1,3,1,4] & = & 2+\cfrac{1}{1+\cfrac{1}{3+\cfrac{1}{1+\frac{1}{4}}}} & = & \frac{67}{24} & \Delta & = & 0 \\
[2;1,3,1] & = & 2+\cfrac{1}{1+\cfrac{1}{3+\frac{1}{1}}} & = & \frac{14}{5} & \Delta & = & \frac{1}{120} \\
[2;1,3] & = & 2+\cfrac{1}{1+\frac{1}{3}} & = & \frac{11}{4} & \Delta & = & -\frac{1}{24} \\
[2;1] & = & 2+\frac{1}{1} & = & 3 & \Delta & = & \frac{5}{24} \\
[2] & & & = & 2 & \Delta & = & -\frac{19}{24}
\end{array}
$$

Observations

- difference alternates sign
- absolute value of difference decreases ↑
- enumerator and denominator increase ↑

Continued Fractions in General

- Input $\frac{a}{b} \in \mathbb{Q}$, ($a > b$ else we start with $[0; \ldots]$)
- Euclid: start with $r_{-1} = a$, $r_0 = b$
  recursion: $r_{k-1} = z_k r_k + r_{k+1}$
- n-th convergent: $[z_0; z_1, \ldots, z_n] = \frac{p_n}{q_n}$

$$p_k = z_k p_{k-1} + p_{k-2} \qquad p_{-1} = 1 \qquad p_{-2} = 0$$
$$q_k = z_k q_{k-1} + q_{k-2} \qquad q_{-1} = 0 \qquad q_{-2} = 1$$

Continued Fractions in General

- Input $\frac{a}{b} \in \mathbb{Q}$, ($a > b$ else we start with $[0; \ldots]$)
- Euclid: start with $r_{-1} = a$, $r_0 = b$
  recursion: $r_{k-1} = z_k r_k + r_{k+1}$
- n-th convergent: $[z_0; z_1, \ldots, z_n] = \frac{p_n}{q_n}$

$$p_k = z_k p_{k-1} + p_{k-2} \qquad p_{-1} = 1 \qquad p_{-2} = 0$$
$$q_k = z_k q_{k-1} + q_{k-2} \qquad q_{-1} = 0 \qquad q_{-2} = 1$$

Remark

Generalised idea also works for $x \in \mathbb{R}$:

$$x_0 = x \qquad z_k = \lfloor x_k \rfloor \qquad x_{k+1} = \frac{1}{x_k - \lfloor x_k \rfloor}$$

Continued Fractions in General

- Input $\frac{a}{b} \in \mathbb{Q}$, ($a > b$ else we start with $[0; \ldots]$)
- Euclid: start with $r_{-1} = a$, $r_0 = b$
  recursion: $r_{k-1} = z_k r_k + r_{k+1}$
- n-th convergent: $[z_0; z_1, \ldots, z_n] = \frac{p_n}{q_n}$

$$p_k = z_k p_{k-1} + p_{k-2} \qquad p_{-1} = 1 \qquad p_{-2} = 0$$
$$q_k = z_k q_{k-1} + q_{k-2} \qquad q_{-1} = 0 \qquad q_{-2} = 1$$

Remark

Generalised idea also works for $x \in \mathbb{R}$:

$$x_0 = x \qquad z_k = \lfloor x_k \rfloor \qquad x_{k+1} = \frac{1}{x_k - \lfloor x_k \rfloor}$$

But we are mainly interested in Rationals and finite fractions.

Bonus Slide

Can be infinite

$$x = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \ldots}}} =: [1; 1, 1, 1, 1, \ldots]$$

yields $x = 1 + \frac{1}{x}$,

Bonus Slide

Can be infinite

$$x = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \ldots}}} =: [1; 1, 1, 1, 1, \ldots]$$

yields $x = 1 + \frac{1}{x}$, hence x is golden ratio φ, so can be irrational

Bonus Slide

Can be infinite

$$x = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \ldots}}} =: [1; 1, 1, 1, 1, \ldots]$$

yields $x = 1 + \frac{1}{x}$, hence x is golden ratio φ, so can be irrational

Theorem

The infinite, periodic continued fractions correspond to solutions of quadratic equations, i.e. algebraic numbers of degree 2.

Bonus Slide

Can be infinite

$$x = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \ldots}}} =: [1; 1, 1, 1, 1, \ldots]$$

yields $x = 1 + \frac{1}{x}$, hence x is golden ratio φ, so can be irrational

Theorem

The infinite, periodic continued fractions correspond to solutions of quadratic equations, i.e. algebraic numbers of degree 2.

- φ has n-th convergent $\frac{F_{n+2}}{F_{n+1}}$, with Fibonacci numbers $F_0 = 0$
- φ is worst number to approximate,
  as Fibonacci numbers are worst case for Euclidean Algorithm

Properties of Continued Fractions

Improving: each step improves approximation

Alternating: even $\rightarrow$ smaller, odd $\rightarrow$ larger value

$$\frac{p_{2i}}{q_{2i}} < \frac{p_{2(i+1)}}{q_{2(i+1)}} \leq x \leq \frac{p_{2(i+1)+1}}{q_{2(i+1)+1}} < \frac{p_{2i+1}}{q_{2i+1}}$$

Properties of Continued Fractions

Improving: each step improves approximation

Alternating: even $\rightarrow$ smaller, odd $\rightarrow$ larger value

$$\frac{p_{2i}}{q_{2i}} < \frac{p_{2(i+1)}}{q_{2(i+1)}} \leq x \leq \frac{p_{2(i+1)+1}}{q_{2(i+1)+1}} < \frac{p_{2i+1}}{q_{2i+1}}$$

good approximation: $\left| x - \frac{p_i}{q_i} \right| < \frac{1}{q_i^2}$

Properties of Continued Fractions

Improving: each step improves approximation

Alternating: even $\rightarrow$ smaller, odd $\rightarrow$ larger value

$$\frac{p_{2i}}{q_{2i}} < \frac{p_{2(i+1)}}{q_{2(i+1)}} \leq x \leq \frac{p_{2(i+1)+1}}{q_{2(i+1)+1}} < \frac{p_{2i+1}}{q_{2i+1}}$$

good approximation: $\left| x - \frac{p_i}{q_i} \right| < \frac{1}{q_i^2}$

best approximation: Let $\frac{p_i}{q_i}$ a cont.frac. of x.

If $\frac{p}{q}$ is a better approximation, then $q > q_i$.

Properties of Continued Fractions

Improving: each step improves approximation

Alternating: even $\rightarrow$ smaller, odd $\rightarrow$ larger value

$$\frac{p_{2i}}{q_{2i}} < \frac{p_{2(i+1)}}{q_{2(i+1)}} \leq x \leq \frac{p_{2(i+1)+1}}{q_{2(i+1)+1}} < \frac{p_{2i+1}}{q_{2i+1}}$$

good approximation: $\left| x - \frac{p_i}{q_i} \right| < \frac{1}{q_i^2}$

best approximation: Let $\frac{p_i}{q_i}$ a cont.frac. of x.

If $\frac{p}{q}$ is a better approximation, then $q > q_i$.

"only" good approximation: $\left| x - \frac{p}{q} \right| < \frac{1}{2q^2} \implies \frac{p}{q}$ is a cont.frac.

Small Private Exponent d

Theorem (Wiener, 1989, slightly generalised)

Assume $q < p < aq$, $e < \varphi(n)$ *and* $d < \frac{1}{\sqrt{2(a+1)}} n^{\frac{1}{4}}$. *Then we can compute* d *from* (n, e) *in* $\mathcal{O}\left(\log(n)^2\right)$ *arithmetic steps.*

Small Private Exponent d

Theorem (Wiener, 1989, slightly generalised)

Assume $q < p < aq$, $e < \varphi(n)$ *and* $d < \frac{1}{\sqrt{2(a+1)}} n^{\frac{1}{4}}$. *Then we can compute* d *from* (n, e) *in* $\mathcal{O}\left(\log(n)^2\right)$ *arithmetic steps.*

Proof Idea.

- Idea: Approximate $\frac{e}{n}$ with cont.frac.
- We have $ed - k\varphi(n) = 1$ for some unknown $k, d, \varphi(n)$
- have $\varphi(n) \approx n$, slightly smaller, hence $\frac{e}{n} \approx \frac{e}{\varphi(n)}$
- estimate error $\left|\frac{e}{n} - \frac{k}{d}\right| < \ldots < \frac{1}{2d^2}$

Small Private Exponent d

Theorem (Wiener, 1989, slightly generalised)

Assume $q < p < aq$, $e < \varphi(n)$ *and* $d < \frac{1}{\sqrt{2(a+1)}} n^{\frac{1}{4}}$. *Then we can compute* d *from* (n, e) *in* $\mathcal{O}\left(\log(n)^2\right)$ *arithmetic steps.*

Proof Idea.

- Idea: Approximate $\frac{e}{n}$ with cont.frac.
- We have $ed - k\varphi(n) = 1$ for some unknown $k, d, \varphi(n)$
- have $\varphi(n) \approx n$, slightly smaller, hence $\frac{e}{n} \approx \frac{e}{\varphi(n)}$
- estimate error $\left|\frac{e}{n} - \frac{k}{d}\right| < \ldots < \frac{1}{2d^2}$
- $\implies$ $\frac{k}{d}$ is a cont.frac. of $\frac{e}{n}$

Small Private Exponent d

Theorem (Wiener, 1989, slightly generalised)

Assume $q < p < aq$, $e < \varphi(n)$ *and* $d < \frac{1}{\sqrt{2(a+1)}} n^{\frac{1}{4}}$. *Then we can compute* d *from* (n, e) *in* $\mathcal{O}\left(\log(n)^2\right)$ *arithmetic steps.*

Proof Idea.

- Idea: Approximate $\frac{e}{n}$ with cont.frac.
- We have $ed - k\varphi(n) = 1$ for some unknown $k, d, \varphi(n)$
- have $\varphi(n) \approx n$, slightly smaller, hence $\frac{e}{n} \approx \frac{e}{\varphi(n)}$
- estimate error $\left|\frac{e}{n} - \frac{k}{d}\right| < \ldots < \frac{1}{2d^2}$
- $\implies$ $\frac{k}{d}$ is a cont.frac. of $\frac{e}{n}$
- compute all continued fractions $\rightsquigarrow$ list of $\log n$ candidates
  a) check decoding: $2^{ed} \bmod n \stackrel{?}{=} 2$
  b) try to factor n, note we also have k, thus $\varphi(n)$ □

Proof for Wiener attack.

Error from $\varphi(n)$ to n:

$$0 < n - \varphi(n) = p + q - 1 < (a+1)q \leq (a+1)\sqrt{n}$$

Proof for Wiener attack.

Error from $\varphi(n)$ to n:

$$0 < n - \varphi(n) = p + q - 1 < (a+1)q \leq (a+1)\sqrt{n}$$

Error between fractions:

$$\begin{aligned}\left|\frac{e}{n} - \frac{k}{d}\right| &= \left|\frac{ed - k\varphi(n) - kn + k\varphi(n)}{nd}\right| \\ &= \left|\frac{1 - k(n - \varphi(n))}{nd}\right| < \frac{(a+1)k\sqrt{n}}{nd} = \frac{(a+1)k}{d\sqrt{n}}\end{aligned}$$

Proof for Wiener attack.

Error from $\varphi(n)$ to n:

$$0 < n - \varphi(n) = p + q - 1 < (a+1)q \leq (a+1)\sqrt{n}$$

Error between fractions:

$$\left|\frac{e}{n} - \frac{k}{d}\right| = \left|\frac{ed - k\varphi(n) - kn + k\varphi(n)}{nd}\right|$$
$$= \left|\frac{1 - k(n - \varphi(n))}{nd}\right| < \frac{(a+1)k\sqrt{n}}{nd} = \frac{(a+1)k}{d\sqrt{n}}$$

$$k\varphi(n) = ed - 1 \text{ and } e < \varphi(n) \implies k < d$$
$$\implies \left|\frac{e}{n} - \frac{k}{d}\right| < \frac{a+1}{\sqrt{n}} \leq \frac{a+1}{2(a+1)d^2} = \frac{1}{2d^2}$$

Proof for Wiener attack.

Error from $\varphi(n)$ to n:

$$0 < n - \varphi(n) = p + q - 1 < (a+1)q \leq (a+1)\sqrt{n}$$

Error between fractions:

$$\left|\frac{e}{n} - \frac{k}{d}\right| = \left|\frac{ed - k\varphi(n) - kn + k\varphi(n)}{nd}\right|$$
$$= \left|\frac{1 - k(n - \varphi(n))}{nd}\right| < \frac{(a+1)k\sqrt{n}}{nd} = \frac{(a+1)k}{d\sqrt{n}}$$

$$k\varphi(n) = ed - 1 \text{ and } e < \varphi(n) \implies k < d$$
$$\implies \left|\frac{e}{n} - \frac{k}{d}\right| < \frac{a+1}{\sqrt{n}} \leq \frac{a+1}{2(a+1)d^2} = \frac{1}{2d^2}$$

hence $\frac{k}{d}$ is a continued fraction of $\frac{e}{n}$ □

Example (Wiener Attack)

- assume given public key

$$n = 389033 \qquad e = 332383$$

- calculate continued fractions

$$\begin{aligned}
\frac{332383}{389033} &= \frac{1}{1+\frac{56650}{332383}} && \rightsquigarrow 1 \\
&= \frac{1}{1+\frac{1}{5+\frac{49133}{56650}}} && \rightsquigarrow \frac{5}{6} \\
&= \frac{1}{1+\frac{1}{5+\frac{1}{1+\frac{7517}{49133}}}} && \rightsquigarrow \frac{6}{7}
\end{aligned}$$

- checking $e \cdot 7 - 1 \bmod 6 = 0$ and $2^{e \cdot 7} \bmod n = 2$
- hence $d = 7$

Outlook on Wiener's Attack

Extension to Wiener's Attack

- via lattice methods breakable for $d < n^{0.292}$
- assumed to work up to $d < \sqrt{n}$, but open problem

Outlook on Wiener's Attack

Extension to Wiener's Attack

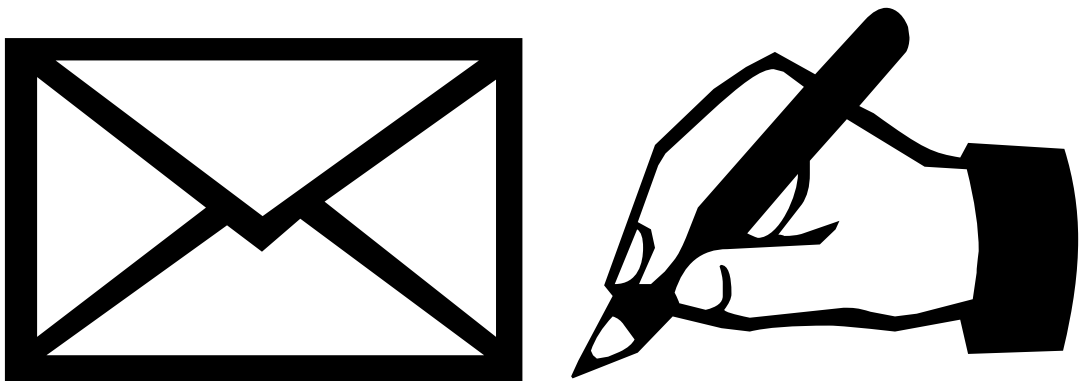
- via lattice methods breakable for $d < n^{0.292}$
- assumed to work up to $d < \sqrt{n}$, but open problem

Possible Countermeasures

- put $e' = e + * \cdot \varphi(n)$, destroys assumption $e < \varphi(n)$
- optimised decryption: make $d_p = d \bmod p - 1$ and $d_q = d \bmod q - 1$ small-ish
  can factor n in $\mathcal{O}\left(\min\left(\sqrt{d_p}, \sqrt{d_q}\right)\right)$

But ongoing research, so security unsure.

Digital Signatures



Digital Signatures

If electronic mail systems are to replace the existing paper mail system for business transactions, "signing" an electronic message must be possible. *(RSA, '77)*

Digital Signatures

If electronic mail systems are to replace the existing paper mail system for business transactions, "signing" an electronic message must be possible. *(RSA, '77)*

- Authentication: sender only has to convince recipient
- Signature: recipient can also convince "judge"
- must depend both on sender and message

  if not message: use old signature from other message
  if not sender: recipient can forge

Desired Property

often Encryption/Decryption commute

$$\mathrm{enc}\left(\mathrm{dec}\left(m\right)\right) = \mathrm{dec}\left(\mathrm{enc}\left(m\right)\right)$$

Desired Property

often Encryption/Decryption commute

$$\mathrm{enc}\left(\mathrm{dec}\left(m\right)\right) = \mathrm{dec}\left(\mathrm{enc}\left(m\right)\right)$$

Basic Idea (RSA, '77)

- Alice sends to Bob:
$$s = m^{d_A} \bmod n_A$$
- Bob gets s, checks with Alice's public key:
$$m = s^{e_A} \bmod n_A$$
- message m, given by s can only have come from Alice?

Desired Property

often Encryption/Decryption commute

$$\mathrm{enc}\,(\mathrm{dec}\,(m)) = \mathrm{dec}\,(\mathrm{enc}\,(m))$$

Basic Idea (RSA, '77)

- Alice sends to Bob:
$$s = m^{d_A} \bmod n_A$$
- Bob gets s, checks with Alice's public key:
$$m = s^{e_A} \bmod n_A$$
- message m, given by s can only have come from Alice? NO!

Desired Property

often Encryption/Decryption commute

$$\mathrm{enc}\,(\mathrm{dec}\,(m)) = \mathrm{dec}\,(\mathrm{enc}\,(m))$$

Basic Idea (RSA, '77)

- Alice sends to Bob:
$$s = m^{d_A} \bmod n_A$$
- Bob gets s, checks with Alice's public key:
$$m = s^{e_A} \bmod n_A$$
- message m, given by s can only have come from Alice? NO!

Problem

- What to check the message against?
- This setting is flawed!

Mathematical Model

Definition (Signature System)

A signature system is a quintuple $(P, S, K, \text{sign}, \text{vrfy})$ where

- P is the set of all plaintexts
- S is the set of all signatures
- K is the set of all keys
- $\text{sign} : P \times K \rightsquigarrow S$ is the signature relation
  (not necessarily a map)
- $\text{vrfy} : P \times S \times K \to \{0, 1\}$ is the verification function

$$\text{vrfy}(m, s, k) = \begin{cases} 1 & : s \in \text{sign}(m, k) \text{ i.e. possible outcome} \\ 0 & : \text{else} \end{cases}$$

- sign, vrfy are efficiently computable

Correctness: $\forall m \in \mathcal{M}, \forall k \in \mathcal{K} \,.\, \text{vrfy}(m, \text{sign}(m, k_{\text{priv}}), k_{\text{pub}}) = 1$

Observations

- We must be able to reject messages.
- Signature + message must contain redundancy.
- If message derived from signature, redundancy must be in message.

Observations

- We must be able to reject messages.
- Signature + message must contain redundancy.
- If message derived from signature, redundancy must be in message.

Improved Plain-RSA signature

- Alice computes $s := \text{sign}\,(m, (n, d)) = m^d \bmod n$
- Send (m, s) to Bob
- Bob gets (m', s'), checks $m' = s'^e \bmod n$. If yes, he accepts it.

Observations

- We must be able to reject messages.
- Signature + message must contain redundancy.
- If message derived from signature, redundancy must be in message.

Improved Plain-RSA signature

- Alice computes $s := \text{sign}\,(m, (n, d)) = m^d \bmod n$
- Send (m, s) to Bob
- Bob gets (m', s'), checks $m' = s'^e \bmod n$. If yes, he accepts it.

How does Bob get (n, e)?

- want to guard against manipulation of message
- transmitted public key could have been changed
- Public Key Infrastructure (PKI): topic of its own

Example

- Alice's key is $(n, e, d) = (1073, 17, 593)$.
- She want to send $m = 123$.
- Compute $s = 123^{593} \bmod 1073 = 219$.
- Bob gets $(m, s) = (123, 219)$ and knows (n, e).
- Bob checks $123 \stackrel{?}{=} 219^e \bmod n$
- They match, so Bob accepts the message.

Example

- Alice's key is $(n, e, d) = (1073, 17, 593)$.
- She want to send $m = 123$.
- Compute $s = 123^{593} \bmod 1073 = 219$.
- Bob gets $(m, s) = (123, 219)$ and knows (n, e).
- Bob checks $123 \stackrel{?}{=} 219^e \bmod n$
- They match, so Bob accepts the message.

RSA-specific problems

- Every number $s < n$ is a valid signature for some $m < n$.
- Plain-RSA is multiplicative: If (m_1, s_1) and (m_2, s_2) are valid pairs, then $(m_1 m_2, s_1 s_2)$ also is valid.

$$m_1 = s_1^e \qquad m_2 = s_2^e \implies m_1 m_2 = (s_1 s_2)^e$$

Signature Oracle Attack

Assumptions

- Assume we sign with plain-RSA
- want to forge signature for message m
- Have access to online oracle, that signs any $m' \neq m$ (or some restricted subset)

Signature Oracle Attack

Assumptions

- Assume we sign with plain-RSA
- want to forge signature for message m
- Have access to online oracle, that signs any $m' \neq m$ (or some restricted subset)

Attack

- factor $m = m_1 \dots m_k \bmod n$ such that all m_i accepted by oracle (not necessarily prime factors)
  e.g. pick some $m_1 < n$ and put $m_2 := m \cdot m_1^{-1} \bmod n$
- get $s_i = m_i^d \bmod n$ for $i = 1, \dots, k$
- have signature $s = \prod s_i$ for m

Example (Signature Oracle Attack)

- public key $(3084396941, 5)$
- want to forge signature for `flag`
- factor `flag`: $5 \cdot 499 \cdot 688729$
- ask signature oracle

$$s_1 = \text{sign}(5) \qquad s_2 = \text{sign}(499) \qquad s_3 = \text{sign}(688729)$$

- send $s := s_1 \cdot s_2 \cdot s_3$

Remark

- *possibly additional tricks with 0-bytes if in C*

Attack Scenarios

What does Eve know?

No message: just public key

Signatures: Eve has some message-signature pairs (m_i, s_i)

e.g. observing traffic

Chosen message: Eve can choose messages m_i to be signed

e.g. impersonating authentication server

Attack Scenarios

What does Eve know?

No message: just public key

Signatures: Eve has some message-signature pairs (m_i, s_i)
e.g. observing traffic

Chosen message: Eve can choose messages m_i to be signed
e.g. impersonating authentication server

What is a success?

Total Break: find private key

Universal Forgeability: forge signature for every message

Selective Forgeability: forge signature for m given by Alice

Existential Forgeability: forge signature for m chosen by Eve

Goal

Strongest Security

EUF-CMA Existential Unforgeability under Chosen message Attack:

- Eve may request signatures s_i for $m_1, \ldots, m_k$
- forges signature s for some $m \notin \{m_1, \ldots, m_k\}$

Goal

Strongest Security

EUF-CMA Existential Unforgeability under Chosen message Attack:

- Eve may request signatures s_i for $m_1, \ldots, m_k$
- forges signature s for some $m \notin \{m_1, \ldots, m_k\}$

sEUF-CMA strong EUF-CMA

- Eve may request signatures $s_i = \mathrm{sign}(m_i)$
- forges pair $(m, s) \notin \{(m_i, s_i) : i = 1, \ldots\}$;
  i.e. m may be among requested messages, but must forge different valid signature

Goal

Strongest Security

EUF-CMA Existential Unforgeability under Chosen message Attack:

- Eve may request signatures s_i for $m_1, \ldots, m_k$
- forges signature s for some $m \notin \{m_1, \ldots, m_k\}$

sEUF-CMA strong EUF-CMA

- Eve may request signatures $s_i = \text{sign}(m_i)$
- forges pair $(m, s) \notin \{(m_i, s_i) : i = 1, \ldots\}$;
  i.e. m may be among requested messages, but must forge different valid signature

Plain RSA fails:

- EUF with no message
- Universal Unforgeability (UUF) under CMA

Public Key Cryptography Standard

OAEP – Optimal Asymmetric Encryption Padding

"How to do it right."

- part of PKCS #1, version 2.2,
- RFC 8017, October 2012, last update Nov. 2016

OAEP – Optimal Asymmetric Encryption Padding

"How to do it right."

- part of PKCS #1, version 2.2,
- RFC 8017, October 2012, last update Nov. 2016

Parameters

- hash function $h : \text{Byte}^* \to \text{Byte}^{\text{hLen}}$
  - recommended: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, and SHA-512/256 (i.e. SHA-2)
  - SHA-3 was too fresh, unclear why not included in update

OAEP – Optimal Asymmetric Encryption Padding

"How to do it right."

- part of PKCS #1, version 2.2,
- RFC 8017, October 2012, last update Nov. 2016

Parameters

- hash function $h : \text{Byte}^* \to \text{Byte}^{\text{hLen}}$
  - recommended: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, and SHA-512/256 (i.e. SHA-2)
  - SHA-3 was too fresh, unclear why not included in update
- mask generation function $\text{MGF} : (\text{seed}, \ell) \mapsto \text{Byte}^{\ell}$

  $T \leftarrow$ empty string
  for $c = 0$ **to** $\lceil \ell / \text{hLen} \rceil - 1$ **do**
  $T \leftarrow T \,||\, h(\text{seed} \,||\, c)$

OAEP-Encryption

Encryption

- (n, e) public RSA key
- m message, $\|m\| \leq \|n\| - 2\text{hLen} - 2$
- L label (optional), default empty

OAEP-Encryption

Encryption

- (n, e) public RSA key
- m message, $\|m\| \leq \|n\| - 2\text{hLen} - 2$
- L label (optional), default empty

function ENCRYPT(m, L)

DB $\leftarrow h(L) \,||\, 00 \ldots 0001 \,||\, m$ ▷ data block

Seed $\leftarrow$ random seed of length hLen

mDB $\leftarrow$ MGF(Seed) $\oplus$ DB ▷ masked DB

mSeed $\leftarrow$ Seed $\oplus$ MGF(mDB) ▷ masked seed

EM $\leftarrow 00 \,||\, \text{mSeed} \,||\, \text{mDB}$ ▷ encoded message, EM $< n$

return $\text{EM}^e \bmod n$

OAEP-Encryption

Encryption

- (n, e) public RSA key
- m message, $\|m\| \leq \|n\| - 2\text{hLen} - 2$
- L label (optional), default empty

function ENCRYPT(m, L)

DB $\leftarrow h(L) \,||\, 00 \ldots 0001 \,||\, m$ ▷ data block

Seed $\leftarrow$ random seed of length hLen

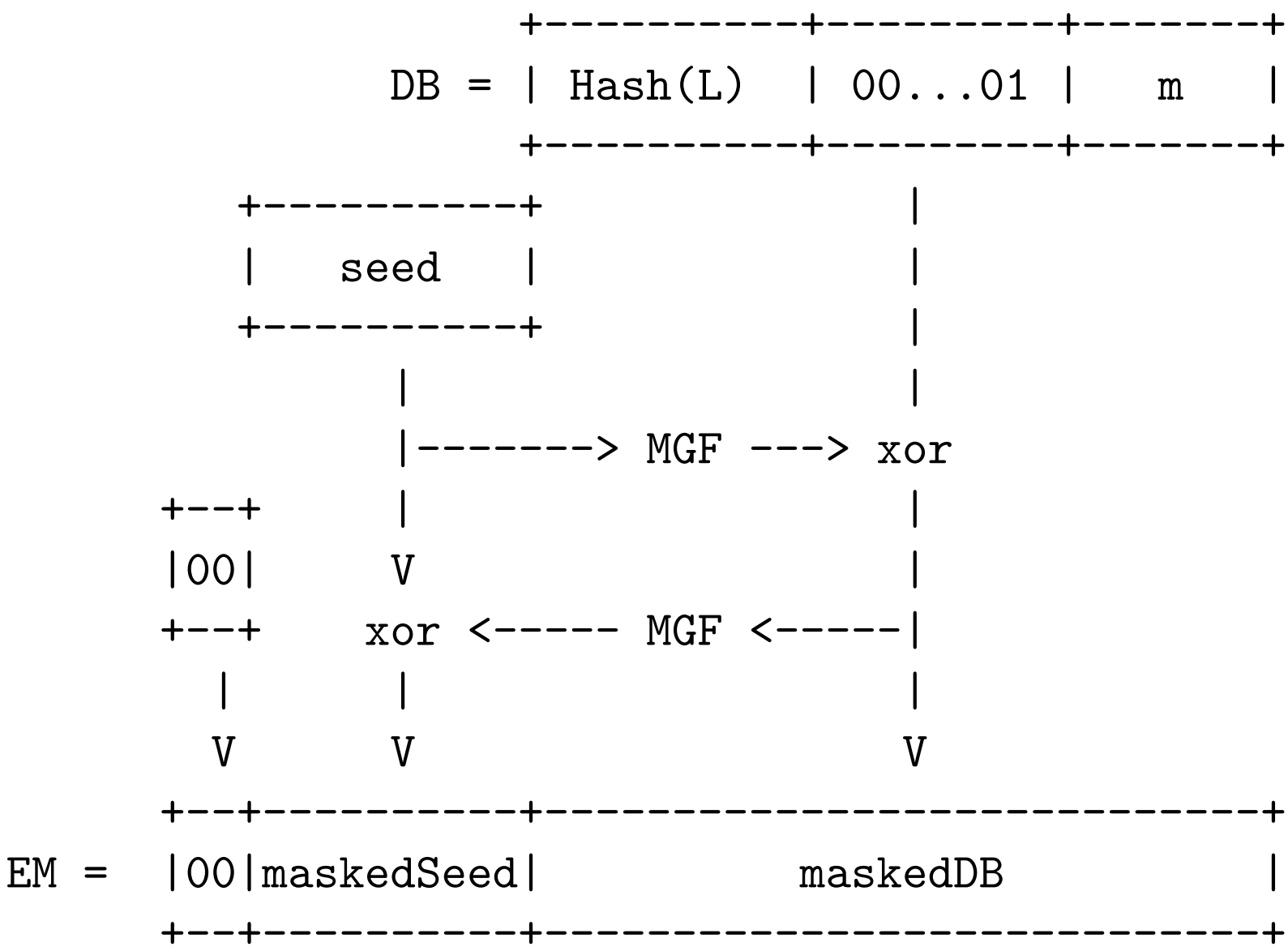
mDB $\leftarrow$ MGF(Seed) $\oplus$ DB ▷ masked DB

mSeed $\leftarrow$ Seed $\oplus$ MGF(mDB) ▷ masked seed

EM $\leftarrow 00 \,||\, \text{mSeed} \,||\, \text{mDB}$ ▷ encoded message, EM $< n$

return $\text{EM}^e \bmod n$

- Payload m: 50% - 89% of cipher, $\geq$ 1000 Bit
- more than enough for AES key
- continue with symmetric encryption


DB = | Hash(L) | 00...01 | m |
seed
MGF
xor
MGF
xor
EM = |00|maskedSeed| maskedDB |

RSASSA-PSS – Idea

Naming

SSA Signature Scheme with Appendix

PSS Probabilistic Signature Scheme

EMSA Encoding Methods for Signatures with Appendix

RSASSA-PSS – Idea

Naming

SSA Signature Scheme with Appendix

PSS Probabilistic Signature Scheme

EMSA Encoding Methods for Signatures with Appendix

Sign

- encode message with EMSA-PSS: $\mathrm{EM} = \mathrm{encode}(m)$
- apply RSA primitive/plain-RSA: $s = \mathrm{EM}^d \bmod n$

RSASSA-PSS – Idea

Naming

SSA Signature Scheme with Appendix

PSS Probabilistic Signature Scheme

EMSA Encoding Methods for Signatures with Appendix

Sign

- encode message with EMSA-PSS: $\mathrm{EM} = \mathrm{encode}(m)$
- apply RSA primitive/plain-RSA: $s = \mathrm{EM}^d \bmod n$

Verify

- apply RSA primitive/plain-RSA: $\mathrm{EM} = s^e \bmod n$
- check consistency with EMSA-PSS-VERIFY

RSASSA-PSS – Idea

Naming

SSA Signature Scheme with Appendix

PSS Probabilistic Signature Scheme

EMSA Encoding Methods for Signatures with Appendix

Sign

- encode message with EMSA-PSS: $\mathrm{EM} = \mathrm{encode}(m)$
- apply RSA primitive/plain-RSA: $s = \mathrm{EM}^d \bmod n$

Verify

- apply RSA primitive/plain-RSA: $\mathrm{EM} = s^e \bmod n$
- check consistency with EMSA-PSS-VERIFY

- encoding uses $\mathrm{hash}(m)$ instead of m
- can sign arbitrarily long message (document)

RSASSA-PSS – Details

Arguments for Encoding

- m message to be signed
- h hash function
- MGF mask generation function
- sLen salt length (bytes), mostly hash length or 0
- L desired output length, $\geq \|h(*)\| + \text{sLen} + 2$

RSASSA-PSS – Details

Arguments for Encoding

- m message to be signed
- h hash function
- MGF mask generation function
- sLen salt length (bytes), mostly hash length or 0
- L desired output length, $\geq \|h(*)\| + \text{sLen} + 2$

Verification

- $m' = 00 \ldots 00 \,||\, h(m) \,||\, \text{salt}$ with 8 Zero-bytes
- $\text{DB} = 00 \ldots 0001 \,||\, \text{salt}$ of length $L - \|h(*)\| - 1$
- mask DB with $M(h(m'))$
- output $\text{EM} = \text{maskedDB} \,||\, h(m') \,||\, 0xbc$

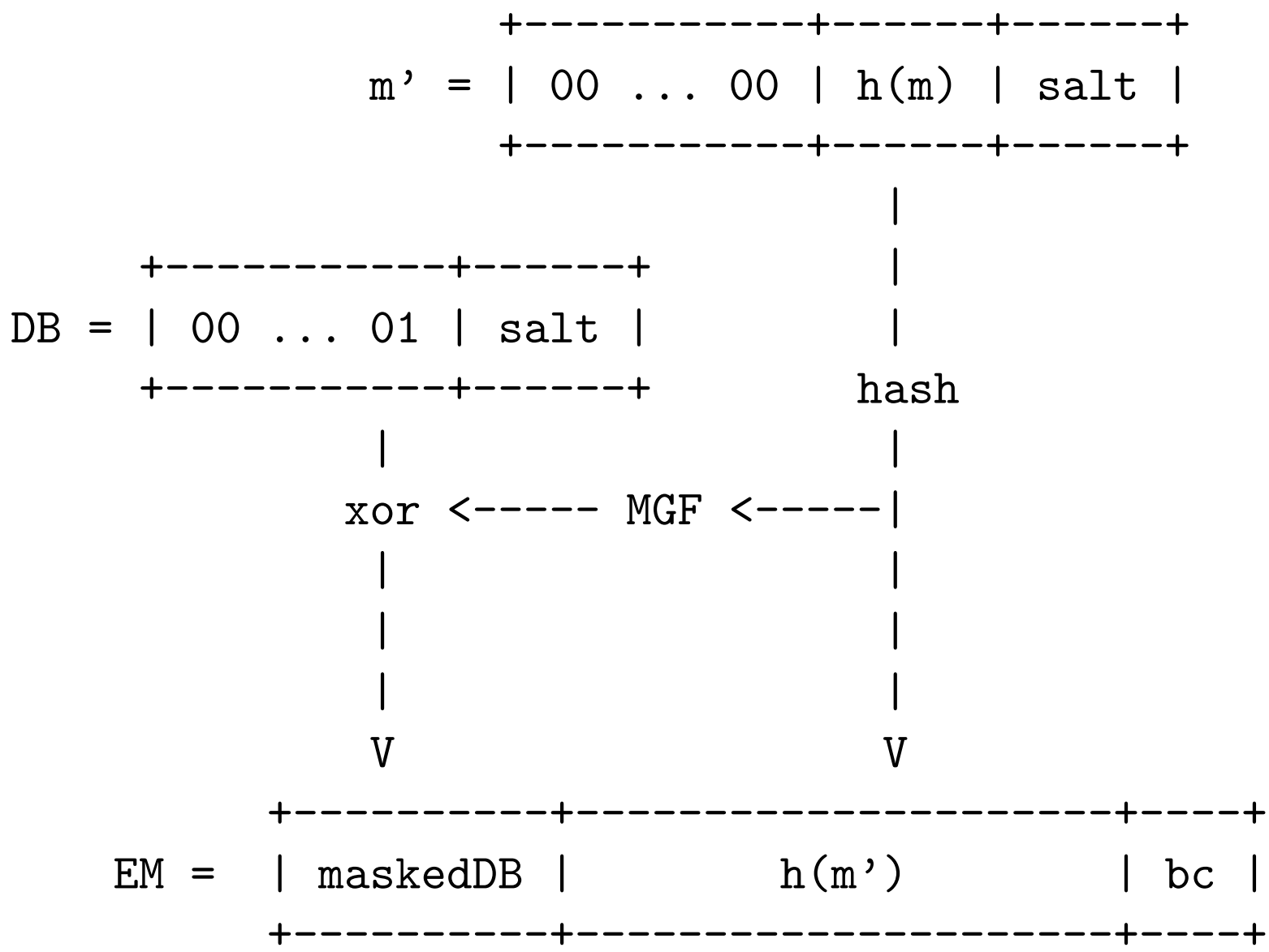
RSASSA-PSS – Details

Arguments for encoding

- m message to be signed
- h hash function
- M mask generation function
- sLen salt length (bytes), mostly hash length or 0
- L desired output length, $\geq \|h(*)\| + \text{sLen} + 2$

Decode

- split EM by length to get the parts maskedDB$'$, H'
- with H' unmask to get DB'
- know salt length, so get salt$'$
- construct $m' = 00 \ldots 00 \,||\, h(m) \,||\, \text{salt}'$
- check $H' = h(m')$,
- if yes (and all hardcoded bytes correct), accept


m' = | 00 ... 00 | h(m) | salt |
DB = | 00 ... 01 | salt |
hash
xor <----- MGF <-----
EM = | maskedDB | h(m') | bc |

Comparison of Schemes

- assuming an RSA modulus of $\|n\| \sim \log n$ bit

Comparison of Schemes

- assuming an RSA modulus of $\|n\| \sim \log n$ bit

OAEP/Encryption

- hash of label, seed, two fixes bytes
- hence $\|n\| \geq \|m\| + 2\mathsf{hLen} + 2$
- maximal payload

Comparison of Schemes

- assuming an RSA modulus of $\|n\| \sim \log n$ bit

OAEP/Encryption

- hash of label, seed, two fixes bytes
- hence $\|n\| \geq \|m\| + 2\text{hLen} + 2$
- maximal payload

RSASSA-PSS/Signing

- L desired output length, so $L = \|n\|$
- one hash, one salt, two fixed bytes
- only restriction $\|n\| \geq \text{hLen} + \text{sLen} + 2$
- no restriction on m, as hashed anyway

Why so complicated

- PKCS #1 v1.5 was easier, for encryption we have

$$00 \,||\, 02 \,||\, \text{random} \,||\, 00 \,||\, m_0$$

- broken 1998 by Bleichenbacher
- SSL 3.0 (from '96) $\rightarrow$ TLS 1.0 (in '99)

Why so complicated

- PKCS #1 v1.5 was easier, for encryption we have

$$00 \,||\, 02 \,||\, \text{random} \,||\, 00 \,||\, m_0$$

- broken 1998 by Bleichenbacher
- SSL 3.0 (from '96) $\rightarrow$ TLS 1.0 (in '99)
- we have error messages for wrong encoding $\rightarrow$ 1 bit information
- acceptance depending on the three fixed bytes
- shifting message gives information
- if server return different errors, can also exploit this

Why so complicated

- PKCS #1 v1.5 was easier, for encryption we have

$$00 \,||\, 02 \,||\, \text{random} \,||\, 00 \,||\, m_0$$

- broken 1998 by Bleichenbacher
- SSL 3.0 (from '96) $\rightarrow$ TLS 1.0 (in '99)
- we have error messages for wrong encoding $\rightarrow$ 1 bit information
- acceptance depending on the three fixed bytes
- shifting message gives information
- if server return different errors, can also exploit this
- Adaptive Chosen Cipher Attack
- in total $\sim$ 1 million messages for practical attack

Why so complicated

- PKCS #1 v1.5 was easier, for encryption we have

$$00 \,||\, 02 \,||\, \text{random} \,||\, 00 \,||\, m_0$$

- broken 1998 by Bleichenbacher
- SSL 3.0 (from '96) $\rightarrow$ TLS 1.0 (in '99)
- we have error messages for wrong encoding $\rightarrow$ 1 bit information
- acceptance depending on the three fixed bytes
- shifting message gives information
- if server return different errors, can also exploit this
- Adaptive Chosen Cipher Attack
- in total $\sim$ 1 million messages for practical attack

PKCS #1 v1.5 only for compatibility, should be avoided if possible.

skip details

Bleichenbacher's Attack – Decrypt

- let B bound on $m := (\text{random} \,||\, 00 \,||\, m_0)$
- input c, get the information whether $2B \leq c^d \bmod n \leq 3B$
- given $c_0 = m^e \bmod n$, find s_i such that $c_0(s_i)^e$ is accepted
- M_i set of intervals, one contains m, $M_0 = \{[2B, 3B]\}$
  finished if $M_* = \{[m, m]\}$, i.e. one singleton

Bleichenbacher's Attack – Decrypt

- let B bound on $m := (\text{random} \,||\, 00 \,||\, m_0)$
- input c, get the information whether $2B \leq c^d \bmod n \leq 3B$
- given $c_0 = m^e \bmod n$, find s_i such that $c_0(s_i)^e$ is accepted
- M_i set of intervals, one contains m, $M_0 = \{[2B, 3B]\}$
  finished if $M_* = \{[m, m]\}$, i.e. one singleton
- s_i is conform if $2B \leq ms_i \bmod n < 3B$
- assume $m \in [a, b]$

$$2B \leq ms_i - rn \leq 3B - 1 \qquad \text{for some } r \in \mathbb{N}$$

$$\xRightarrow{a \leq m \leq b} as_i - (3B - 1) \leq rn \leq bs_i - 2B \qquad \text{inductive bounds}$$

some candidates for r, for each

$$\frac{2B + rn}{s_i} \leq m \leq \frac{3B - 1 + rn}{s_i}$$

$i = 1$: smallest $s_1 \geq n/(3B)$ s.t. conform

$|M_{i-1}| > 1$: smallest $s_i > s_{i-1}$ s.t. conform

$|M_{i-1}| = 1$: find smallest r_i, then s_i with

$$r_i \geq 2\frac{bs_{i-1} - 2B}{n} \qquad \frac{2B + r_i n}{b} \leq s_i \leq \frac{3B + r_i n}{a}$$

$i = 1$: smallest $s_1 \geq n/(3B)$ s.t. conform

$|M_{i-1}| > 1$: smallest $s_i > s_{i-1}$ s.t. conform

$|M_{i-1}| = 1$: find smallest r_i, then s_i with

$$r_i \geq 2\frac{bs_{i-1} - 2B}{n} \qquad \frac{2B + r_i n}{b} \leq s_i \leq \frac{3B + r_i n}{a}$$

- combine both old and new bounds

$$M_i = \left\{ \left[\max\left(a, \frac{2B + rn}{s_i}\right), \min\left(b, \frac{3B - 1 + rn}{s_i}\right) \right] : [a, b] \in M_i, \frac{as_i - 3B + 1}{n} \leq r \leq \frac{bs_i - 2B}{n} \right\}$$

$i = 1$: smallest $s_1 \geq n/(3B)$ s.t. conform

$|M_{i-1}| > 1$: smallest $s_i > s_{i-1}$ s.t. conform

$|M_{i-1}| = 1$: find smallest r_i, then s_i with

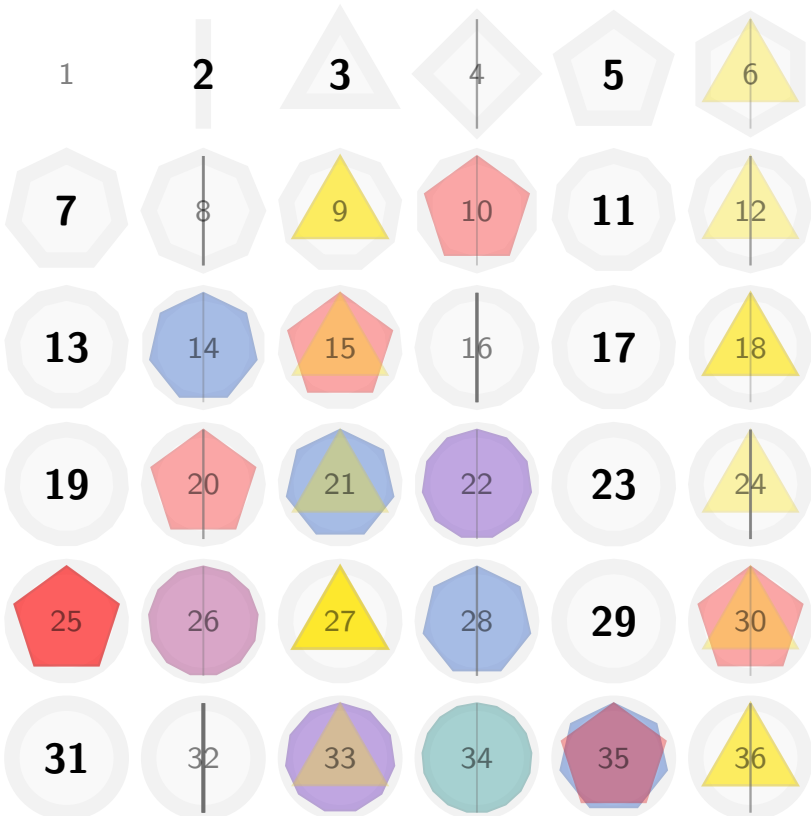
$$r_i \geq 2\frac{bs_{i-1} - 2B}{n} \qquad \frac{2B + r_i n}{b} \leq s_i \leq \frac{3B + r_i n}{a}$$

- combine both old and new bounds

$$M_i = \left\{ \left[\max\left(a, \frac{2B + rn}{s_i}\right), \min\left(b, \frac{3B - 1 + rn}{s_i}\right) \right] : [a, b] \in M_i, \frac{as_i - 3B + 1}{n} \leq r \leq \frac{bs_i - 2B}{n} \right\}$$

- probability analysis to get expected number of attempts
- experiments on 1024 bit key: between 300k and 2M
- allows practical attacks on SSL 3.0

Primality Tests



Sieve of Erathosthenes – Thanks to Todd Lehmann on texoverflow

Primality Testing

- RSA needs big primes
- earlier we suggested: create random number and check
- chance is good (Prime Number Theorem)
- but need efficient checker

Primality Testing

- RSA needs big primes
- earlier we suggested: create random number and check
- chance is good (Prime Number Theorem)
- but need efficient checker

Input: candidate $n \in \mathbb{N}$

Output: True/False

Time must be polynomial in $\|n\| \approx \log n$.

Primality Testing

- RSA needs big primes
- earlier we suggested: create random number and check
- chance is good (Prime Number Theorem)
- but need efficient checker

Input: candidate $n \in \mathbb{N}$

Output: True/False

Time must be polynomial in $\|n\| \approx \log n$.

- exact checkers are too slow, even though polynomial $\sim \|n\|^{6+\varepsilon}$
- use probabilistic method
  (chance of wrong answer $\sim$ chance of guessing key)

Naive Test

Naive Test

function PRIME(n)
    for $i = 2, \ldots, \lfloor \sqrt{n} \rfloor$ **do**
        if $n \bmod i = 0$ **then**
            return False
    return True

Naive Test

Naive Test

function PRIME(n)
  for $i = 2, \ldots, \lfloor \sqrt{n} \rfloor$ **do**
    if $n \bmod i = 0$ **then**
      return False
  return True

- obviously works correctly
- time $\mathcal{O}^*(\sqrt{n}) = \mathcal{O}^*\left(2^{\frac{1}{2}\log n}\right)$, i.e. exponential
  ($\mathcal{O}^*$ means, we leave out polynomial factors)

Fermat Test

Lemma

If p is prime, and $\gcd(a, p) = 1$, *then* $a^{p-1} \equiv 1 \mod p$.

Only implication! No "if and only if"!

Fermat Test

Lemma

If p is prime, and $\gcd(a, p) = 1$, *then* $a^{p-1} \equiv 1 \mod p$.

Only implication! No "if and only if"!

Fermat Test

- pick some random $a < n$
- optional: check $\gcd(a, n) = 1$
- check whether n satisfies this lemma

Fermat Test

Lemma

If p is prime, and $\gcd(a, p) = 1$, *then* $a^{p-1} \equiv 1 \mod p$.

Only implication! No "if and only if"!

Fermat Test

- pick some random $a < n$
- optional: check $\gcd(a, n) = 1$
- check whether n satisfies this lemma

Example

- Let $n = 97$, $a = 68$. Have $a^{n-1} \mod n = 1$, so n passes.

Fermat Test

Lemma

If p is prime, and $\gcd(a, p) = 1$, *then* $a^{p-1} \equiv 1 \mod p$.

Only implication! No "if and only if"!

Fermat Test

- pick some random $a < n$
- optional: check $\gcd(a, n) = 1$
- check whether n satisfies this lemma

Example

- Let $n = 97$, $a = 68$. Have $a^{n-1} \mod n = 1$, so n passes.
- Let $n = 91$, $a = 23$ yields $a^{n-1} \mod n = 1$, so n passes.

Fermat Test

Lemma

If p is prime, and $\gcd(a, p) = 1$, *then* $a^{p-1} \equiv 1 \mod p$.

Only implication! No "if and only if"!

Fermat Test

- pick some random $a < n$
- optional: check $\gcd(a, n) = 1$
- check whether n satisfies this lemma

Example

- Let $n = 97$, $a = 68$. Have $a^{n-1} \mod n = 1$, so n passes.
- Let $n = 91$, $a = 23$ yields $a^{n-1} \mod n = 1$, so n passes.
  But $a = 19 \implies a^{n-1} \mod n = 64 \implies n$ is not prime.

Fermat Test

Lemma

If p is prime, and $\gcd(a, p) = 1$, *then* $a^{p-1} \equiv 1 \mod p$.

Only implication! No "if and only if"!

Fermat Test

- pick some random $a < n$
- optional: check $\gcd(a, n) = 1$
- check whether n satisfies this lemma

Example

- Let $n = 97$, $a = 68$. Have $a^{n-1} \mod n = 1$, so n passes.
- Let $n = 91$, $a = 23$ yields $a^{n-1} \mod n = 1$, so n passes.
  But $a = 19 \implies a^{n-1} \mod n = 64 \implies n$ is not prime.
- Let $n = 561$, then n passes for every a, but $n = 3 \cdot 11 \cdot 17$.

Fermat Test – Analysis

- quick: need $\mathcal{O}(\|n\|)$ arithmetic operations per run,
- run t times, with different a
- correct answer if n prime
- may give false answer for composite number

Fermat Test – Analysis

- quick: need $\mathcal{O}(\|n\|)$ arithmetic operations per run,
- run t times, with different a
- correct answer if n prime
- may give false answer for composite number

Definition

A Carmichael number is a composite number n, that passes the Fermat test for every base a coprime to n.

561 is smallest Carmichael number.

Fermat Test – Analysis

- quick: need $\mathcal{O}(\|n\|)$ arithmetic operations per run,
- run t times, with different a
- correct answer if n prime
- may give false answer for composite number

Definition

A Carmichael number is a composite number n, that passes the Fermat test for every base a coprime to n.

561 is smallest Carmichael number.

Lemma (Alford, Granville, Pomerance; 1994)

There are infinitely many Carmichael numbers.

Fermat Test – Analysis

- quick: need $\mathcal{O}(\|n\|)$ arithmetic operations per run,
- run t times, with different a
- correct answer if n prime
- may give false answer for composite number

Definition

A Carmichael number is a composite number n, that passes the Fermat test for every base a coprime to n.

561 is smallest Carmichael number.

Lemma (Alford, Granville, Pomerance; 1994)

There are infinitely many Carmichael numbers.

Fermat test has no success guarantee > 0.

Miller-Rabin Test

- Developed by Artjuhov ('67), Miller ('76'), Rabin ('80).
- removes problem of Carmichael numbers

Miller-Rabin Test

- Developed by Artjuhov ('67), Miller ('76'), Rabin ('80).
- removes problem of Carmichael numbers

Idea

- if n prime, then $x^2 \equiv 1 \mod n$ only has solutions $x = \pm 1$
- in Fermat a^{n-1} is an even power
- taking roots, we should arrive at -1
- for odd powers, we cannot compute root
  (find root equivalent to factoring),
  so we must stop

Miller-Rabin Test

Miller-Rabin-Test

function MILLER-RABIN(n)

pick random $a < n$

if $\gcd(a, n) \neq 1$ **then return** False

write $n - 1 = 2^s \cdot k$, for k odd

if $a^k \equiv 1 \mod n$ **then return** True

for $i = 0, \ldots, s - 1$ **do**

if $(a^k)^{2^i} \equiv -1 \mod n$ **then return** True

return False

Miller-Rabin Test

Miller-Rabin-Test

function MILLER-RABIN(n)
  pick random $a < n$
  if $\gcd(a, n) \neq 1$ **then return** False
  write $n - 1 = 2^s \cdot k$, for k odd
  if $a^k \equiv 1 \mod n$ **then return** True
  for $i = 0, \ldots, s - 1$ **do**
    if $(a^k)^{2^i} \equiv -1 \mod n$ **then return** True
  return False

- If n not prime, $\leq \frac{\varphi(n)}{4}$ choices of base a give false answer
- run test t times, takes $\mathcal{O}(t \cdot \|n\|)$ arithmetic operations
- reject n if one iteration fails
- $\rightsquigarrow$ error chance $\leq \left(\frac{1}{4}\right)^t$

Example Miller-Rabin

Example

- Let $n = 91$
- write $n - 1 = 90 = 45 \cdot 2^1$

Example Miller-Rabin

Example

- Let $n = 91$
- write $n - 1 = 90 = 45 \cdot 2^1$
- pick base $a = 38$
  - $a^{45} \equiv 90 \equiv -1 \mod 91$
  - hence accepted (for now)

Example Miller-Rabin

Example

- Let $n = 91$
- write $n - 1 = 90 = 45 \cdot 2^1$
- pick base $a = 38$
  - $a^{45} \equiv 90 \equiv -1 \mod 91$
  - hence accepted (for now)
- pick base $a = 23$
  - $a^{45} \mod 91 = 64 \neq \pm 1$
  - $\left(a^{45}\right)^2 \mod 91 = 1$
  - hence composite

Prime and Prejudice (2018)

- usually, testing few small numbers suffices
- many implementation fix(ed?) t or bases a_i
- but error chance needs randomness
- in crypto always consider adversarial input

Prime and Prejudice (2018)

- usually, testing few small numbers suffices
- many implementation fix(ed?) t or bases a_i
- but error chance needs randomness
- in crypto always consider adversarial input

Failure Chance Against Adversary

OpenSSL 1.1.1-pre6 fix $t = 2$ for $\log n \geq 1300$,
failure chance $\frac{1}{16}$

GNU GMP bases a_i depend deterministically on n,
100% failure for $t \leq 15$

LibTomMath $t \leq 256$, use first t primes as bases,
100% failure

Creating Adversarial Input

Counting false witnesses

- let $S(n)$ be how many bases pass test for (composite) n
- so far, we have upper bound $S(n) \leq \frac{\varphi(n)}{4}$ for false witnesses
- Can we reach this bound?

Creating Adversarial Input

Counting false witnesses

- let $S(n)$ be how many bases pass test for (composite) n
- so far, we have upper bound $S(n) \leq \frac{\varphi(n)}{4}$ for false witnesses
- Can we reach this bound?

Theorem (Monier, '80)

Assume we write

$$n = 2^s \cdot k + 1 = \prod_{i=1}^{m} p_i^{e_i}$$

with primes $p_i = 2^{s_i} k_i + 1$ *and* k, k_i *odd. Then*

$$S(n) = \left(\prod \gcd(k, k_i)\right) \left(\frac{2^{\min(s_i)\cdot m} - 1}{2^m - 1} + 1\right)$$

$$n = 2^s \cdot k + 1 = \prod_{i=1}^{m} p_i^{e_i} \qquad p_i = 2^{s_i} k_i + 1$$

$$S(n) = \left(\prod \gcd(k, k_i) \right) \left(\frac{2^{\min(s_i) \cdot m} - 1}{2^m - 1} + 1 \right)$$

Corollary

Let x odd with with $2x + 1$ and $4x + 1$ prime. Then $n = (2x + 1)(4x + 1)$ achieves the worst error chance for Miller-Rabin.

$$n = 2^s \cdot k + 1 = \prod_{i=1}^{m} p_i^{e_i} \qquad p_i = 2^{s_i} k_i + 1$$

$$S(n) = \left(\prod \gcd(k, k_i) \right) \left(\frac{2^{\min(s_i) \cdot m} - 1}{2^m - 1} + 1 \right)$$

Corollary

Let x odd with with $2x+1$ and $4x+1$ prime. Then $n = (2x+1)(4x+1)$ achieves the worst error chance for Miller-Rabin.

apply formula.

- $p_1 = 2x+1$, $p_2 = 4x+1$, so $k_1 = k_2 = x$, $s_1 = 1$, $s_2 = 2$
- $n = 8x^2 + 6x + 1$, so $s = 1$ and $k = 4x^2 + 3x$
- $\gcd(k, k_i) = x$, hence $S(n) = 2x^2 = \frac{\varphi(n)}{4}$ □

$$n = 2^s \cdot k + 1 = \prod_{i=1}^{m} p_i^{e_i} \qquad p_i = 2^{s_i} k_i + 1$$

$$S(n) = \left(\prod \gcd(k, k_i)\right) \left(\frac{2^{\min(s_i)\cdot m} - 1}{2^m - 1} + 1\right)$$

Corollary

Let x odd with with $2x+1$ and $4x+1$ prime. Then $n = (2x+1)(4x+1)$ achieves the worst error chance for Miller-Rabin.

apply formula.

- $p_1 = 2x+1$, $p_2 = 4x+1$, so $k_1 = k_2 = x$, $s_1 = 1$, $s_2 = 2$
- $n = 8x^2 + 6x + 1$, so $s = 1$ and $k = 4x^2 + 3x$
- $\gcd(k, k_i) = x$, hence $S(n) = 2x^2 = \frac{\varphi(n)}{4}$ □

Construction: guess x and check primality

Miller-Rabin – Wrap-Up

Do's

- stick to the pseudo-code
- use random bases
- t rounds give $2t$ bit security level

Miller-Rabin – Wrap-Up

Do's

- stick to the pseudo-code
- use random bases
- t rounds give $2t$ bit security level

Don't's

- small number of rounds t: can efficiently create adversarial input
- fixed bases: can create input with guaranteed false answer, procedure more involved, but feasible

AKS Primality Test

- Developed by Agrawal, Kayal, Saxena
- famous paper “Primes is in P”, 2002
- first provable polynomial time algorithm

AKS Primality Test

- Developed by Agrawal, Kayal, Saxena
- famous paper "Primes is in P", 2002
- first provable polynomial time algorithm

Idea for AKS

- Let $a, n \in \mathbb{N}$. We have $(x+a)^n \equiv x^n + a^n$ in $\mathbb{Z}_n[x]$ iff n is prime.
- reduce this modulo smaller polynomial
- give bound on values a to check

AKS Primality Test

- Developed by Agrawal, Kayal, Saxena
- famous paper "Primes is in P", 2002
- first provable polynomial time algorithm

Idea for AKS

- Let $a, n \in \mathbb{N}$. We have $(x+a)^n \equiv x^n + a^n$ in $\mathbb{Z}_n[x]$ iff n is prime.
- reduce this modulo smaller polynomial
- give bound on values a to check

- Takes time $\mathcal{O}\left(\|n\|^{6+\varepsilon}\right)$, too much

Sieve of Erathosthenes

- not really a primality test
- good method to generate all primes $p \leq n$

function ERATHOSTHENES(n)
  create array $a_i = 1$ for $i \leq n$
  $i \leftarrow 2$
  while $i^2 \leq n$ **do**
    if $a_i = 1$ **then**
      for $j = 2, \ldots, \lfloor n/i \rfloor$ **do**
        $a_{i \cdot j} \leftarrow 0$
  return $\{p : a_p = 1\}$

Sieve of Erathosthenes

- not really a primality test
- good method to generate all primes $p \leq n$

function ERATHOSTHENES(n)
  create array $a_i = 1$ for $i \leq n$
  $i \leftarrow 2$
  while $i^2 \leq n$ **do**
    if $a_i = 1$ **then**
      for $j = 2, \ldots, \lfloor n/i \rfloor$ **do**
        $a_{i \cdot j} \leftarrow 0$
  return $\{p : a_p = 1\}$

- on PC, for $n = 2^{30}$ about 6 seconds (with some optimisations)
- running time $\mathcal{O}(n \log \log n)$, space $\mathcal{O}(n)$
- only use, if array fits into RAM!

Primality Tests – Overview

Fermat: easy, fast, can have one-sided errors, fails for some numbers

Miller-Rabin: Method of choice

- as fast as Fermat,
- also one-sided error
- repeated, independent calls: error $\searrow 0$
- in most crypto-libraries,
  but many implementations were (are?) vulnerable to
  malicious input $\rightsquigarrow$ "Prime and Prejudice"

AKS: no error, but long running time

Erathosthenes: no test, but a method to generate all primes $\leq n$
only recommended for $n <$ RAM

Prime Generation – Revisited

Optimisation

- chance of primality is $\sim 1/\log n$
- improve factor by ruling out small primes as divisors
  e.g. prime odd, better $p = 6 \cdot k \pm 1, \ldots$

Prime Generation – Revisited

Optimisation

- chance of primality is $\sim 1/\log n$
- improve factor by ruling out small primes as divisors
  e.g. prime odd, better $p = 6 \cdot k \pm 1, \ldots$

How not to do it:

RSAlib by Infineon

- enumerate primes p_i for $i = 1, 2, \ldots$
- put $M := \prod_{i \leq s} p_i$ (primorial),
  for $s = 39, 71, 126, 225$, depending on key-size
- choose random k, a and put $p = kM + (65537^a \bmod M)$

Prime Generation – Revisited

Optimisation

- chance of primality is $\sim 1/\log n$
- improve factor by ruling out small primes as divisors
  e.g. prime odd, better $p = 6 \cdot k \pm 1, \ldots$

How not to do it:

RSAlib by Infineon

- enumerate primes p_i for $i = 1, 2, \ldots$
- put $M := \prod_{i \leq s} p_i$ (primorial),
  for $s = 39, 71, 126, 225$, depending on key-size
- choose random k, a and put $p = kM + (65537^a \bmod M)$
- p not divisible by any of the small primes
- increase the chance of p to be prime

RSAlib – First Concern

Numbers for 2048 Bit Key – 1024 Bit Prime

- $s = 126$, so $M = 2 \cdot 3 \cdot \ldots \cdot 701 \sim 971$ Bit
- leaves only $k \sim 53$ Bit
- $\varphi(M) \sim 968$ Bit, but $o_{\mathbb{Z}_M^*}(65537) \sim 255$ Bit,
- i.e. 713 Bits entropy lost!

RSAlib – First Concern

Numbers for 2048 Bit Key – 1024 Bit Prime

- $s = 126$, so $M = 2 \cdot 3 \cdot \ldots \cdot 701 \sim 971$ Bit
- leaves only $k \sim 53$ Bit
- $\varphi(M) \sim 968$ Bit, but $o_{\mathbb{Z}_M^*}(65537) \sim 255$ Bit,
- i.e. 713 Bits entropy lost!

bit-size key	# primes	entropy	bit lost
512 - 960	39	62	154
992 - 1952	71	134	338
1984 - 3936	126	255	713
3968 - 4096	225	434	1525

- Calculation: separately for every prime, then lcm

RSAlib – First Concern

Numbers for 2048 Bit Key – 1024 Bit Prime

- $s = 126$, so $M = 2 \cdot 3 \cdot \ldots \cdot 701 \sim 971$ Bit
- leaves only $k \sim 53$ Bit
- $\varphi(M) \sim 968$ Bit, but $o_{\mathbb{Z}_M^*}(65537) \sim 255$ Bit,
- i.e. 713 Bits entropy lost!

bit-size key	# primes	entropy	bit lost
512 - 960	39	62	154
992 - 1952	71	134	338
1984 - 3936	126	255	713
3968 - 4096	225	434	1525

- Calculation: separately for every prime, then lcm

but it gets worse

ROCA – Return of Coppersmith Attack

Theorem (Coppersmith)

Let $p = \sum a_{ij} x^i y^j \in \mathbb{Z}[x, y]$ irreducible; X, Y bounds for solutions. Put $W := \max\{|a_{ij}| \cdot X^i Y^j : i, j\}$ and $\delta = \max(\deg_x(p), \deg_y(p))$. Assume $XY < W^{2/(3\delta)}$. Then we can find integer root (x_0, y_0) with $|x_0| < X$, $|y_0| < Y$, if it exists.

Find "small" integer roots in 2 variables.

ROCA – Return of Coppersmith Attack

Theorem (Coppersmith)

Let $p = \sum a_{ij} x^i y^j \in \mathbb{Z}[x, y]$ irreducible; X, Y bounds for solutions. Put $W := \max\{|a_{ij}| \cdot X^i Y^j : i, j\}$ and $\delta = \max(\deg_x(p), \deg_y(p))$. Assume $XY < W^{2/(3\delta)}$. Then we can find integer root (x_0, y_0) with $|x_0| < X$, $|y_0| < Y$, if it exists.

Find "small" integer roots in 2 variables.

$$p = kM + (65537^a \bmod M) \qquad q = \ell M + \left(65537^b \bmod M\right)$$

$$n = pq = * \cdot M + \left(65537^{a+b} \bmod M\right)$$

ROCA – Return of Coppersmith Attack

Theorem (Coppersmith)

Let $p = \sum a_{ij} x^i y^j \in \mathbb{Z}[x, y]$ irreducible; X, Y bounds for solutions. Put $W := \max\{|a_{ij}| \cdot X^i Y^j : i, j\}$ and $\delta = \max(\deg_x(p), \deg_y(p))$. Assume $XY < W^{2/(3\delta)}$. Then we can find integer root (x_0, y_0) with $|x_0| < X$, $|y_0| < Y$, if it exists.

Find "small" integer roots in 2 variables.

$$p = kM + (65537^a \bmod M) \qquad q = \ell M + \left(65537^b \bmod M\right)$$

$$n = pq = * \cdot M + \left(65537^{a+b} \bmod M\right)$$

- $a + b = \log_{65537}(n \bmod M)$,
  easy to compute, since M has only small primes
  (compute for each prime, compose with CRT)
- guess a, yields b, then compute k, ℓ via Coppersmith

ROCA – Application

Assume 2048 bit key $\rightsquigarrow$ 1024 bit prime

$$s = 126 \qquad M \sim 971 \text{ bit} \qquad k < 2^{53}$$

ROCA – Application

Assume 2048 bit key $\rightsquigarrow$ 1024 bit prime

$$s = 126 \qquad M \sim 971 \text{ bit} \qquad k < 2^{53}$$

For some given guess of a (and b) we have

$$A := 65537^a \bmod M \qquad B := 65537^b \bmod M$$

$$p(x, y) = Mxy + Ay + Bx + (AB - n)/M$$

ROCA – Application

Assume 2048 bit key $\rightsquigarrow$ 1024 bit prime

$$s = 126 \qquad M \sim 971 \text{ bit} \qquad k < 2^{53}$$

For some given guess of a (and b) we have

$$A := 65537^a \bmod M \qquad B := 65537^b \bmod M$$

$$p(x, y) = Mxy + Ay + Bx + (AB - n)/M$$

$$\delta = 1 \qquad X = Y = 2^{53}$$

$$W = \max(MXY, AY, BX, (AB - n)/M) = MXY$$

ROCA – Application

Assume 2048 bit key $\rightsquigarrow$ 1024 bit prime

$$s = 126 \qquad M \sim 971 \text{ bit} \qquad k < 2^{53}$$

For some given guess of a (and b) we have

$$A := 65537^a \bmod M \qquad B := 65537^b \bmod M$$

$$p(x, y) = Mxy + Ay + Bx + (AB - n)/M$$

$$\delta = 1 \qquad X = Y = 2^{53}$$

$$W = \max(MXY, AY, BX, (AB - n)/M) = MXY$$

Then check $XY < W^{2/3}$, which holds (by far).
Hence, Coppersmith finds solution k, ℓ, i.e. the primes

Refining ROCA

Problem

guessing a still has ~ 255 bits in our example $\rightsquigarrow$ too much

Refining ROCA

Problem

guessing a still has ~ 255 bits in our example $\rightsquigarrow$ too much

- But we are far from the limit in Coppersmith.

need: $XY < W^{2/3}$ have: $W = MXY$

Refining ROCA

Problem

guessing a still has ~ 255 bits in our example $\rightsquigarrow$ too much

- But we are far from the limit in Coppersmith.

$$\text{need: } XY < W^{2/3} \qquad \text{have: } W = MXY$$

- use divisor $M' \mid M$,
  sufficient to have $M' > X = Y$
  note: X, Y increase for smaller M'

Refining ROCA

Problem

guessing a still has ~ 255 bits in our example $\rightsquigarrow$ too much

- But we are far from the limit in Coppersmith.

$$\text{need: } XY < W^{2/3} \qquad \text{have: } W = MXY$$

- use divisor $M' \mid M$,
  sufficient to have $M' > X = Y$
  note: X, Y increase for smaller M'
- less unknown bits for a, but more unknown bits for k, ℓ
- Coppersmith takes longer, but much less attempts

Refining ROCA

Problem

guessing a still has ~ 255 bits in our example $\rightsquigarrow$ too much

- But we are far from the limit in Coppersmith.

$$\text{need: } XY < W^{2/3} \qquad \text{have: } W = MXY$$

- use divisor $M' \mid M$,
  sufficient to have $M' > X = Y$
  note: X, Y increase for smaller M'
- less unknown bits for a, but more unknown bits for k, ℓ
- Coppersmith takes longer, but much less attempts
- for each key-length find optimal trade-off
- 2048 bit takes ca. 35 CPU-years, cost ~ 1240 € (rough guess)
  $\rightsquigarrow$ feasible for private person
- easily in parallel

Fix ROCA

- random number $p \leq 2^B$ is prime with probability $1/(B \ln 2)$
- put last bit 1, make sure $2 \nmid p$
  double chance of success

Fix ROCA

- random number $p \leq 2^B$ is prime with probability $1/(B \ln 2)$
- put last bit 1, make sure $2 \nmid p$
  double chance of success
- generalise idea for other primes:
  generate p that is for sure not divisible by $2, 3, 5, 7, 11, \ldots, p_s$
- create remainders $a_i < p_i$, use CRT on

$$x \equiv a_i \mod p_i \qquad \text{for } i = 1, 2, \ldots, s$$

- increasing $s \rightsquigarrow$ higher chance for prime, but also more time

Fix ROCA

- random number $p \leq 2^B$ is prime with probability $1/(B \ln 2)$
- put last bit 1, make sure $2 \nmid p$
  double chance of success
- generalise idea for other primes:
  generate p that is for sure not divisible by $2, 3, 5, 7, 11, \ldots, p_s$
- create remainders $a_i < p_i$, use CRT on

$$x \equiv a_i \mod p_i \qquad \text{for } i = 1, 2, \ldots, s$$

- increasing $s \rightsquigarrow$ higher chance for prime, but also more time

Exercise

Analyse Effort and speedup of this idea: theory and practice
Warning: don't have too much hope

Theory

- as before use primorial $M := \prod_{i \leq s} p_i$
- prime candidate $kM + a$ where a is solution of CRT
- thus $a \in \mathbb{Z}_M^*$ random, instead of random from $\mathbb{Z}_M$
- increase chance by factor $M/\varphi(M)$, practically ≤ 12

$$\frac{M}{\varphi(M)} = \prod \frac{p_i}{p_i - 1} = \prod \left(1 + \frac{1}{p_i - 1}\right) > \sum \frac{1}{p_i} \sim \ln \ln p_s$$

Theory

- as before use primorial $M := \prod_{i \leq s} p_i$
- prime candidate $kM + a$ where a is solution of CRT
- thus $a \in \mathbb{Z}_M^*$ random, instead of random from $\mathbb{Z}_M$
- increase chance by factor $M/\varphi(M)$, practically ≤ 12

$$\frac{M}{\varphi(M)} = \prod \frac{p_i}{p_i - 1} = \prod \left(1 + \frac{1}{p_i - 1}\right) > \sum \frac{1}{p_i} \sim \ln \ln p_s$$

Practice

- begin primality test with trial division
- anything divisible by small p_i ruled out quickly
- long part is Miller-Rabin on actual prime
- less random bits, but barely any speed gain

Theory

- as before use primorial $M := \prod_{i \leq s} p_i$
- prime candidate $kM + a$ where a is solution of CRT
- thus $a \in \mathbb{Z}_M^*$ random, instead of random from $\mathbb{Z}_M$
- increase chance by factor $M/\varphi(M)$, practically ≤ 12

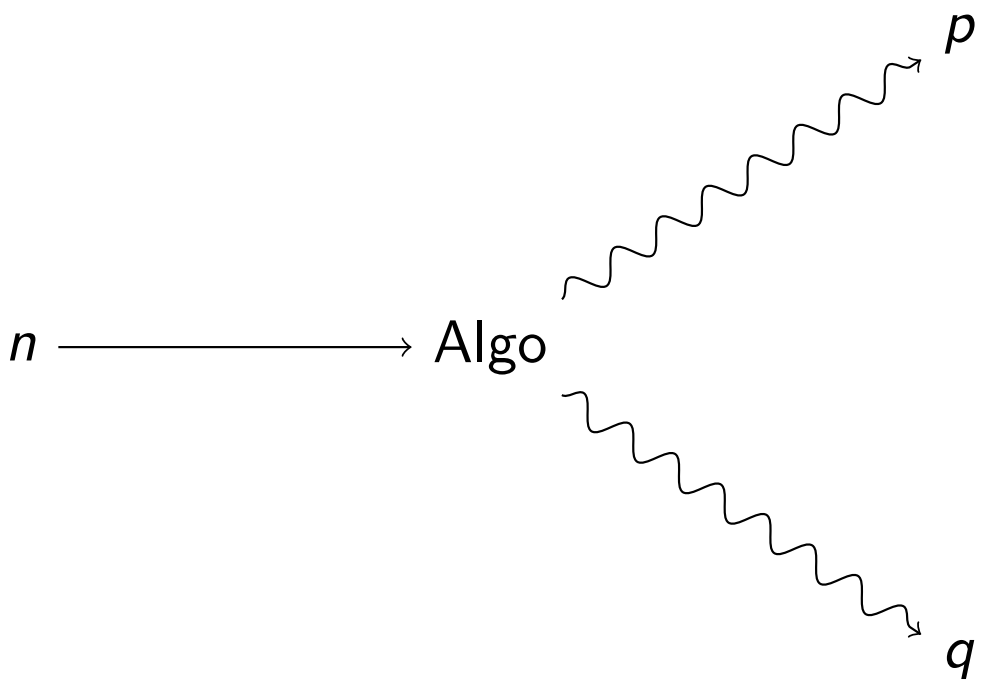
$$\frac{M}{\varphi(M)} = \prod \frac{p_i}{p_i - 1} = \prod \left(1 + \frac{1}{p_i - 1}\right) > \sum \frac{1}{p_i} \sim \ln \ln p_s$$

Practice

- begin primality test with trial division
- anything divisible by small p_i ruled out quickly
- long part is Miller-Rabin on actual prime
- less random bits, but barely any speed gain

$\rightsquigarrow$ picking odd random number works well enough to find prime

Factoring



The Factorisation Problem

Task

Given $n \in \mathbb{N}$, find a non-trivial divisor d of n.

The Factorisation Problem

Task

Given $n \in \mathbb{N}$, find a non-trivial divisor d of n.

- prefer decision problem; suggestions?

The Factorisation Problem

Task

Given $n \in \mathbb{N}$, find a non-trivial divisor d of n.

- prefer decision problem; suggestions?
- whether such d exists is primality testing
- have to adjust decision version

The Factorisation Problem

Task

Given $n \in \mathbb{N}$, find a non-trivial divisor d of n.

- prefer decision problem; suggestions?
- whether such d exists is primality testing
- have to adjust decision version

Decision problem

Given $n, U \in \mathbb{N}$, does n have a prime divisor p with $p \leq U$?

- Factoring is neither known to be in P nor known to be NP-complete.
- problem lies in NP $\cap$ coNP, hence most likely not NP-complete
  - prime factor $p \leq U$ serves as witness
  - factorisation with all $p_i > U$ serves as non-witness

Fermat Factorisation

Method

Let $n = pq$. If p, q (not necessarily prime) are close, we can factor n.

Fermat Factorisation

Method

Let $n = pq$. If p, q (not necessarily prime) are close, we can factor n.

- Key Observation

$$n = pq = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2$$

Fermat Factorisation

Method

Let $n = pq$. If p, q (not necessarily prime) are close, we can factor n.

- Key Observation

$$n = pq = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2$$

Pseudocode

$m \leftarrow \lceil \sqrt{n} \rceil$
for $i \in \mathbb{N}$ **do**
    $\Delta_i = \sqrt{(m+i)^2 - n}$
    if $\Delta_i \in \mathbb{N}$ **then**
        return $p \leftarrow m + i - \Delta_i$

Fermat Factorisation – Analysis

$$\Delta_i = \sqrt{(\lceil \sqrt{n} \rceil + i)^2 - n}$$

$$n = pq = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2$$

- wlog $p > q$, then $\Delta_i = \frac{p-q}{2}$ in the end
- reach this when $m + i = \frac{p+q}{2}$,

Fermat Factorisation – Analysis

$$\Delta_i = \sqrt{(\lceil \sqrt{n} \rceil + i)^2 - n}$$

$$n = pq = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2$$

- wlog $p > q$, then $\Delta_i = \frac{p-q}{2}$ in the end
- reach this when $m + i = \frac{p+q}{2}$, i.e.

$$i \approx \frac{p+q}{2} - \sqrt{n} = \frac{\left(\sqrt{p} - \sqrt{q}\right)^2 + 2\sqrt{pq}}{2} - \sqrt{n}$$

$$= \frac{\left(\sqrt{p} - \sqrt{q}\right)^2}{2} = \frac{\left(\sqrt{p} - \sqrt{q}\right)^2 \cdot \sqrt{q}^2}{2q} = \frac{\left(\sqrt{n} - q\right)^2}{2q}$$

Fermat Factorisation – Analysis

$$\Delta_i = \sqrt{(\lceil \sqrt{n} \rceil + i)^2 - n}$$

$$n = pq = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2$$

- wlog $p > q$, then $\Delta_i = \frac{p-q}{2}$ in the end
- reach this when $m + i = \frac{p+q}{2}$, i.e.

$$i \approx \frac{p+q}{2} - \sqrt{n} = \frac{\left(\sqrt{p} - \sqrt{q}\right)^2 + 2\sqrt{pq}}{2} - \sqrt{n}$$

$$= \frac{\left(\sqrt{p} - \sqrt{q}\right)^2}{2} = \frac{\left(\sqrt{p} - \sqrt{q}\right)^2 \cdot \sqrt{q}^2}{2q} = \frac{\left(\sqrt{n} - q\right)^2}{2q}$$

- in total: works well if p, q nearly same in upper half

Fermat Factorisation – Example

Example

Let $n = 583$, thus $m = 25$

$$
\begin{aligned}
i &= 0 & \Delta_i^2 &= 2 \cdot 3 \cdot 7 \\
i &= 1 & \Delta_i^2 &= 3 \cdot 31 \\
i &= 2 & \Delta_i^2 &= 2 \cdot 73 \\
i &= 3 & \Delta_i^2 &= 3 \cdot 67 \\
i &= 4 & \Delta_i^2 &= 2 \cdot 3 \cdot 43 \\
i &= 5 & \Delta_i^2 &= 317 \\
i &= 6 & \Delta_i^2 &= 2 \cdot 3^3 \cdot 7 \\
i &= 7 & \Delta_i^2 &= 3^2 \cdot 7^2
\end{aligned}
$$

Fermat Factorisation – Example

Example

Let $n = 583$, thus $m = 25$

$$
\begin{aligned}
i &= 0 & \Delta_i^2 &= 2 \cdot 3 \cdot 7 \\
i &= 1 & \Delta_i^2 &= 3 \cdot 31 \\
i &= 2 & \Delta_i^2 &= 2 \cdot 73 \\
i &= 3 & \Delta_i^2 &= 3 \cdot 67 \\
i &= 4 & \Delta_i^2 &= 2 \cdot 3 \cdot 43 \\
i &= 5 & \Delta_i^2 &= 317 \\
i &= 6 & \Delta_i^2 &= 2 \cdot 3^3 \cdot 7 \\
i &= 7 & \Delta_i^2 &= 3^2 \cdot 7^2
\end{aligned}
$$

$\Delta_7 = 21 \rightsquigarrow p = 25 + 7 - 21 = 11$ and $q = 25 + 7 + 21 = 53$

Quadratic Sieve

Idea

- construct $a^2 \equiv b^2 \bmod n$ from steps of Fermat-factorisation
  i.e. $a^2 - b^2 = k \cdot n$ instead of $a^2 - b^2 = n$
- take them as collection of congruences
- combine some, to get squares on both sides
- $\gcd(a \pm b, n)$ divisor with probability $\geq \frac{1}{2}$

Quadratic Sieve

Idea

- construct $a^2 \equiv b^2 \bmod n$ from steps of Fermat-factorisation
  i.e. $a^2 - b^2 = k \cdot n$ instead of $a^2 - b^2 = n$
- take them as collection of congruences
- combine some, to get squares on both sides
- $\gcd(a \pm b, n)$ divisor with probability $\geq \frac{1}{2}$

- details, "why" it works, too complicated for this lecture
  but "how" is okay

Quadratic Sieve

Idea

- construct $a^2 \equiv b^2 \bmod n$ from steps of Fermat-factorisation
  i.e. $a^2 - b^2 = k \cdot n$ instead of $a^2 - b^2 = n$
- take them as collection of congruences
- combine some, to get squares on both sides
- $\gcd(a \pm b, n)$ divisor with probability $\geq \frac{1}{2}$

- details, "why" it works, too complicated for this lecture
  but "how" is okay
- good for up to 100 decimal digits
- used for RSA-129 (from 1977), solved in 1994
- running time

$$\mathcal{O} * \left(\exp \left(\sqrt{\log n \cdot \log \log n} \right) \right)$$

Quadratic Sieve – Example

Example

Let $n = 583 = 11 \cdot 53$, use Fermat

$$i = 0 \qquad m + i = 25 \qquad \Delta_i^2 = 2 \cdot 3 \cdot 7$$

$$i = 6 \qquad m + i = 31 \qquad \Delta_i^2 = 2 \cdot 3^3 \cdot 7$$

Quadratic Sieve – Example

Example

Let $n = 583 = 11 \cdot 53$, use Fermat

$$i = 0 \qquad m + i = 25 \qquad \Delta_i^2 = 2 \cdot 3 \cdot 7$$
$$i = 6 \qquad m + i = 31 \qquad \Delta_i^2 = 2 \cdot 3^3 \cdot 7$$

in Fermat factorisation, we have

$$\Delta_i^2 = (m + i)^2 - n$$

Quadratic Sieve – Example

Example

Let $n = 583 = 11 \cdot 53$, use Fermat

$$i = 0 \qquad m + i = 25 \qquad \Delta_i^2 = 2 \cdot 3 \cdot 7$$
$$i = 6 \qquad m + i = 31 \qquad \Delta_i^2 = 2 \cdot 3^3 \cdot 7$$

in Fermat factorisation, we have

$$\Delta_i^2 = (m + i)^2 - n$$

combine $i = 0$ and $i = 6$ to get square

$$(25 \cdot 31)^2 \equiv (2 \cdot 3^2 \cdot 7)^2 \mod 583$$

Quadratic Sieve – Example

Example

Let $n = 583 = 11 \cdot 53$, use Fermat

$$i = 0 \qquad m + i = 25 \qquad \Delta_i^2 = 2 \cdot 3 \cdot 7$$
$$i = 6 \qquad m + i = 31 \qquad \Delta_i^2 = 2 \cdot 3^3 \cdot 7$$

in Fermat factorisation, we have

$$\Delta_i^2 = (m + i)^2 - n$$

combine $i = 0$ and $i = 6$ to get square

$$(25 \cdot 31)^2 \equiv (2 \cdot 3^2 \cdot 7)^2 \mod 583$$

obtain $\gcd(25 \cdot 31 - 2 \cdot 9 \cdot 7, 583) = 11$

Quadratic Sieve – Digging Deeper

Big Question

How do we know which values to combine?

Quadratic Sieve – Digging Deeper

Big Question

How do we know which values to combine?

- compute new step, and try with every previous step
  quadratic in number of steps

Quadratic Sieve – Digging Deeper

Big Question

How do we know which values to combine?

- compute new step, and try with every previous step
  quadratic in number of steps
- but we may also combine 3 or more
  $\rightsquigarrow$ exponential number of steps

Quadratic Sieve – Digging Deeper

Big Question

How do we know which values to combine?

- compute new step, and try with every previous step
  quadratic in number of steps
- but we may also combine 3 or more
  $\rightsquigarrow$ exponential number of steps

Solution: factor into small primes

- more general approach
- try to factor the Δ_i into small primes
- regard exponent modulo 2 (square or not)
- solve linear equation system modulo 2 to find combination

Factoring Through p_k-smooth Numbers

Let p_k be the k-th prime.

Definition

An integer is p_k-smooth, if all its prime divisors are $\leq p_k$.

- use Sieve of Erathosthenes!

Factoring Through p_k-smooth Numbers

Let p_k be the k-th prime.

Definition

An integer is p_k-smooth, if all its prime divisors are $\leq p_k$.

- use Sieve of Erathosthenes!

Idea (Morrison & Brillhart '75; Dixon '81)

- search for numbers a such that ($a^2 \bmod n$) is p_k-smooth
- construct x, y with $x^2 \equiv y^2 \mod n$
- construct some number that has a common divisor with n (with some probability)

Factoring Through p_k-smooth Numbers

Let p_k be the k-th prime.

Definition

An integer is p_k-smooth, if all its prime divisors are $\leq p_k$.

- use Sieve of Erathosthenes!

Idea (Morrison & Brillhart '75; Dixon '81)

- search for numbers a such that ($a^2 \bmod n$) is p_k-smooth
- construct x, y with $x^2 \equiv y^2 \mod n$
- construct some number that has a common divisor with n (with some probability)

Idea also used by Schnorr in his recent (failed) attempt at factoring. Instead of the first k primes, we may use any set of fixed primes.

- assume we have $k+1$ such numbers $a_0, \ldots, a_k$ with

$$(a_j^2 \bmod n) = \prod_{i=1}^{k} p_i^{d_{ij}}$$

- define the matrix $\boldsymbol{D} \in \{0,1\}^{(k+1)\times k}$ via $D_{ij} := d_{ij} \bmod 2$. regard it as row vectors
- non-trivial solution $\boldsymbol{t} \cdot \boldsymbol{D} = \boldsymbol{0} \mod 2 \quad \boldsymbol{t} \in \{0,1\}^{k+1}$

- assume we have $k+1$ such numbers $a_0, \ldots, a_k$ with

$$(a_j^2 \bmod n) = \prod_{i=1}^{k} p_i^{d_{ij}}$$

- define the matrix $\boldsymbol{D} \in \{0,1\}^{(k+1)\times k}$ via $D_{ij} := d_{ij} \bmod 2$. regard it as row vectors
- non-trivial solution $\boldsymbol{t} \cdot \boldsymbol{D} = \boldsymbol{0} \mod 2 \qquad \boldsymbol{t} \in \{0,1\}^{k+1}$
- note: $\frac{1}{2}\boldsymbol{t}\boldsymbol{D}$ is thus integer. put

$$x := \prod_{j:t_j=1} a_j \qquad\qquad y := \prod_{i=1}^{k} p_i^{\frac{1}{2}\sum_{j=0}^{k} t_j d_{ij}}$$

- assume we have $k+1$ such numbers $a_0, \ldots, a_k$ with

$$(a_j^2 \bmod n) = \prod_{i=1}^{k} p_i^{d_{ij}}$$

- define the matrix $\boldsymbol{D} \in \{0,1\}^{(k+1)\times k}$ via $D_{ij} := d_{ij} \bmod 2$. regard it as row vectors
- non-trivial solution $\boldsymbol{t} \cdot \boldsymbol{D} = \boldsymbol{0} \mod 2 \qquad \boldsymbol{t} \in \{0,1\}^{k+1}$
- note: $\frac{1}{2}\boldsymbol{t}\boldsymbol{D}$ is thus integer. put

$$x := \prod_{j:t_j=1} a_j \qquad\qquad y := \prod_{i=1}^{k} p_i^{\frac{1}{2}\sum_{j=0}^{k} t_j d_{ij}}$$

- have $x^2 \equiv y^2 \mod n$ (mult. the entries with $t_j = 1$)
- 50% chance: $x \equiv y \bmod p$ and $x \equiv -y \bmod q$ (or vice versa)
- get p, q from $\gcd(x+y, n)$ or $\gcd(x-y, n)$

Quadratic Sieve – Algorithm

- put $m = \lceil \sqrt{n} \rceil$, empty matrix $\boldsymbol{D}$
- for $j = 0, 1, \ldots$ try to factor

$$(m+j)^2 - n = \prod p_i^{d_{ij}} \cdot \text{remainder}$$

- if remainder $= 1$:
  store $m + j$ and append $d_{*,j}$ mod 2 to matrix $\boldsymbol{D}$

Quadratic Sieve – Algorithm

- put $m = \lceil \sqrt{n} \rceil$, empty matrix $\boldsymbol{D}$
- for $j = 0, 1, \ldots$ try to factor

$$(m+j)^2 - n = \prod p_i^{d_{ij}} \cdot \text{remainder}$$

- if remainder $= 1$:
  store $m + j$ and append $d_{*,j}$ mod 2 to matrix $\boldsymbol{D}$
- do Gaussian elimination on a copy $\boldsymbol{D}'$
- break if $\boldsymbol{D}'$ has zero-row
- construct $x^2 \equiv y^2 \mod n$ as above
- $p := \gcd(x \pm y, n)$, if $p \in \{1, n\}$, try again

Example (Back to the Quadratic Sieve)

Let $n = 583$, and factor base $S = \{2, 3, 5, 7\}$

i	2	3	5	7	remainder	store
0	1	1	0	1	1	✓
1	0	1	0	0	31	
2	1	0	0	0	73	
3	0	1	0	0	67	
4	1	1	0	0	43	
5	0	0	0	0	317	
6	1	1	0	1	1	✓

Example (Back to the Quadratic Sieve)

Let $n = 583$, and factor base $S = \{2, 3, 5, 7\}$

i	2	3	5	7	remainder	store
0	1	1	0	1	1	✓
1	0	1	0	0	31	
2	1	0	0	0	73	
3	0	1	0	0	67	
4	1	1	0	0	43	
5	0	0	0	0	317	
6	1	1	0	1	1	✓

already have matrix of lower rank $\leadsto$ break

$$\boldsymbol{D} = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

Example (Back to the Quadratic Sieve, cont.)

- try to factor $(m+i)^2 - n$ by factors from S
- if fully factors, add exponents to matrix $\boldsymbol{D}$
- stop if non-trivial kernel (rank lower than rows)

Example (Back to the Quadratic Sieve, cont.)

- try to factor $(m+i)^2 - n$ by factors from S
- if fully factors, add exponents to matrix $\boldsymbol{D}$
- stop if non-trivial kernel (rank lower than rows)
- end with matrix

$$\boldsymbol{D} = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix} \qquad \text{steps} = \begin{pmatrix} 0 \\ 6 \end{pmatrix}$$

- element $(1,1)$ from kernel tells us, which steps to combine

Example (Back to the Quadratic Sieve, cont.)

- try to factor $(m+i)^2 - n$ by factors from S
- if fully factors, add exponents to matrix $\boldsymbol{D}$
- stop if non-trivial kernel (rank lower than rows)
- end with matrix

$$\boldsymbol{D} = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix} \qquad \text{steps} = \begin{pmatrix} 0 \\ 6 \end{pmatrix}$$

- element $(1,1)$ from kernel tells us, which steps to combine
- hence, we get

$$(25 \cdot 31)^2 \equiv (2 \cdot 3^2 \cdot 7)^2 \mod 583$$

General Factorisation

Rough Steps

- primality test
- Check, whether n is (prime-)power
- Assume n has two different divisors
- look for any non-trivial factor
- recursively yields prime factorisation

General Factorisation

Rough Steps

- primality test
- Check, whether n is (prime-)power
- Assume n has two different divisors
- look for any non-trivial factor
- recursively yields prime factorisation

Check Power

- test, whether $x^e = n$ has solution for $e = 2, \ldots, \log n$

General Factorisation

Rough Steps

- primality test
- Check, whether n is (prime-)power
- Assume n has two different divisors
- look for any non-trivial factor
- recursively yields prime factorisation

Check Power

- test, whether $x^e = n$ has solution for $e = 2, \ldots, \log n$
- bisection with upper bound $2^{\lceil \|n\|/e \rceil}$
- each at most $\frac{1}{e} \log n$ bisection steps
- in each step, $\leq \log n$ mult. of size $\|n\|$

General Factorisation

Rough Steps

- primality test
- Check, whether n is (prime-)power
- Assume n has two different divisors
- look for any non-trivial factor
- recursively yields prime factorisation

Check Power

- test, whether $x^e = n$ has solution for $e = 2, \ldots, \log n$
- bisection with upper bound $2^{\lceil \|n\|/e \rceil}$
- each at most $\frac{1}{e} \log n$ bisection steps
- in each step, $\leq \log n$ mult. of size $\|n\|$

$\rightsquigarrow$ time polynomial in $\|n\|$

Factoring – Overview

- Trial division only for small numbers, $\leq 2^{64}$
- checking power is feasible
- $\leq 10^{100}$ quadratic sieve
- beyond: Number Field Sieve, time roughly $\mathcal{O}^*(\exp(c\sqrt[3]{\log n}))$

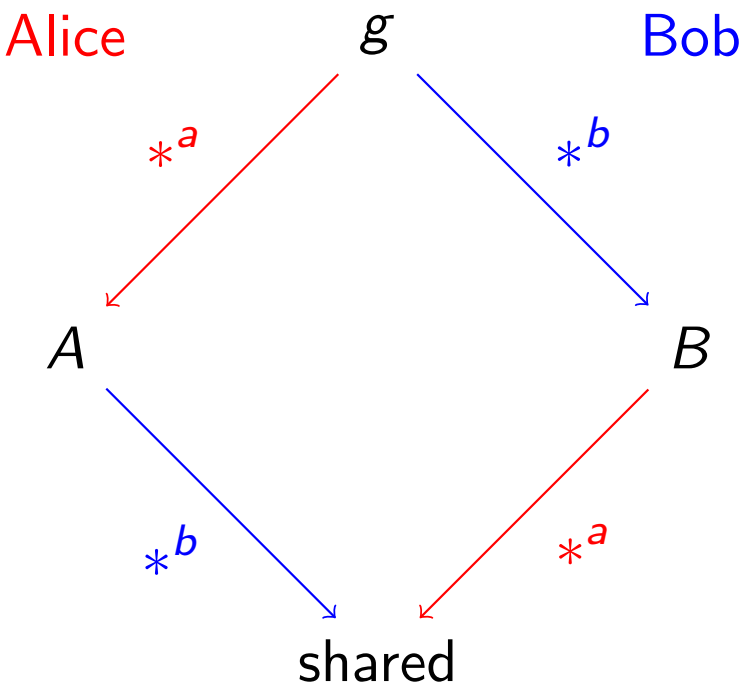
Factoring – Overview

- Trial division only for small numbers, $\leq 2^{64}$
- checking power is feasible
- $\leq 10^{100}$ quadratic sieve
- beyond: Number Field Sieve, time roughly $\mathcal{O}^*(\exp(c\sqrt[3]{\log n}))$

Implementations

- SymPy (slows down quickly)
- YAFU: quadratic Sieve
- cypari: number field sieve, easy from Python
- cado-nfs: fastest(?) number field sieve

Group Based Cryptography



Reminder Groups

What is a group?

- set with a single operation
- have neutral element and inverse
- we use $G = \langle g \rangle = \{g^n : n \in \mathbb{N}\}$, finite
- our groups are commutative

Reminder Groups

What is a group?

- set with a single operation
- have neutral element and inverse
- we use $G = \langle g \rangle = \{g^n : n \in \mathbb{N}\}$, finite
- our groups are commutative

Example

- just think of $G = \mathbb{Z}_p^* = \{1, \ldots, p-1\}$ with mult. for some prime p
- neutral element 1, modular inverse

Lemma

There always is some $g \in \mathbb{Z}_p^*$ with $\langle g \rangle = \mathbb{Z}_p^*$.

Motivation

- RSA works in $\mathbb{Z}_n$ for $n = pq$
- mathematical problem is e-th root modulo n

Motivation

- RSA works in $\mathbb{Z}_n$ for $n = pq$
- mathematical problem is e-th root modulo n
- now we regard systems that work in arbitrary groups
  but regard cyclic subgroup $\langle g \rangle \leq G$ for some $g \in G$
- the underlying problem is the discrete logarithm problem (DLP):

$$\text{given } g, g^x \in G, \text{ find } x \in \mathbb{N}.$$

Motivation

- RSA works in $\mathbb{Z}_n$ for $n = pq$
- mathematical problem is e-th root modulo n
- now we regard systems that work in arbitrary groups but regard cyclic subgroup $\langle g \rangle \leq G$ for some $g \in G$
- the underlying problem is the discrete logarithm problem (DLP):

$$\text{given } g, g^x \in G, \text{ find } x \in \mathbb{N}.$$

- framework for cryptosystem, until we decide which group comparable to abstract classes in programming
- keywords: Diffie-Hellman, Elliptic Curves, DSA, ElGamal

Motivation

- RSA works in $\mathbb{Z}_n$ for $n = pq$
- mathematical problem is e-th root modulo n
- now we regard systems that work in arbitrary groups but regard cyclic subgroup $\langle g \rangle \leq G$ for some $g \in G$
- the underlying problem is the discrete logarithm problem (DLP):

$$\text{given } g, g^x \in G, \text{ find } x \in \mathbb{N}.$$

- framework for cryptosystem, until we decide which group comparable to abstract classes in programming
- keywords: Diffie-Hellman, Elliptic Curves, DSA, ElGamal

Exercise

If we can solve the DLP in $\mathbb{Z}_n$, we can also factor n.

Diffie-Hellman Key-Exchange (1976)

Overview

- first published idea of public key cryptography
- no crypto-system, but key exchange
- we do not encode messages (yet), but get a common key then e.g. continue with symm. encryption
- also solves problem from symmetric encryption

Diffie-Hellman Key-Exchange (1976)

Overview

- first published idea of public key cryptography
- no crypto-system, but key exchange
- we do not encode messages (yet), but get a common key then e.g. continue with symm. encryption
- also solves problem from symmetric encryption

Method

- Alice chooses $a < o(g)$, computes $A = g^a$, sends A to Bob
- Bob chooses $b < o(g)$, computes $B = g^b$, sends B to Alice
- Alice computes key $K = B^a$
- Bob computed key $K = A^b$
- works, because $(g^a)^b = g^{ab} = g^{ba} = (g^b)^a$

Diffie-Hellman Key-Exchange – Example

Common, public agreement

- put $p = 22721$
- computation in $\mathbb{Z}_p^*$
- generator $g = 3$

Diffie-Hellman Key-Exchange – Example

Common, public agreement

- put $p = 22721$
- computation in $\mathbb{Z}_p^*$
- generator $g = 3$

Alice:

- choose $a = 18883$
- yields $A := g^a = 14581$
- send A to Bob
- compute $K_A := B^a = 5997$

Bob:

- choose $b = 5456$
- yields $B := g^b = 16742$
- send B to Alice
- compute $K_B := A^b = 5997$

Diffie-Hellman Key-Exchange – Example

Common, public agreement

- put $p = 22721$
- computation in $\mathbb{Z}_p^*$
- generator $g = 3$

Alice:

- choose $a = 18883$
- yields $A := g^a = 14581$
- send A to Bob
- compute $K_A := B^a = 5997$

Bob:

- choose $b = 5456$
- yields $B := g^b = 16742$
- send B to Alice
- compute $K_B := A^b = 5997$

common secret $K = 5997$

ElGamal (1985)

Key Generation

- secret key: choose random $a < o(g)$
- public key: $A = g^a$

ElGamal (1985)

Key Generation

- secret key: choose random $a < o(g)$
- public key: $A = g^a$

Usage

- Encrypt: m message to be encrypted
  choose random $b < o(g)$,
  send $(B, c) = \left(g^b, m \cdot A^b\right)$
- Decrypt: get (B, c), compute $m = c \cdot (B^a)^{-1}$

ElGamal (1985)

Key Generation

- secret key: choose random $a < o(g)$
- public key: $A = g^a$

Usage

- Encrypt: m message to be encrypted
  choose random $b < o(g)$,
  send $(B, c) = \left(g^b, m \cdot A^b\right)$
- Decrypt: get (B, c), compute $m = c \cdot (B^a)^{-1}$

- in fact just "asynchronous" Diffie-Hellman,
- use secret from handshake as mask

Example

- as before $\mathbb{Z}_p^*$ with $p = 22721$, $g = 3$
- Alice chooses $a = 18883$,
  public key $A = g^a = 14581$

Example

- as before $\mathbb{Z}_p^*$ with $p = 22721$, $g = 3$
- Alice chooses $a = 18883$,
  public key $A = g^a = 14581$

Bob now want to send message $m = 102$

- Bob chooses $b = 5456$,
  hence $B = g^b = 16742$
- masked message $c = mA^b = 20948$
- full cipher (B, c)

Example

- as before $\mathbb{Z}_p^*$ with $p = 22721$, $g = 3$
- Alice chooses $a = 18883$,
  public key $A = g^a = 14581$

Bob now want to send message $m = 102$

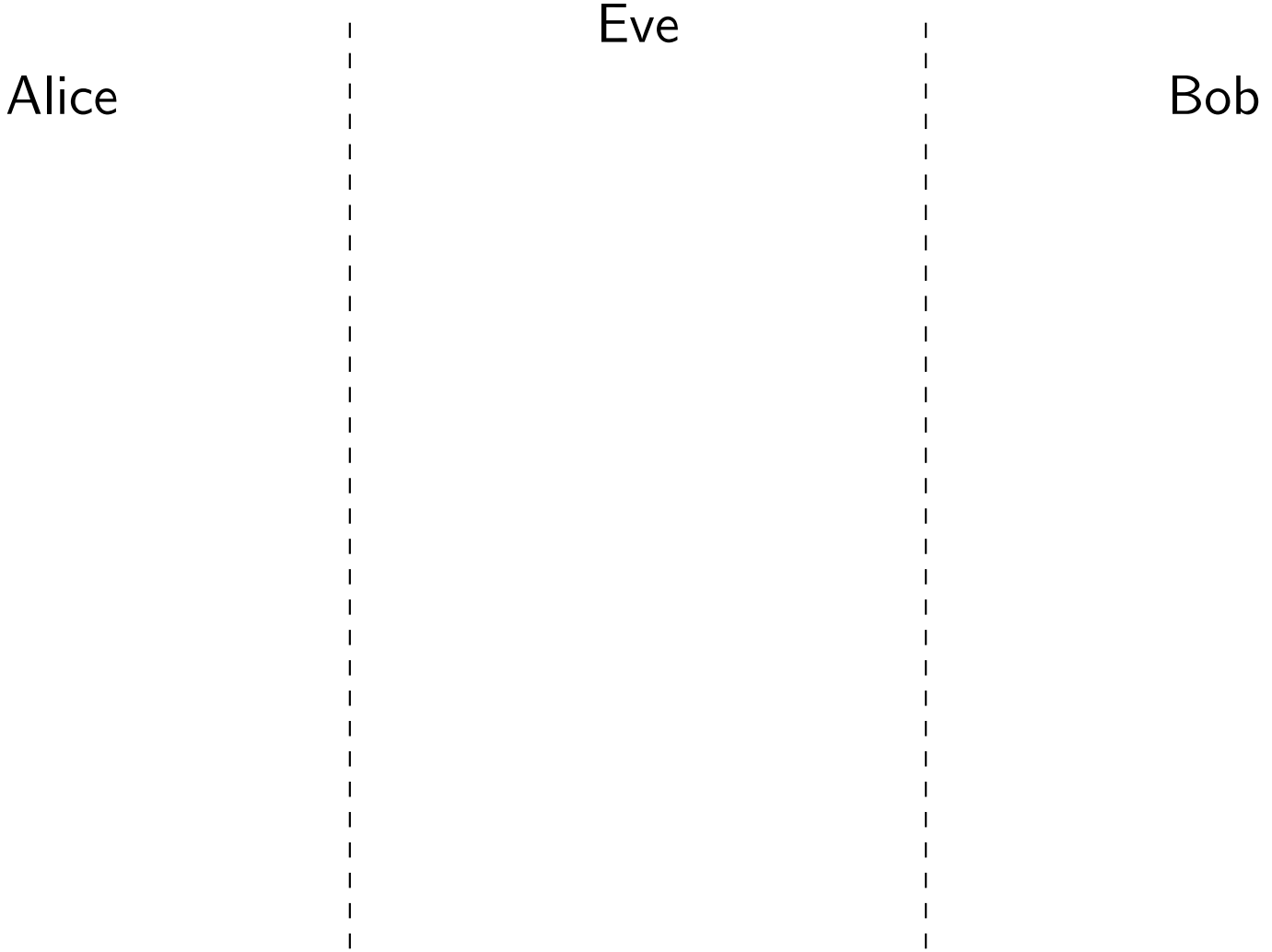
- Bob chooses $b = 5456$,
  hence $B = g^b = 16742$
- masked message $c = mA^b = 20948$
- full cipher (B, c)

Alice decrypts:

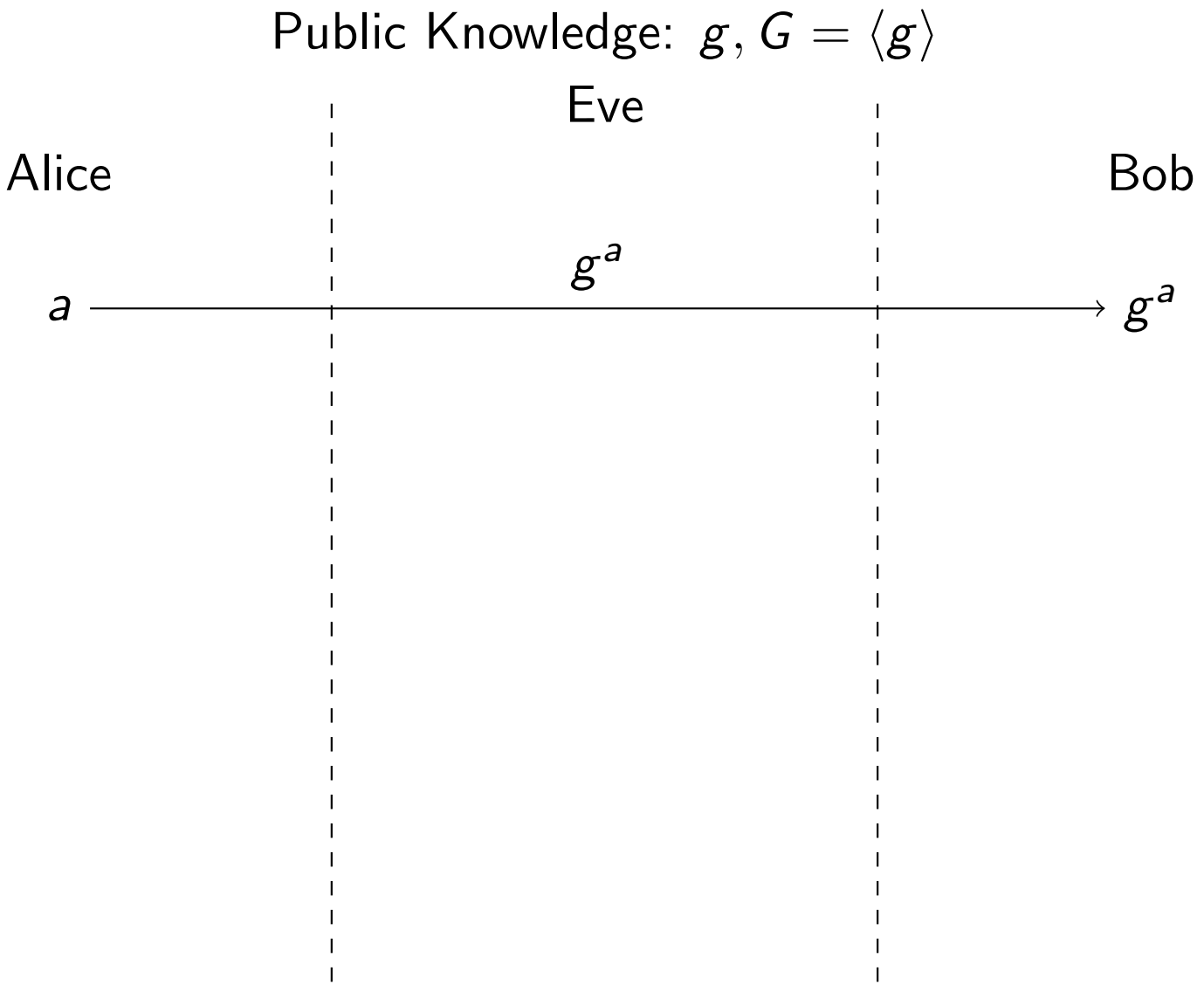
- original message via $(B^a)^{-1} \cdot c = 102$

Diffie-Hellman from Eve's view

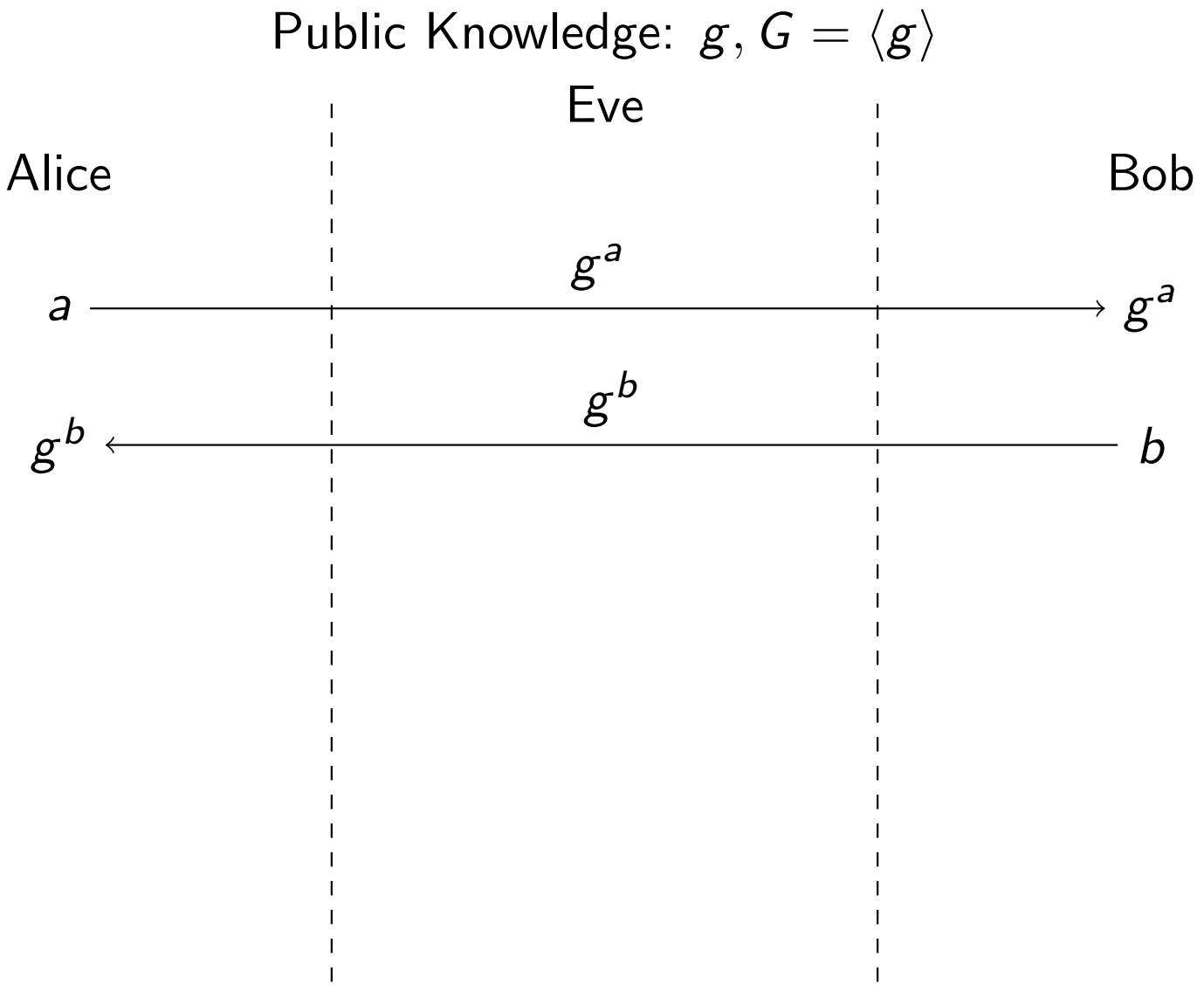
Public Knowledge: $g, G = \langle g \rangle$



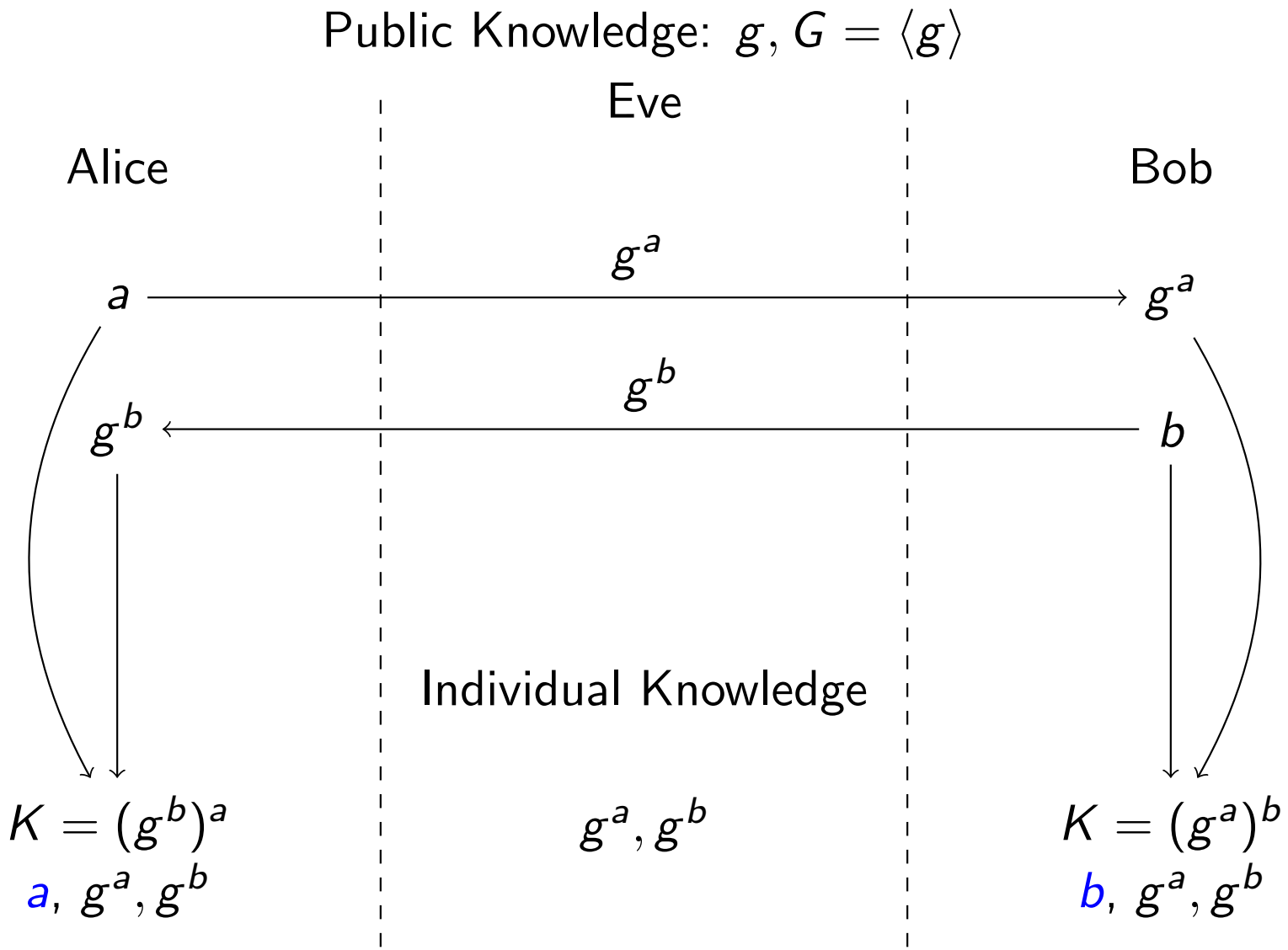
Diffie-Hellman from Eve's view



Diffie-Hellman from Eve's view



Diffie-Hellman from Eve's view



Attack Scenarios on Diffie-Hellman

Discrete Logarithm (DLP): given g, g^x, find x

find secret key

Computational DH (CDH): given g, g^a, g^b, find g^{ab}

find session key

Decisional DH (DDH): given g, g^a, g^b, h, decide $g^{ab} = h$

decide, which cipher belongs to message

recognise session key

Attack Scenarios on Diffie-Hellman

Discrete Logarithm (DLP): given g, g^x, find x

find secret key

Computational DH (CDH): given g, g^a, g^b, find g^{ab}

find session key

Decisional DH (DDH): given g, g^a, g^b, h, decide $g^{ab} = h$

decide, which cipher belongs to message

recognise session key

Trivially have hierarchy

$$\mathrm{DDH} \leq_p \mathrm{CDH} \leq_p \mathrm{DLP}$$

Attack Scenarios on Diffie-Hellman

Discrete Logarithm (DLP): given g, g^x, find x

find secret key

Computational DH (CDH): given g, g^a, g^b, find g^{ab}

find session key

Decisional DH (DDH): given g, g^a, g^b, h, decide $g^{ab} = h$

decide, which cipher belongs to message

recognise session key

Trivially have hierarchy

$$\mathrm{DDH} \leq_p \mathrm{CDH} \leq_p \mathrm{DLP}$$

Attacks on DLP

- Generic Attacks
- Attacks that exploit properties of the group

Brute-Force

Brute-Force attack on DLP

Input:

g – generator of group

$y = g^x$ for unknown x

Output: x – discrete log

function $\mathrm{DLP}(g,y)$

for $x = 0$ **to** n **do**

if $g^x = y$ **then**

return x

Brute-Force

Brute-Force attack on DLP

Input:

g – generator of group

$y = g^x$ for unknown x

Output: x – discrete log

function $\mathrm{DLP}(g,y)$

for $x = 0$ **to** n **do**

if $g^x = y$ **then**

return x

Analysis

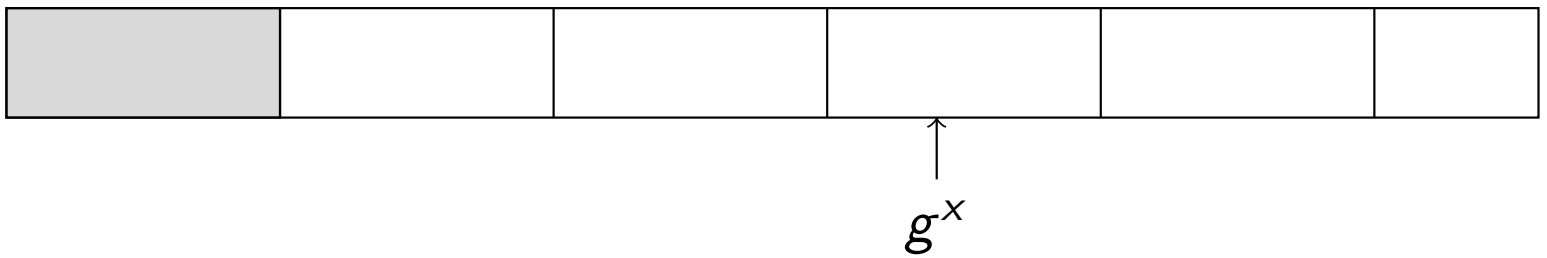
Let $n = o(g)$

Time: $\mathcal{O}(n)$ worst-case and expected

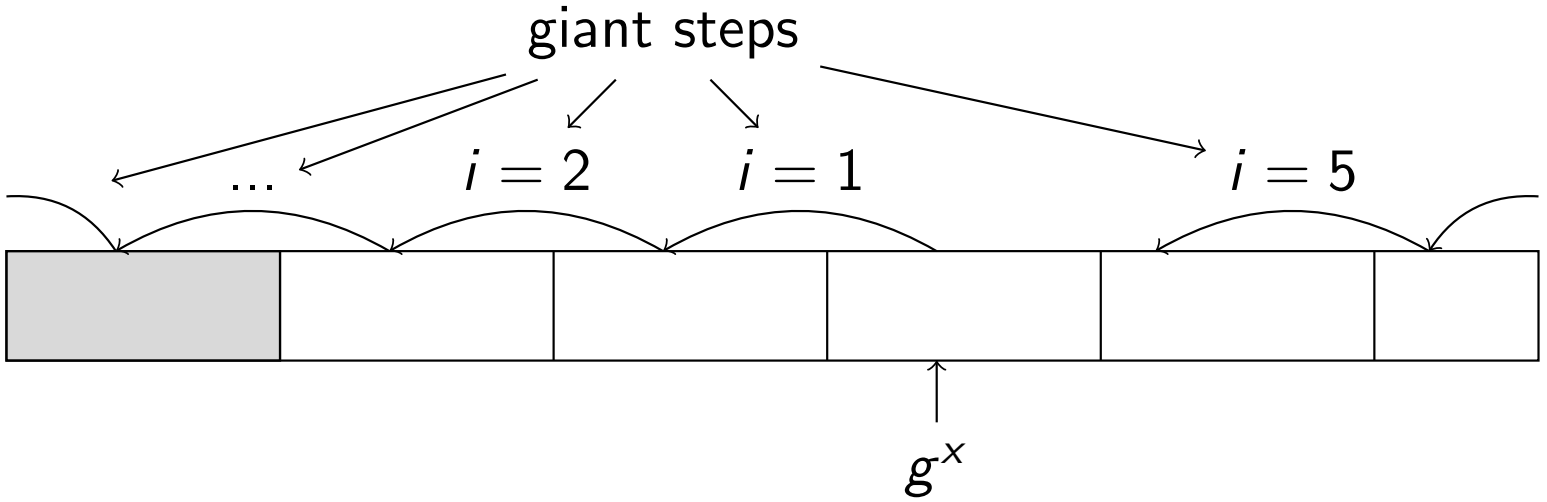
Space: $\mathcal{O}(1)$ numbers/group elements

which are technically of size $\mathcal{O}(\log n)$ each

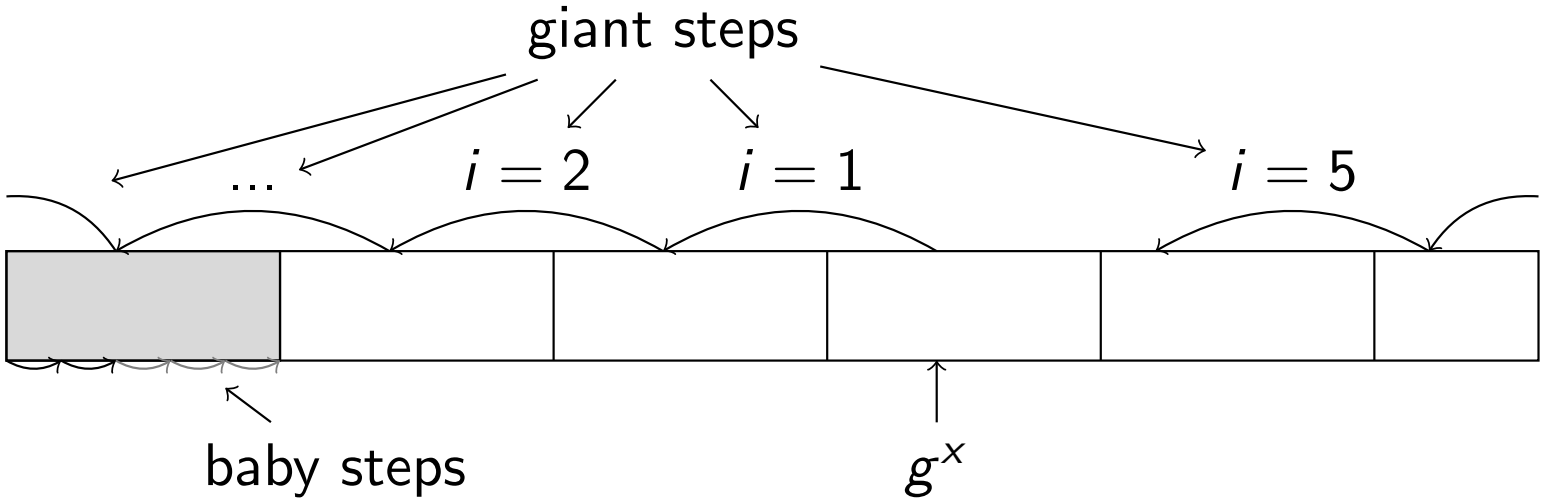
Shanks Baby-Step-Giant-Step – Picture



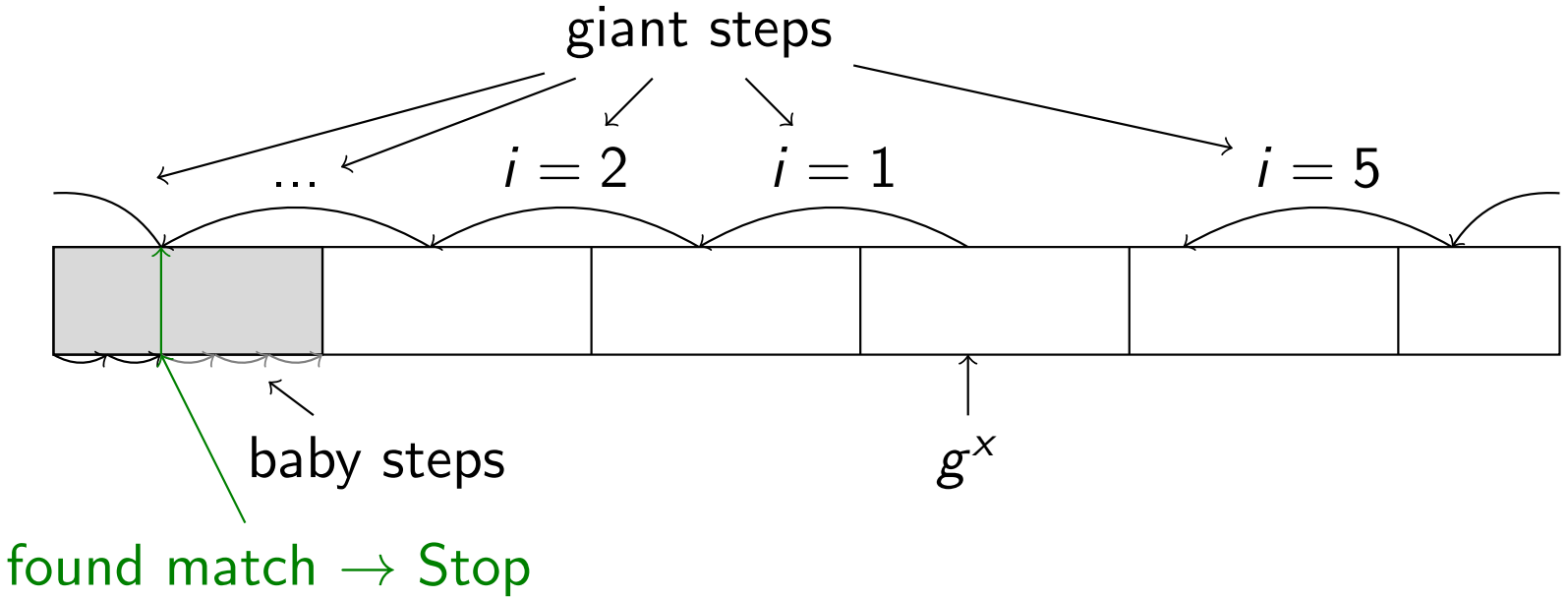
Shanks Baby-Step-Giant-Step – Picture



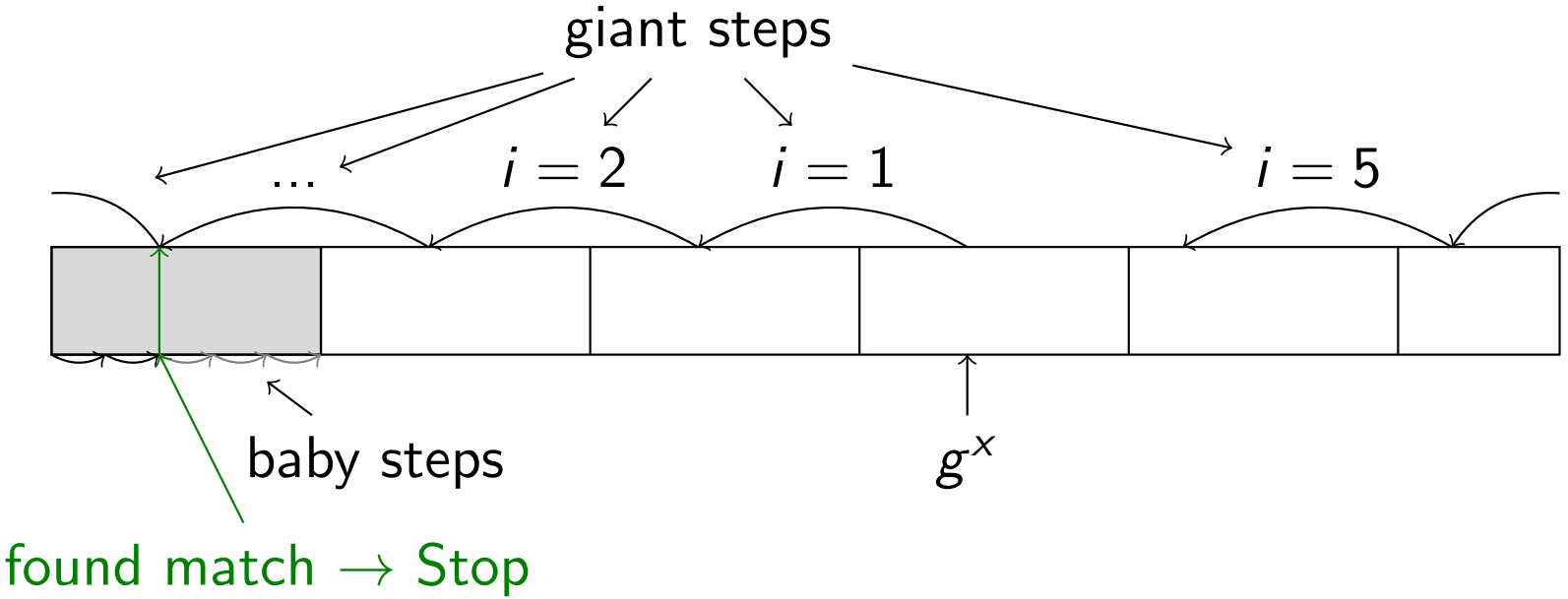
Shanks Baby-Step-Giant-Step – Picture



Shanks Baby-Step-Giant-Step – Picture



Shanks Baby-Step-Giant-Step – Picture



- store all giant steps
- can forget past baby steps
- some giant step will land in first (grey) block (but don't know which)
- some baby step will give a match

Baby-Step-Giant-Step – Formal

- Solve DLP: given g, g^x find x
- Let $n = o(g)$, pick giant-step size k
- secret x has unique representation $x = ki + j$ with $j < k$

Baby-Step-Giant-Step – Formal

- Solve DLP: given g, g^x find x
- Let $n = o(g)$, pick giant-step size k
- secret x has unique representation $x = ki + j$ with $j < k$
- Meet-in-the-middle:
  - List all $(g^x) \cdot g^{-ki}$ for $0 \leq i \leq \left\lfloor \frac{n}{k} \right\rfloor$
  - for g^j check whether in list
  - if match $g^j = g^x \cdot g^{-ki}$, we found $x = ki + j$

Baby-Step-Giant-Step – Formal

- Solve DLP: given g, g^x find x
- Let $n = o(g)$, pick giant-step size k
- secret x has unique representation $x = ki + j$ with $j < k$
- Meet-in-the-middle:
  - List all $(g^x) \cdot g^{-ki}$ for $0 \leq i \leq \left\lfloor \frac{n}{k} \right\rfloor$
  - for g^j check whether in list
  - if match $g^j = g^x \cdot g^{-ki}$, we found $x = ki + j$
- $k \approx \sqrt{n}$ yields time and space $\mathcal{O}(\sqrt{n})$
  always choose $k \geq \sqrt{n}$, keep space $\frac{n}{k}$ low

Baby-Step-Giant-Step – Formal

- Solve DLP: given g, g^x find x
- Let $n = o(g)$, pick giant-step size k
- secret x has unique representation $x = ki + j$ with $j < k$
- Meet-in-the-middle:
  - List all $(g^x) \cdot g^{-ki}$ for $0 \leq i \leq \left\lfloor \frac{n}{k} \right\rfloor$
  - for g^j check whether in list
  - if match $g^j = g^x \cdot g^{-ki}$, we found $x = ki + j$
- $k \approx \sqrt{n}$ yields time and space $\mathcal{O}(\sqrt{n})$
  always choose $k \geq \sqrt{n}$, keep space $\frac{n}{k}$ low
- compute powers via single steps, to improve speed
- compute once $s = \left(g^k\right)^{-1}$,
  then always "multiply" s in first loop
- always "multiply" g in second loop

Pohlig-Hellman Algorithm

Overview

- improve computation if factorisation of $n = o(g)$ is known
- solve problem for subgroups of prime power size
- compose with CRT
- Let p be largest prime divisor of n, running time

$$\mathcal{O}(\text{poly}(\|n\|) \cdot \sqrt{p})$$

- essential part of ROCA (see earlier)

Pohlig-Hellman Algorithm

Overview

- improve computation if factorisation of $n = o(g)$ is known
- solve problem for subgroups of prime power size
- compose with CRT
- Let p be largest prime divisor of n, running time

$$\mathcal{O}(\text{poly}(\|n\|) \cdot \sqrt{p})$$

- essential part of ROCA (see earlier)

Protection

- ensure n has large prime divisor
- "safe prime" p: select p such that $\frac{p-1}{2}$ also is prime, if $G = \mathbb{Z}_p^*$, then $n = p - 1$, ensured $n = 2 \cdot p'$, Pohlig-Hellman does not help

Breaking Prime Powers (Hensel Lifting)

- assume $|G| = n = p^e$ and $y = g^x$
- write $x = \sum_{i<e} x_i p^i$ in base p, then find "digits"
- put $h = g^{p^{e-1}}$, of order p (note $h^p = g^{p^e} = 1$)
- in each iteration eliminate all but one x_i

Breaking Prime Powers (Hensel Lifting)

- assume $|G| = n = p^e$ and $y = g^x$
- write $x = \sum_{i<e} x_i p^i$ in base p, then find "digits"
- put $h = g^{p^{e-1}}$, of order p (note $h^p = g^{p^e} = 1$)
- in each iteration eliminate all but one x_i
- first iteration

$$y^{p^{e-1}} = \left(g^{x_0 + x_1 p + \ldots + x_{e-1} p^{e-1}}\right)^{p^{e-1}} = g^{x_0 p^{e-1} + p^e \cdot *} = h^{x_0}$$

- find $x_0 = \log_h \left(y^{p^{e-1}}\right)$, e.g. via Shanks in $\mathcal{O}(\sqrt{p})$

Breaking Prime Powers (Hensel Lifting)

- assume $|G| = n = p^e$ and $y = g^x$
- write $x = \sum_{i<e} x_i p^i$ in base p, then find "digits"
- put $h = g^{p^{e-1}}$, of order p (note $h^p = g^{p^e} = 1$)
- in each iteration eliminate all but one x_i
- first iteration

$$y^{p^{e-1}} = \left(g^{x_0 + x_1 p + \ldots + x_{e-1} p^{e-1}}\right)^{p^{e-1}} = g^{x_0 p^{e-1} + p^e \cdot *} = h^{x_0}$$

- find $x_0 = \log_h \left(y^{p^{e-1}}\right)$, e.g. via Shanks in $\mathcal{O}(\sqrt{p})$
- continue for $i = 1, \ldots, e-1$

$$\left(y \cdot g^{-(x_0 + \ldots + x_{i-1} p^{i-1})}\right)^{p^{e-i-1}} = g^{x_i p^{e-1} + p^e \cdot *} = h^{x_i}$$

Composing Solution

- Assume we have factorisation

$$n = |G| = p_1^{e_1} \cdot \ldots \cdot p_\ell^{e_\ell}$$

- cancel out all components but i-th

$$n_i := n/p_i^{e_i} \qquad g_i := g^{n_i} \qquad y_i := y^{n_i}$$

Composing Solution

- Assume we have factorisation

$$n = |G| = p_1^{e_1} \cdot \ldots \cdot p_\ell^{e_\ell}$$

- cancel out all components but i-th

$$n_i := n/p_i^{e_i} \qquad g_i := g^{n_i} \qquad y_i := y^{n_i}$$

- compute $x_i = \log_{g_i}(y_i)$ as before,
  easy as group has order $p_i^{e_i}$

Composing Solution

- Assume we have factorisation

$$n = |G| = p_1^{e_1} \cdot \ldots \cdot p_\ell^{e_\ell}$$

- cancel out all components but i-th

$$n_i := n/p_i^{e_i} \qquad g_i := g^{n_i} \qquad y_i := y^{n_i}$$

- compute $x_i = \log_{g_i}(y_i)$ as before, easy as group has order $p_i^{e_i}$
- via CRT solve the system

$$x \equiv x_i \mod p_i^{e_i} \qquad \text{for } i = 1, \ldots, \ell$$

Composing Solution

- Assume we have factorisation

$$n = |G| = p_1^{e_1} \cdot \ldots \cdot p_\ell^{e_\ell}$$

- cancel out all components but i-th

$$n_i := n/p_i^{e_i} \qquad g_i := g^{n_i} \qquad y_i := y^{n_i}$$

- compute $x_i = \log_{g_i}(y_i)$ as before,
  easy as group has order $p_i^{e_i}$
- via CRT solve the system

$$x \equiv x_i \mod p_i^{e_i} \qquad \text{for } i = 1, \ldots, \ell$$

- running time $\mathcal{O}\left(\sum_i e_i(\log n + \sqrt{p_i})\right)$ group operations
  note: $\sum e_i \leq \log n$

Return of ROCA

Recall – Return of Coppersmith

- primorial $M = \prod p_i$ product of first primes
- given $65537^{a+b} \bmod M$, find $a + b$

Return of ROCA

Recall – Return of Coppersmith

- primorial $M = \prod p_i$ product of first primes
- given $65537^{a+b} \bmod M$, find $a + b$

- work in $\mathbb{Z}_M^*$
  in fact just the subgroup generated by 65537
- group size is $\varphi(M) = \prod_i^s (p_i - 1)$
- each factor small $< p_s$, so only small prime factors
  can actually factor each factor by trial division

$\Longrightarrow$ Pohlig-Hellman works

Overview Generic Attacks for DLP

In group $G = \langle g \rangle$ of size n, given g, g^x find x

Presented Methods

Shanks: meet-in-the-middle, space and time $\mathcal{O}(\sqrt{n})$

Pohlig-Hellman: faster, if factorisation of n known

let p largest prime factor of n:
time $\mathcal{O}(\text{poly}(\|n\|) \cdot \sqrt{p})$

Overview Generic Attacks for DLP

In group $G = \langle g \rangle$ of size n, given g, g^x find x

Presented Methods

Shanks: meet-in-the-middle, space and time $\mathcal{O}(\sqrt{n})$

Pohlig-Hellman: faster, if factorisation of n known
let p largest prime factor of n:
time $\mathcal{O}(\text{poly}(\|n\|) \cdot \sqrt{p})$

Other Methods

Pollard's Rho algorithm: probabilistic,
avoids large storage, time $\mathcal{O}(\sqrt{n})$

Pollard's Lambda/kangaroo algorithm: probabilistic,
if restricted to interval of size w, time $\mathcal{O}(\sqrt{w})$

DLP in Different Groups

Diffie-Hellman handshake is a template

DLP in Different Groups

Diffie-Hellman handshake is a template

Common Examples of (Finite) Groups

- additive group $(\mathbb{Z}_n, +)$
- multiplicative group $(\mathbb{Z}_n^*, \cdot)$
- symmetric group S_n (permutations)
- invertible matrices $GL(n, p^k) = \left\{ M \in \mathrm{GF}(p^k)^{n \times n} : \det M \neq 0 \right\}$

DLP in Different Groups

Diffie-Hellman handshake is a template

Common Examples of (Finite) Groups

- additive group $(\mathbb{Z}_n, +)$
- multiplicative group $(\mathbb{Z}_n^*, \cdot)$
- symmetric group S_n (permutations)
- invertible matrices $GL(n, p^k) = \left\{ M \in \mathrm{GF}(p^k)^{n \times n} : \det M \neq 0 \right\}$

Do not yield significant cryptographic advantage over RSA.

Elliptic Curves

- regard some curve in dimension 2
- define an "addition" for the points of that curve
- $\rightsquigarrow$ new kind of group (actually since end of 19th cent.)
- make everything discrete and finite

Additive Groups

Additive Group $(\mathbb{Z}_n, +)$

- group exponent is just multiple

$$y := g^x = \underbrace{g + \ldots + g}_{x\text{-times}} = x \cdot g$$

- trivial to break: $\log_g(y) = y \cdot g^{-1} \bmod n$

Additive Groups

Additive Group $(\mathbb{Z}_n, +)$

- group exponent is just multiple

$$y := g^x = \underbrace{g + \ldots + g}_{x\text{-times}} = x \cdot g$$

- trivial to break: $\log_g(y) = y \cdot g^{-1} \bmod n$

- actually have isomorphism $\varphi : \langle g \rangle \cong (\mathbb{Z}_{o(g)}, +)$, with $\varphi(g) = 1$, in general $\varphi(g^k) = k$ no matter which choice of g and G

Additive Groups

Additive Group $(\mathbb{Z}_n, +)$

- group exponent is just multiple

$$y := g^x = \underbrace{g + \ldots + g}_{x\text{-times}} = x \cdot g$$

- trivial to break: $\log_g(y) = y \cdot g^{-1} \bmod n$

- actually have isomorphism $\varphi : \langle g \rangle \cong (\mathbb{Z}_{o(g)}, +)$, with $\varphi(g) = 1$, in general $\varphi(g^k) = k$ no matter which choice of g and G
- translate from any group into $(\mathbb{Z}_n, +)$?
- but finding isomorphism is the DLP

Symmetric Group S_n

- breakable (bit like Pohlig-Hellman): regard cycles!

$$g = c_1 \circ c_2 \circ \ldots \circ c_k$$

- disjoint cycles independent: $g^x = c_1^x \circ \ldots \circ c_k^x$

Symmetric Group S_n

- breakable (bit like Pohlig-Hellman): regard cycles!

$$g = c_1 \circ c_2 \circ \ldots \circ c_k$$

- disjoint cycles independent: $g^x = c_1^x \circ \ldots \circ c_k^x$
- for each cycle, count base steps from first to second element

$$\log_{(1,5,3,2,4)}((1,2,5,4,3)) \mathrel{\widehat{=}} (1 \mapsto 5 \mapsto 3 \mapsto 2) \rightsquigarrow 3$$

Symmetric Group S_n

- breakable (bit like Pohlig-Hellman): regard cycles!

$$g = c_1 \circ c_2 \circ \ldots \circ c_k$$

- disjoint cycles independent: $g^x = c_1^x \circ \ldots \circ c_k^x$
- for each cycle, count base steps from first to second element

$$\log_{(1,5,3,2,4)}((1,2,5,4,3)) \mathrel{\widehat{=}} (1 \mapsto 5 \mapsto 3 \mapsto 2) \rightsquigarrow 3$$

- let ℓ_i be cycle lengths, then have x mod ℓ_i for all i
- compose with (generalised) CRT modulo the lcm

Symmetric Group S_n

- breakable (bit like Pohlig-Hellman): regard cycles!

$$g = c_1 \circ c_2 \circ \ldots \circ c_k$$

- disjoint cycles independent: $g^x = c_1^x \circ \ldots \circ c_k^x$
- for each cycle, count base steps from first to second element

$$\log_{(1,5,3,2,4)}((1,2,5,4,3)) \mathrel{\widehat{=}} (1 \mapsto 5 \mapsto 3 \mapsto 2) \rightsquigarrow 3$$

- let ℓ_i be cycle lengths, then have $x \bmod \ell_i$ for all i
- compose with (generalised) CRT modulo the lcm
- result is bounded by

$$o(g) \leq \ell_1 \cdot \ldots \cdot \ell_k \overset{\text{AM-GM}}{\leq} \left(\frac{\ell_1 + \ldots + \ell_k}{k}\right)^k = \left(\frac{n}{k}\right)^k \leq e^{n/e}$$

bit size $\|o(g)\| \in \mathcal{O}(n)$, for input size $\mathcal{O}(n \log n)$

Symmetric Group S_n – Example

Example (DLP in S_n)

take base element

$$g = (1, 7)(2, 6, 8)(3, 5, 4, 9, 10)$$

Symmetric Group S_n – Example

Example (DLP in S_n)

take base element

$$g = (1,7)(2,6,8)(3,5,4,9,10)$$

we are given Alice's public value

$$g^a = (1,7)(2,8,6)(3,4,10,5,9)$$

Symmetric Group S_n – Example

Example (DLP in S_n)

take base element

$$g = (1,7)(2,6,8)(3,5,4,9,10)$$

we are given Alice's public value

$$g^a = (1,7)(2,8,6)(3,4,10,5,9)$$

yields system

$$a \equiv 1 \mod 2 \qquad a \equiv 2 \mod 3 \qquad a \equiv 2 \mod 5$$

with solution $a = 17$

Invertible Matrices

General Linear Group

$$\mathrm{GL}(n, p^k) = \left\{ M \in \mathrm{GF}(p^k)^{n \times n} : \det M \neq 0 \right\}$$

$\mathrm{GF}(q)$ field with q elements (not $\mathbb{Z}_q$ if q is proper prime power)

Invertible Matrices

General Linear Group

$$\mathrm{GL}(n, p^k) = \left\{ M \in \mathrm{GF}(p^k)^{n \times n} : \det M \neq 0 \right\}$$

$\mathrm{GF}(q)$ field with q elements (not $\mathbb{Z}_q$ if q is proper prime power)

Theorem (Menezes, Wu, 1997)

We can transfer the DLP in $\mathrm{GL}(n, p^k)$ *to the DLP in* $\mathrm{GF}(p^{kn})$.

Invertible Matrices

General Linear Group

$$\mathrm{GL}(n, p^k) = \left\{ M \in \mathrm{GF}(p^k)^{n \times n} : \det M \neq 0 \right\}$$

$\mathrm{GF}(q)$ field with q elements (not $\mathbb{Z}_q$ if q is proper prime power)

Theorem (Menezes, Wu, 1997)

We can transfer the DLP in $\mathrm{GL}(n, p^k)$ *to the DLP in* $\mathrm{GF}(p^{kn})$.

- transfer t group with $p^{nk} - 1$ elements
- attack that one like $\mathbb{Z}_p$
- computation with matrices more expensive

$\Longrightarrow$ matrix has no advantage over $\mathrm{GF}(p^{nk})$

DLP in $\mathbb{Z}_p$

Task: given $g, n = o(g), y = g^x \mod p$, find x

Attack by Index Calculus

- pick up ideas from quadratic sieve
- let $p_1, \ldots, p_k$ be primes that can be written as $p_i = g^* \mod p$
- find powers g^r with

$$g^r \cdot y \equiv p_1^{e_1} \ldots p_k^{e_k} \mod p$$

DLP in $\mathbb{Z}_p$

Task: given $g, n = o(g), y = g^x \mod p$, find x

Attack by Index Calculus

- pick up ideas from quadratic sieve
- let $p_1, \ldots, p_k$ be primes that can be written as $p_i = g^* \mod p$
- find powers g^r with

$$g^r \cdot y \equiv p_1^{e_1} \ldots p_k^{e_k} \mod p$$

- enough of them give linear equation system

$$\log_g y \equiv -r + e_1 \log_g p_1 + \ldots + e_k log_g p_k \mod n$$

variables $\log_g y$ and the $\log_g p_i \rightsquigarrow$ solve

DLP in $\mathbb{Z}_p$

Task: given $g, n = o(g), y = g^x \mod p$, find x

Attack by Index Calculus

- pick up ideas from quadratic sieve
- let $p_1, \ldots, p_k$ be primes that can be written as $p_i = g^* \mod p$
- find powers g^r with

$$g^r \cdot y \equiv p_1^{e_1} \ldots p_k^{e_k} \mod p$$

- enough of them give linear equation system

$$\log_g y \equiv -r + e_1 \log_g p_1 + \ldots + e_k log_g p_k \mod n$$

variables $\log_g y$ and the $\log_g p_i \rightsquigarrow$ solve

running time like factoring

DLP in $\mathbb{Z}_p$

Setup

- prime n such that $p = 2n + 1$ is prime (Sophie-Germain prime)
- work in $\mathbb{Z}_p$, has order $\varphi(p) = p - 1 = 2n$
- pick random $g \neq 1$ until $g^n = 1$ (chance $\approx \frac{1}{2}$)
- then $G = \langle g \rangle$ has n elements

DLP in $\mathbb{Z}_p$

Setup

- prime n such that $p = 2n + 1$ is prime (Sophie-Germain prime)
- work in $\mathbb{Z}_p$, has order $\varphi(p) = p - 1 = 2n$
- pick random $g \neq 1$ until $g^n = 1$ (chance $\approx \frac{1}{2}$)
- then $G = \langle g \rangle$ has n elements

- best protection against Pohlig-Hellman
- same bit size as RSA for given security level
- Alice/Bob have two large exponentiations per handshake
- $\rightsquigarrow$ no advantage over RSA in that aspect
  (though better for "perfect forward secrecy")

Elliptic Curves

Definition

- let K be a finite field, $2 \neq 0 \neq 3$; e.g. $K = \mathbb{Z}_p$
- let $a, b \in K$ be parameters with $4a^3 + 27b^2 \neq 0$ (discriminant), needed to avoid degenerate case (curve behaves nicely)
- then the elliptic curve over K (in Weierstrass form) is

$$E(K) := \left\{(x, y) \in K^2 : y^2 = x^3 + ax + b\right\} \cup \{\infty\}$$

Elliptic Curves

Definition

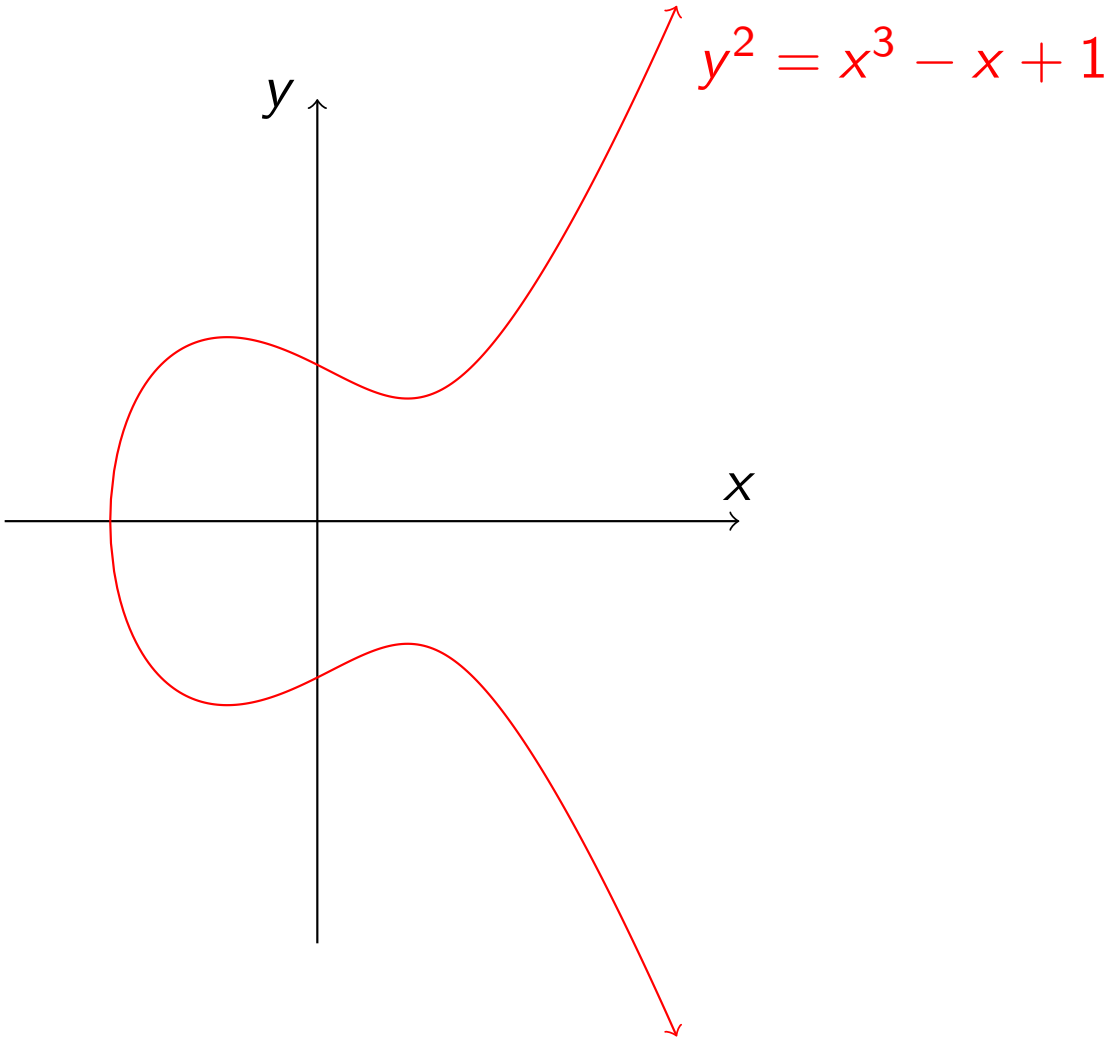
- let K be a finite field, $2 \neq 0 \neq 3$; e.g. $K = \mathbb{Z}_p$
- let $a, b \in K$ be parameters with $4a^3 + 27b^2 \neq 0$ (discriminant), needed to avoid degenerate case (curve behaves nicely)
- then the elliptic curve over K (in Weierstrass form) is

$$E(K) := \{(x, y) \in K^2 : y^2 = x^3 + ax + b\} \cup \{\infty\}$$

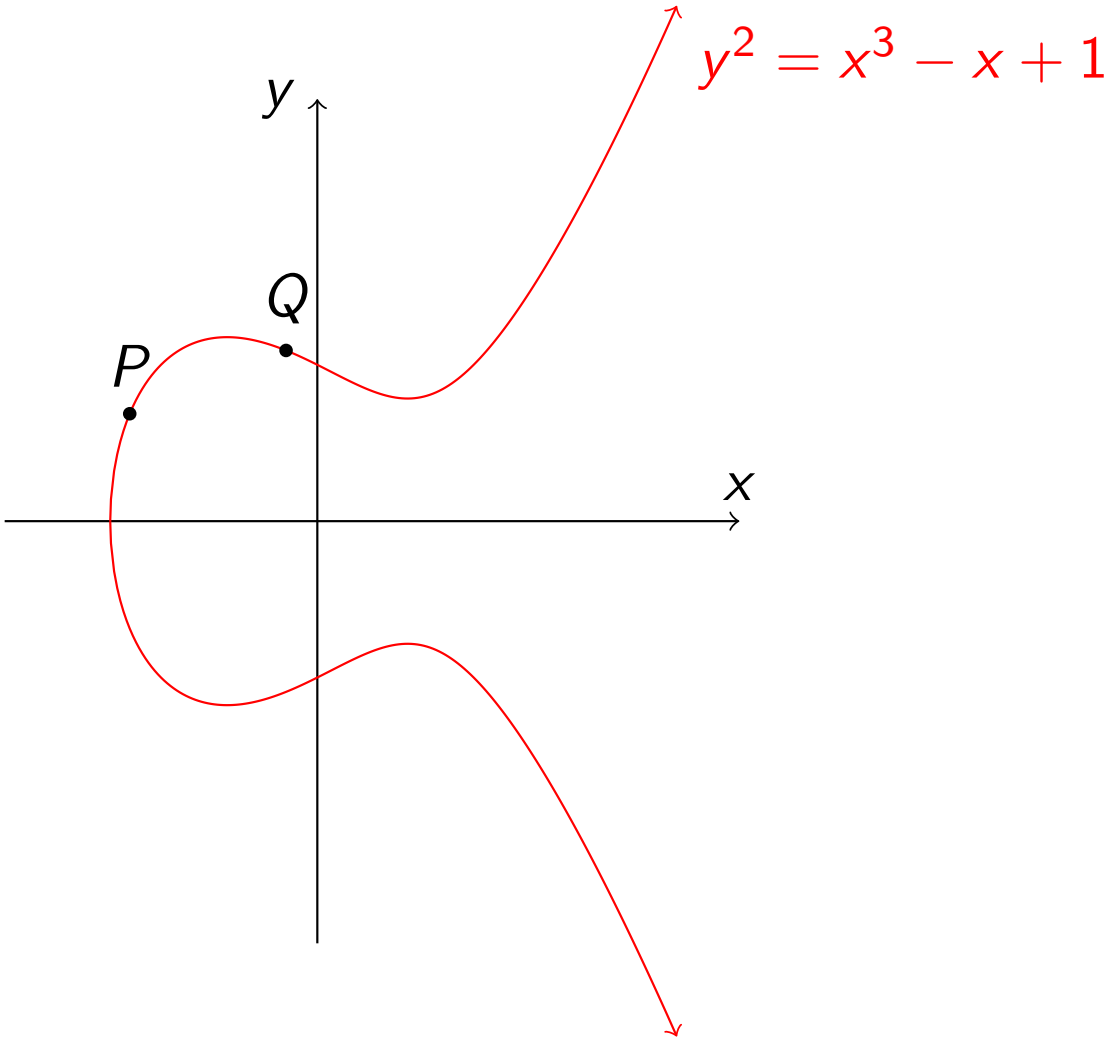
Remark

- *often used in projective coordinates, i.e. in K^3*
  no inversion in addition $\rightsquigarrow$ speed-up
- *alternative form: Montgomery curve*
  different formulas for addition

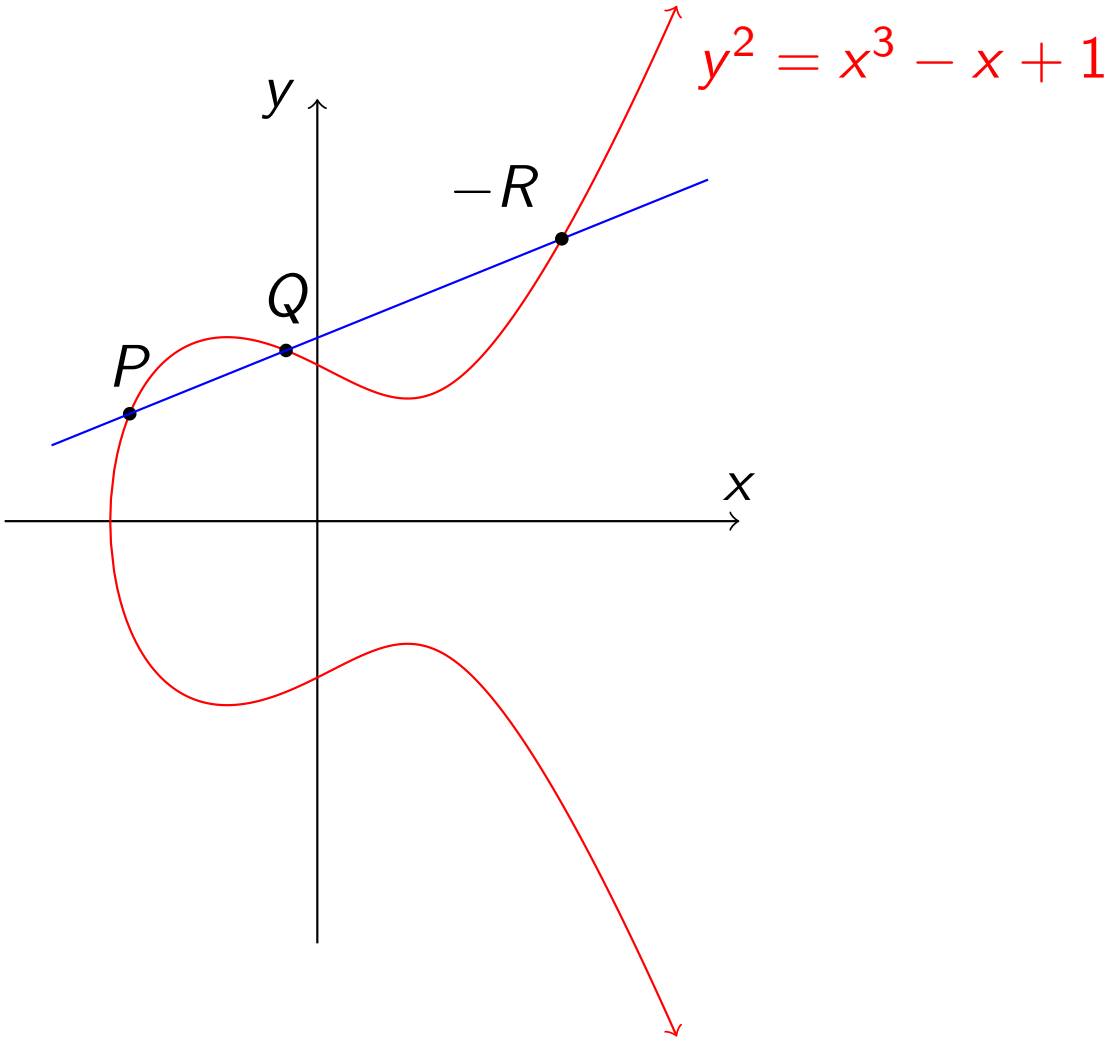
Elliptic Curves as Group



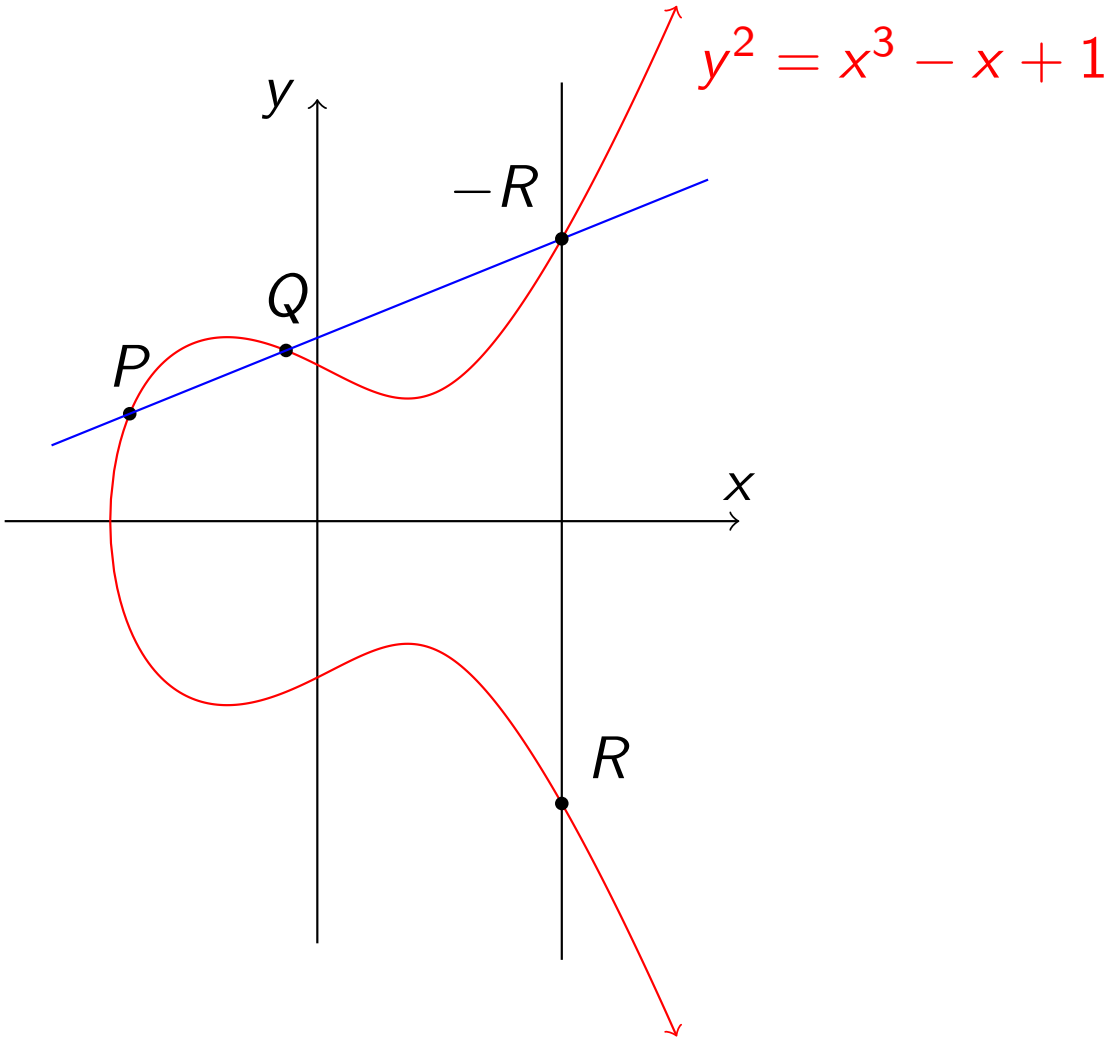
Elliptic Curves as Group



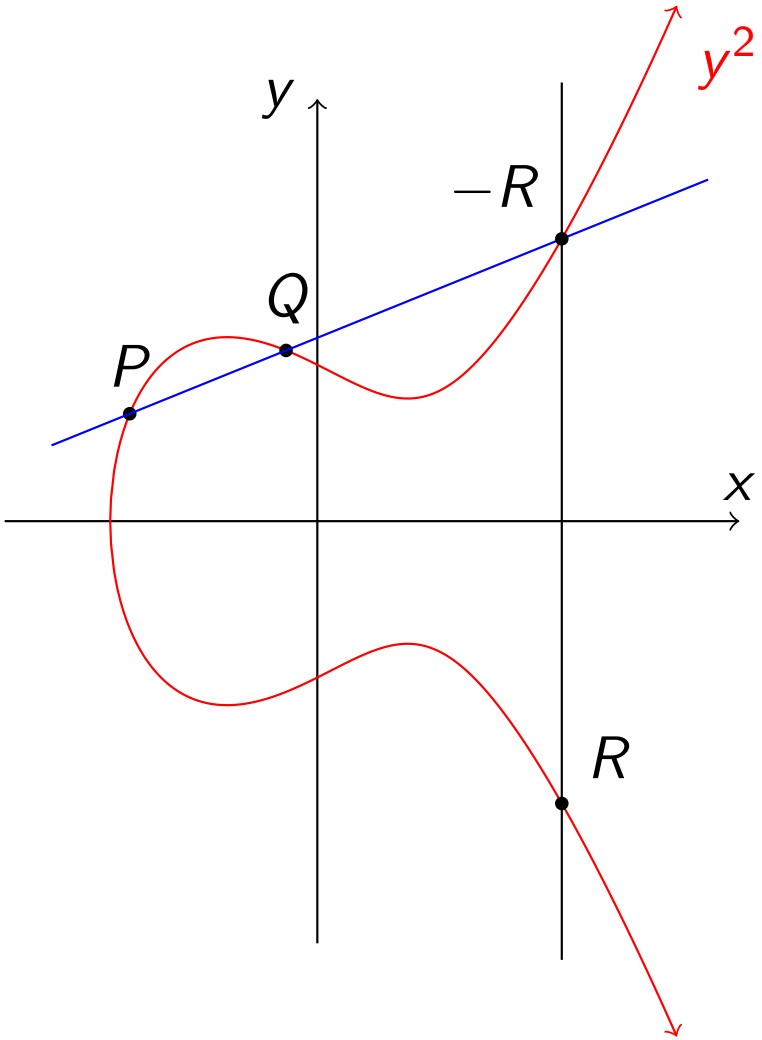
Elliptic Curves as Group



Elliptic Curves as Group



Elliptic Curves as Group



Point "Addition": $R = P \cdot Q$

- draw line through P and Q
- get third intersection with curve $\rightsquigarrow R^{-1}$
  (ensured by discriminant)
- mirror on x-axis

Case $R = P \cdot P$:

- use tangent instead

Case $P_x = Q_x$, $P_y \neq Q_y$:

- $R = \infty$

Formulas for point addition

Neutral element: $P \cdot \infty = P$ for all P

Inverse: $P_x = Q_x$ but $P_y = -Q_y$, then $P \cdot Q = \infty$

General case: $R = P \cdot Q$

$$\lambda = \begin{cases} \frac{Q_y - P_y}{Q_x - P_x} & : P \neq Q \\ \frac{3P_x^2 + a}{2P_y} & : P = Q \end{cases}$$

$$R_x = \lambda^2 - P_x - Q_x$$

$$R_y = \lambda(P_x - R_x) - P_y$$

formulas work in every field (as long as $2 \neq 0 \neq 3$)

Formulas for point addition

Neutral element: $P \cdot \infty = P$ for all P

Inverse: $P_x = Q_x$ but $P_y = -Q_y$, then $P \cdot Q = \infty$

General case: $R = P \cdot Q$

$$\lambda = \begin{cases} \frac{Q_y - P_y}{Q_x - P_x} & : P \neq Q \\ \frac{3P_x^2 + a}{2P_y} & : P = Q \end{cases}$$

$$R_x = \lambda^2 - P_x - Q_x$$

$$R_y = \lambda(P_x - R_x) - P_y$$

formulas work in every field (as long as $2 \neq 0 \neq 3$)

in fact, they even work for points not on the curve

Faulty Curve Injection

get Alice's secret key in ElGamal

Condition

- Chosen Cipher Attack
- Alice does not check whether $B \in E$

Faulty Curve Injection

get Alice's secret key in ElGamal

Condition

- Chosen Cipher Attack
- Alice does not check whether $B \in E$

- work in curve $y^2 = x^3 + ax + b$
- computations don't use b
- works for $P, Q \notin E$

Faulty Curve Injection

get Alice's secret key in ElGamal

Condition

- Chosen Cipher Attack
- Alice does not check whether $B \in E$

- work in curve $y^2 = x^3 + ax + b$
- computations don't use b
- works for $P, Q \notin E$

Idea

- Eve sends $X \notin E$ instead of $B \in E$
- can extract Alice's secret exponent

Faulty Curve Injection – Attack

Attack: given $A = g^a$, find a

- Eve picks random point $B' \in K^2$
- gives point on new curve $E' : y^2 = x^3 + ax + b'$
- try until order $o(B')$ has only small prime divisors

  chance is good enough, offline search

Faulty Curve Injection – Attack

Attack: given $A = g^a$, find a

- Eve picks random point $B' \in K^2$
- gives point on new curve $E' : y^2 = x^3 + ax + b'$
- try until order $o(B')$ has only small prime divisors
  - chance is good enough, offline search
- send (B'^{-1}, ∞) (recall: ∞ is neutral element)
  - usually would send $(B, c) = (g^b, m \cdot A^b)$
- decryption: $m = \infty \cdot (B'^{-1})^{-a} = B'^a$
  - usually would be $m = c \cdot B^{-a}$
- use Pohlig-Hellman to compute a

Faulty Curve Injection – Attack

Attack: given $A = g^a$, find a

- Eve picks random point $B' \in K^2$
- gives point on new curve $E' : y^2 = x^3 + ax + b'$
- try until order $o(B')$ has only small prime divisors

  chance is good enough, offline search
- send (B'^{-1}, ∞) (recall: ∞ is neutral element)

  usually would send $(B, c) = (g^b, m \cdot A^b)$
- decryption: $m = \infty \cdot (B'^{-1})^{-a} = B'^a$

  usually would be $m = c \cdot B^{-a}$
- use Pohlig-Hellman to compute a

Remark

- could have used real message
- since we know c, m, we always get B'^a

Open Questions

Question

- How do we get an elliptic curve?
- Which base element do we pick?
- What is the group size?
- Where can we optimise?

Open Questions

Question

- How do we get an elliptic curve?
- Which base element do we pick?
- What is the group size?
- Where can we optimise?

Constructing an Elliptic Curve

- choose random prime p of chosen bit size, work in $\mathbb{Z}_p$
- choose random $a, b \in \mathbb{Z}_p \rightsquigarrow$ defines curve
- compute size of group, want prime order
- then find generator g

Open Questions

Question

- How do we get an elliptic curve?
- Which base element do we pick?
- What is the group size?
- Where can we optimise?

Constructing an Elliptic Curve

- choose random prime p of chosen bit size, work in $\mathbb{Z}_p$
- choose random $a, b \in \mathbb{Z}_p \rightsquigarrow$ defines curve
- compute size of group, want prime order
- then find generator g

Do once $\rightsquigarrow$ just pick some standard curve

Counting Points

Theorem (Hasse, 1933)

Let $K = \mathbb{Z}_p$. For the size $|E|$ of the curve, we have the bound

$$||E| - (p+1)| \leq 2\sqrt{p}$$

Counting Points

Theorem (Hasse, 1933)

Let $K = \mathbb{Z}_p$. For the size $|E|$ of the curve, we have the bound

$$||E| - (p+1)| \leq 2\sqrt{p}$$

Counting Points

let $M(k)$ denote complexity of multiplication in k digits

Baby-Step-Giant-Step: $\mathcal{O}(\sqrt[4]{p})$ group operations

Schoof: time $\mathcal{O}\left(\|p\|^2 M(\|p\|^3)/\log\|p\|\right) \approx \mathcal{O}\left(\|p\|^5\right)$

Schoof-Elkies-Atkin: time $\mathcal{O}\left(\|p\|^2 M(\|p\|^2)/\log\|p\|\right) \approx \mathcal{O}\left(\|p\|^4\right)$

significant improvement: $p^{\frac{1}{4}} \rightsquigarrow \text{poly}(\log p)$

slow, but feasible

Ideas Behind Methods

Baby-Step-Giant-Step

- $|E|$ lies in interval $\left[p+1 \pm 2\sqrt{p}\right]$ of size $4\sqrt{p}$
- pick random $P \in E$: pick random x, until $x^3 + ax + b$ is square (50% chance), compute y as root
- if only single k in interval with $P^k = \infty$, then $|E| = k$
- else try new P, chance sufficiently good
- reduce time via meet-in-the-middle

Ideas Behind Methods

Baby-Step-Giant-Step

- $|E|$ lies in interval $\left[p+1 \pm 2\sqrt{p}\right]$ of size $4\sqrt{p}$
- pick random $P \in E$: pick random x, until $x^3 + ax + b$ is square (50% chance), compute y as root
- if only single k in interval with $P^k = \infty$, then $|E| = k$
- else try new P, chance sufficiently good
- reduce time via meet-in-the-middle

Schoof (with a lot of Galois theory)

- find $|E|$ mod q_i for some primes q_i
- until $\prod q_i > 4\sqrt{p}$
- then $|E|$ is CRT solution in interval $p+1 \pm 2\sqrt{p}$

Find Base Element

Design Goal

subgroup $\langle g \rangle = G \leq E$ of prime order $|G| = p$ with $\|p\| \approx \|n\|$

- computational effort grows with n
- want high security level (large p) with low effort (small n)

Find Base Element

Design Goal

subgroup $\langle g \rangle = G \leq E$ of prime order $|G| = p$ with $\|p\| \approx \|n\|$

- computational effort grows with n
- want high security level (large p) with low effort (small n)

Method

- create elliptic curve E
- compute size $n := |E|$
- trial division by the first few primes
- if remainder p not prime, start again

Find Base Element

Design Goal

subgroup $\langle g \rangle = G \leq E$ of prime order $|G| = p$ with $\|p\| \approx \|n\|$

- computational effort grows with n
- want high security level (large p) with low effort (small n)

Method

- create elliptic curve E
- compute size $n := |E|$
- trial division by the first few primes
- if remainder p not prime, start again
- Algebra: for prime $p \mid n$, there is g with $o(g) = p$

  actually $p - 1$ many
- try random g, chance $\approx p/n$

Computational Effort in Key Generation

Construct Group

group generation involves

- point counting, $\mathcal{O}^*\left(\|p\|^4\right)$
  feasible, but may need several attempts $\rightsquigarrow$ long time
- factoring: only trial division $\rightsquigarrow$ fast
- find generator: good chance $\rightsquigarrow$ fast

Computational Effort in Key Generation

Construct Group

group generation involves

- point counting, $\mathcal{O}^*\left(\|p\|^4\right)$
  feasible, but may need several attempts $\rightsquigarrow$ long time
- factoring: only trial division $\rightsquigarrow$ fast
- find generator: good chance $\rightsquigarrow$ fast

Key Observation

- Alice and Bob use the same curve
- same curve for everyone
- expensive computations have to be done only once

Computational Effort in Key Generation

Construct Group

group generation involves

- point counting, $\mathcal{O}^*\left(\|p\|^4\right)$
  feasible, but may need several attempts $\rightsquigarrow$ long time
- factoring: only trial division $\rightsquigarrow$ fast
- find generator: good chance $\rightsquigarrow$ fast

Key Observation

- Alice and Bob use the same curve
- same curve for everyone
- expensive computations have to be done only once

Individual Part

- create random number $r < p$, compute $g^r \rightsquigarrow$ easy

Optimisation in ECDH

Speed Up Computations

- frequently have to compute P^k
- use square-and-multiply (double-and-add),
  $\mathcal{O}(\log k)$ operations
- negation cheap: also use subtraction
  $k = *0, 1, 1, \ldots, 1, 1, 0*$ becomes $*1, 0, \ldots, 0, -1, 0*$
  some doubling + 1 subtraction
  worst case: $\frac{3}{2}k$ ops. (instead of $2k$)

Optimisation in ECDH

Speed Up Computations

- frequently have to compute P^k
- use square-and-multiply (double-and-add), $\mathcal{O}(\log k)$ operations
- negation cheap: also use subtraction
  $k = *0, 1, 1, \ldots, 1, 1, 0*$ becomes $*1, 0, \ldots, 0, -1, 0*$
  some doubling + 1 subtraction
  worst case: $\frac{3}{2}k$ ops. (instead of $2k$)

Side Channel Attacks

- varying time leaks information
- mostly aim for constant time
- even at the prize of longer time

Other Insecure Special Cases

Standard curves are also tested against other attacks.

Multiplicative Transfer

- let $\ell = o(g)$ with $\gcd(\ell, p) = 1$, k minimal with $\ell \mid p^k - 1$
- can transfer DLP to $\left(\mathrm{GF}(p^k)^*, \cdot\right)$, subexponential solutions

Other Insecure Special Cases

Standard curves are also tested against other attacks.

Multiplicative Transfer

- let $\ell = o(g)$ with $\gcd(\ell, p) = 1$, k minimal with $\ell \mid p^k - 1$
- can transfer DLP to $\left(\mathrm{GF}(p^k)^*, \cdot\right)$, subexponential solutions

Additive Transfer

- anomalous curve: $|E| = p$
- can transfer DLP to $(\mathbb{Z}_p, +)$, easy to solve

Other Insecure Special Cases

Standard curves are also tested against other attacks.

Multiplicative Transfer

- let $\ell = o(g)$ with $\gcd(\ell, p) = 1$, k minimal with $\ell \mid p^k - 1$
- can transfer DLP to $\left(\mathrm{GF}(p^k)^*, \cdot\right)$, subexponential solutions

Additive Transfer

- anomalous curve: $|E| = p$
- can transfer DLP to $(\mathbb{Z}_p, +)$, easy to solve

and some others...

Other Insecure Special Cases

Standard curves are also tested against other attacks.

Multiplicative Transfer

- let $\ell = o(g)$ with $\gcd(\ell, p) = 1$, k minimal with $\ell \mid p^k - 1$
- can transfer DLP to $\left(\mathrm{GF}(p^k)^*, \cdot\right)$, subexponential solutions

Additive Transfer

- anomalous curve: $|E| = p$
- can transfer DLP to $(\mathbb{Z}_p, +)$, easy to solve

and some others...

Final Take-Away

Just use a given curve, maybe not from NIST.

Encryption with Elliptic Curves – Overview

Encryption

- choose one of the standard curves
  - all in every standard library $\rightsquigarrow$ no effort
  - even implementing them on your own is dangerous

Encryption with Elliptic Curves – Overview

Encryption

- choose one of the standard curves
  - all in every standard library $\rightsquigarrow$ no effort
  - even implementing them on your own is dangerous
- Alice and Bob perform DH handshake
  - create one random number
  - perform two group exponentiations
  - check that result lies on curve
- use ElGamal or continue with AES

Encryption with Elliptic Curves – Overview

Encryption

- choose one of the standard curves
  - all in every standard library $\rightsquigarrow$ no effort
  - even implementing them on your own is dangerous
- Alice and Bob perform DH handshake
  - create one random number
  - perform two group exponentiations
  - check that result lies on curve
- use ElGamal or continue with AES

But what about signatures?

ECDSA – Elliptic Curve Digital Signature Algorithm

Setting

- subgroup $\langle g \rangle$ of prime size n in an elliptic curve
- $a < n$ – secret key
- $A = g^a$ – public key
- hash – some hash function, e.g. SHA
- msg – message to be signed

ECDSA – Elliptic Curve Digital Signature Algorithm

Setting

- subgroup $\langle g \rangle$ of prime size n in an elliptic curve
- $a < n$ – secret key
- $A = g^a$ – public key
- hash – some hash function, e.g. SHA
- msg – message to be signed

Signature (ignoring edge cases)

- random $k < n$, compute $(x, y) = g^k$
- $r = x \bmod n$
- $s = k^{-1}(\text{hash}(\text{msg}) + r \cdot a) \bmod n$
- signature (r, s)

- $(x, y) = g^k$ and $r = x \bmod n$
- $s = k^{-1}(\text{hash}(\text{msg}) + r \cdot a) \bmod n$

- $(x, y) = g^k$ and $r = x \bmod n$
- $s = k^{-1}(\text{hash}(\text{msg}) + r \cdot a) \bmod n$

Verification

- receive signature (r, s) and message msg
- compute $u = \text{hash}(\text{msg}) \cdot s^{-1} \bmod n$ and $v = rs^{-1} \bmod n$
- compute $(x', y') = g^u \cdot A^v$ in the curve
- accept if $r \equiv x' \bmod n$

- $(x, y) = g^k$ and $r = x \bmod n$
- $s = k^{-1}(\text{hash}(\text{msg}) + r \cdot a) \bmod n$

Verification

- receive signature (r, s) and message msg
- compute $u = \text{hash}(\text{msg}) \cdot s^{-1} \bmod n$ and $v = rs^{-1} \bmod n$
- compute $(x', y') = g^u \cdot A^v$ in the curve
- accept if $r \equiv x' \mod n$

Correctness

plugging in the supposed values:

$$\begin{aligned}(x', y') &= g^u \cdot A^v \\ &= \left(g^{\text{hash}(\text{msg})} \cdot g^{ra}\right)^{s^{-1}} \\ &= g^k = (x, y)\end{aligned}$$

Psychic Paper

- $(x, y) = g^k$ and $r = x \bmod n$
- $s = k^{-1}(\text{hash}(\text{msg}) + r \cdot a) \bmod n$

Why edge cases are important

- above pseudo code is vulnerable
- some implementations say $0^{-1} \bmod n = 0$
- also ∞ not treated correctly, but as with zero
- then signature $(0, 0)$ always accepted

Psychic Paper

- $(x, y) = g^k$ and $r = x \bmod n$
- $s = k^{-1}(\text{hash}(\text{msg}) + r \cdot a) \bmod n$

Why edge cases are important

- above pseudo code is vulnerable
- some implementations say $0^{-1} \bmod n = 0$
- also ∞ not treated correctly, but as with zero
- then signature $(0, 0)$ always accepted
- discovered in the wild in April 2022, in Java 15 to 18
- unintended vuln at NullConCTF Goa 2022

Psychic Paper

- $(x, y) = g^k$ and $r = x \bmod n$
- $s = k^{-1}(\text{hash}(\text{msg}) + r \cdot a) \bmod n$

Why edge cases are important

- above pseudo code is vulnerable
- some implementations say $0^{-1} \bmod n = 0$
- also ∞ not treated correctly, but as with zero
- then signature $(0, 0)$ always accepted
- discovered in the wild in April 2022, in Java 15 to 18
- unintended vuln at NullConCTF Goa 2022

Fun Fact

vuln named after psychic paper in Doctor Who

Sony's failure with the PS3

- fixed value k (instead of random)
- for two messages m, m' get signatures (r, s) and (r, s')

$$s - s' = k^{-1}(\text{hash}(m) + ra - \text{hash}(m') - ra)$$

$$\implies k = \frac{\text{hash}(m) - \text{hash}(m')}{s - s'}$$

Sony's failure with the PS3

- fixed value k (instead of random)
- for two messages m, m' get signatures (r, s) and (r, s')

$$s - s' = k^{-1}(\text{hash}(m) + ra - \text{hash}(m') - ra)$$
$$\implies k = \frac{\text{hash}(m) - \text{hash}(m')}{s - s'}$$

- also get secret key $a = (sk - \text{hash}(m))r^{-1} \bmod n$

Sony's failure with the PS3

- fixed value k (instead of random)
- for two messages m, m' get signatures (r, s) and (r, s')

$$s - s' = k^{-1}(\text{hash}(m) + ra - \text{hash}(m') - ra)$$
$$\Longrightarrow \quad k = \frac{\text{hash}(m) - \text{hash}(m')}{s - s'}$$

- also get secret key $a = (sk - \text{hash}(m))r^{-1} \bmod n$

Countermesaure Without Randomness – RFC 6979

- generate k from msg and a
  iterated use of HMAC (hash, concatenate, xor)
- k still is unique for every message

ECDSA – Overview

Overview

- public key is g^a, hence also based on DLP
- signature is pair of numbers

ECDSA – Overview

Overview

- public key is g^a, hence also based on DLP
- signature is pair of numbers

Lesson Learned

- even large corporations/libraries fail
- edge cases are important in adversarial setting
- follow the pseudo code

Post Quantum Cryptography

What to do, if Eve has a quantum computer and I don't.

Current Situation

When Quantum Computers Arrive

- quantum computers can solve factoring and DLP
- both RSA and every DH scheme get broken

Current Situation

When Quantum Computers Arrive

- quantum computers can solve factoring and DLP
- both RSA and every DH scheme get broken

Remark

Even with quantum computers, we do not know how to solve NP*-hard problems or break AES.*

Current Situation

When Quantum Computers Arrive

- quantum computers can solve factoring and DLP
- both RSA and every DH scheme get broken

Remark

Even with quantum computers, we do not know how to solve NP*-hard problems or break AES.*

New Crypto Schemes

- base crypto scheme on NP-hard problem, hard on average
- most common candidates:
  - lattice problems
  - multivariate polynomials
  - problems from coding theory

Lattice

Definition

Given a base of vectors $B = \{v_1, \ldots, v_n\}$, their lattice is

$$L(B) = \text{span}_{\mathbb{Z}}(B) = B\mathbb{Z}^n = \left\{ \sum_{i=1}^{n} a_i v_i : a_i \in \mathbb{Z} \right\}$$

- for simplicity $v_i \in \mathbb{Z}^n$

Lattice

Definition

Given a base of vectors $B = \{v_1, \ldots, v_n\}$, their lattice is

$$L(B) = \text{span}_{\mathbb{Z}}(B) = B\mathbb{Z}^n = \left\{ \sum_{i=1}^{n} a_i v_i : a_i \in \mathbb{Z} \right\}$$

- for simplicity $v_i \in \mathbb{Z}^n$

Properties

- bases A, B create same lattice if $A = UB$ for some $U \in \mathbb{Z}^{n \times n}$ with $\det U = \pm 1$ (U is unimodular)
- isomorphism $L \cong \mathbb{Z}^n$ for every lattice
  but isomorphism destroys angles and distances

Lattice Problems

Shortest Vector Problem (SVP)

- given L, find a vector $v \in L \setminus \{\mathbf{0}\}$ with $\|v\|$ minimal

Lattice Problems

Shortest Vector Problem (SVP)

- given L, find a vector $v \in L \setminus \{\mathbf{0}\}$ with $\|v\|$ minimal

Closest Vector Problem (CVP)

- given L and $u \in \mathbb{Z}^n$, find a vector $v \in L$, with $\|u - v\|$ minimal

Lattice Problems

Shortest Vector Problem (SVP)

- given L, find a vector $v \in L \setminus \{\mathbf{0}\}$ with $\|v\|$ minimal

Closest Vector Problem (CVP)

- given L and $u \in \mathbb{Z}^n$, find a vector $v \in L$, with $\|u - v\|$ minimal

Shortest Base Problem (SBP)

- given B, find base $B' = \{v'_1, \ldots, v'_n\}$ with $B\mathbb{Z}^n = B'\mathbb{Z}^n$ such that $\prod \|v'_i\|$ minimal

Lattice Problems

Shortest Vector Problem (SVP)

- given L, find a vector $v \in L \setminus \{\mathbf{0}\}$ with $\|v\|$ minimal

Closest Vector Problem (CVP)

- given L and $u \in \mathbb{Z}^n$, find a vector $v \in L$, with $\|u - v\|$ minimal

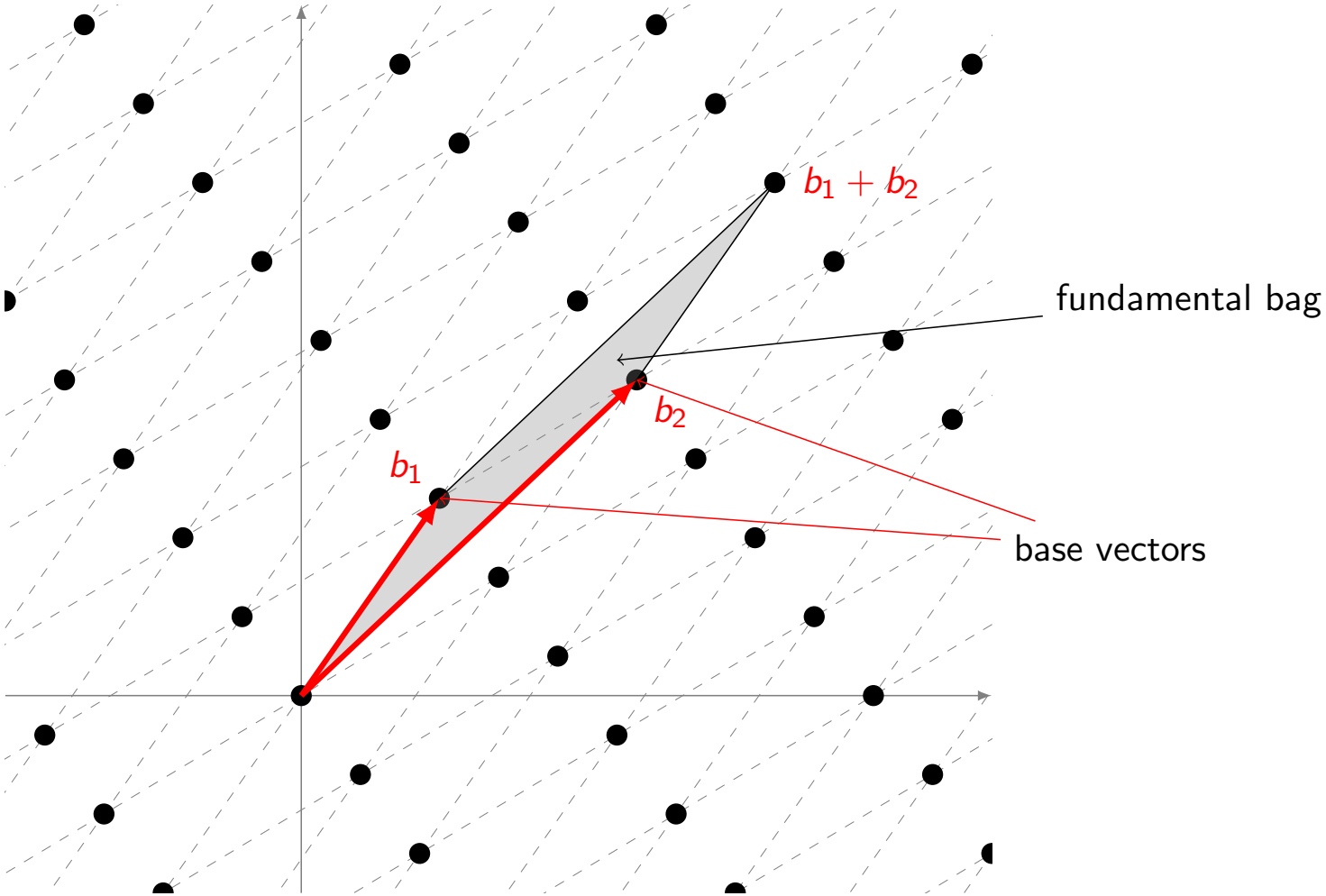
Shortest Base Problem (SBP)

- given B, find base $B' = \{v'_1, \ldots, v'_n\}$ with $B\mathbb{Z}^n = B'\mathbb{Z}^n$ such that $\prod \|v'_i\|$ minimal

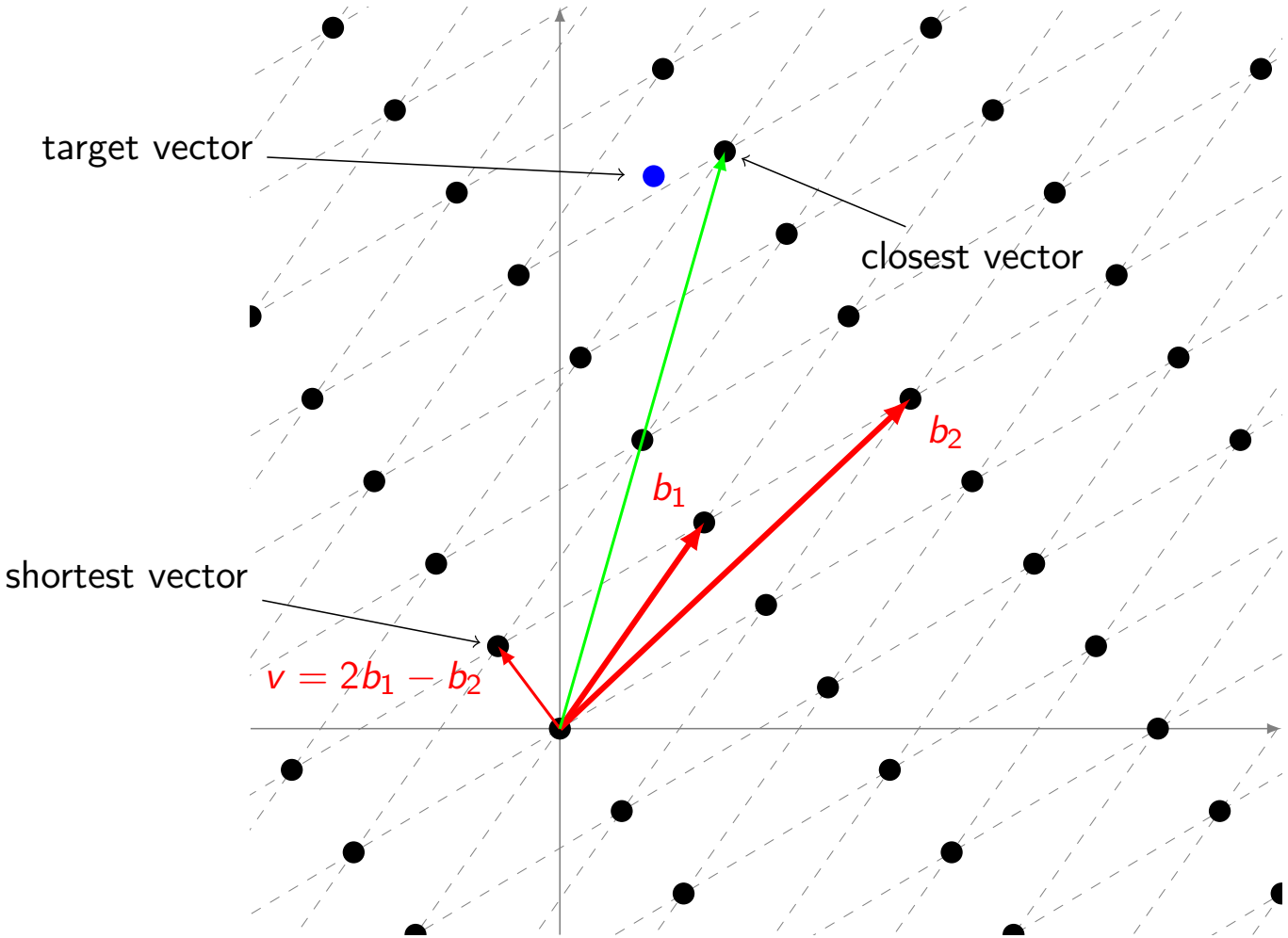
Hardness

- SVP NP-hard under randomised reduction
- CVP is NP-complete
- short base makes SVP and CVP significantly easier

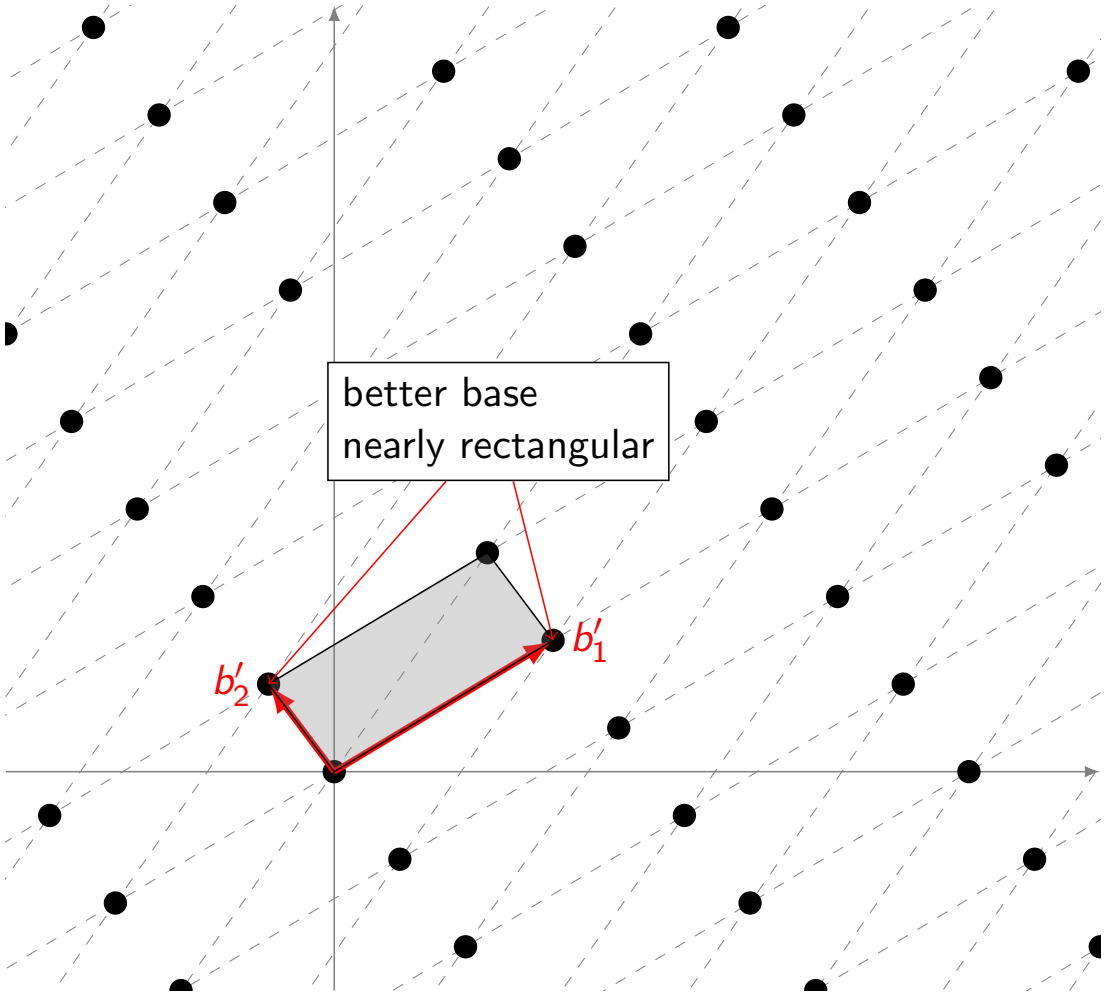
Lattices



Lattices



Lattices



Heuristic Solutions

CVP – Babai's Roundoff

- lattice $L = L(B)$
- given $u \in \mathbb{Z}^n$, find closest $v \in L$
- solve linear equation system $Bx = u$ in $\mathbb{Q}$
- round entries of x to get $v \in L$ via $v = B \cdot \text{round}(x)$

Heuristic Solutions

CVP – Babai's Roundoff

- lattice $L = L(B)$
- given $u \in \mathbb{Z}^n$, find closest $v \in L$
- solve linear equation system $Bx = u$ in $\mathbb{Q}$
- round entries of x to get $v \in L$ via $v = B \cdot \text{round}(x)$

$$B = \begin{pmatrix} 6 & 10 \\ 7 & 12 \end{pmatrix} \qquad u = \begin{pmatrix} 3.8 \\ 4.1 \end{pmatrix} \qquad \Longrightarrow \; x = \begin{pmatrix} 2.3 \\ -1 \end{pmatrix}$$

returns $v = (2, 2)$, but closest point is $(4, 4)$

Heuristic Solutions

CVP – Babai's Roundoff

- lattice $L = L(B)$
- given $u \in \mathbb{Z}^n$, find closest $v \in L$
- solve linear equation system $Bx = u$ in $\mathbb{Q}$
- round entries of x to get $v \in L$ via $v = B \cdot \text{round}(x)$

$$B = \begin{pmatrix} 6 & 10 \\ 7 & 12 \end{pmatrix} \qquad u = \begin{pmatrix} 3.8 \\ 4.1 \end{pmatrix} \qquad \Longrightarrow \; x = \begin{pmatrix} 2.3 \\ -1 \end{pmatrix}$$

returns $v = (2, 2)$, but closest point is $(4, 4)$

SBP/SVP — LLL Algorithm

- LLL algorithm gives reduced lattice
- shortest base vector can differ from optimum by exponential factor

From Lattices to Cryptography

Tasks

- math problem $\rightarrow$ crypto scheme/key exchange
- $\mathbb{Z}^n$ is unbounded
  - want something finite
  - what changes if we add some mod p?
- how to create "always hard instances"?
- actual parameters?

From Lattices to Cryptography

Tasks

- math problem $\rightarrow$ crypto scheme/key exchange
- $\mathbb{Z}^n$ is unbounded
  - want something finite
  - what changes if we add some mod p?
- how to create "always hard instances"?
- actual parameters?

What can go wrong?

current research

NTRU – (n-th Degree Truncated Polynomial Ring)

Overview

- proposed in 1997, relatively mature
- feasible key size, still unbroken, NIST post-quantum candidate

NTRU – (n-th Degree Truncated Polynomial Ring)

Overview

- proposed in 1997, relatively mature
- feasible key size, still unbroken, NIST post-quantum candidate

Parameters

- $n \in \mathbb{Z}$, computation in $R = \mathbb{Z}[x]/(x^n - 1)$, i.e. integer polynomials with $x^n = 1$

NTRU – (n-th Degree Truncated Polynomial Ring)

Overview

- proposed in 1997, relatively mature
- feasible key size, still unbroken, NIST post-quantum candidate

Parameters

- $n \in \mathbb{Z}$, computation in $R = \mathbb{Z}[x]/(x^n - 1)$, i.e. integer polynomials with $x^n = 1$
- coprime numbers p, q; standard $p = 3$, $q = 2^*$
- sets $\mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_r, \mathcal{L}_m \subseteq \mathbb{Z}[x]$ of polynomials with "small" coefficients, usually coeff.s $\{-1, 0, 1\}$

NTRU – (n-th Degree Truncated Polynomial Ring)

Overview

- proposed in 1997, relatively mature
- feasible key size, still unbroken, NIST post-quantum candidate

Parameters

- $n \in \mathbb{Z}$, computation in $R = \mathbb{Z}[x]/(x^n - 1)$, i.e. integer polynomials with $x^n = 1$
- coprime numbers p, q; standard $p = 3$, $q = 2^*$
- sets $\mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_r, \mathcal{L}_m \subseteq \mathbb{Z}[x]$ of polynomials with "small" coefficients, usually coeff.s $\{-1, 0, 1\}$

Warning

- Not all parameter sets work!
- notion of "correct" parameters, details later

see modulo as $\mathbb{Z}_p = \{\lfloor -p/2 \rfloor, \ldots, 0, \ldots, \lfloor p/2 \rfloor\}$

see modulo as $\mathbb{Z}_p = \{\lfloor -p/2 \rfloor, \ldots, 0, \ldots, \lfloor p/2 \rfloor\}$

Key Generation

- pick random $f, g \in R$ with small coefficients, $f \in \mathcal{L}_f, g \in \mathcal{L}_g$
- let $f_q = f^{-1} \bmod q$ and $f_p = f^{-1} \bmod p$
  solve linear equation systems; fail $\rightsquigarrow$ new f
- public key: $h := p \cdot f_q \cdot g \bmod q$
- secret key: f, f_p (g, f_q not needed any more)

see modulo as $\mathbb{Z}_p = \{\lfloor -p/2 \rfloor, \ldots, 0, \ldots, \lfloor p/2 \rfloor\}$

Key Generation

- pick random $f, g \in R$ with small coefficients, $f \in \mathcal{L}_f, g \in \mathcal{L}_g$
- let $f_q = f^{-1} \bmod q$ and $f_p = f^{-1} \bmod p$
  solve linear equation systems; fail $\rightsquigarrow$ new f
- public key: $h := p \cdot f_q \cdot g \bmod q$
- secret key: f, f_p (g, f_q not needed any more)

Encryption

- encode message as polynomial with small coefficients, $m \in \mathcal{L}_m$
- pick random $r \in R$ with small coefficients, $r \in \mathcal{L}_r$
- cipher $c = r \cdot h + m \bmod q$

see modulo as $\mathbb{Z}_p = \{\lfloor -p/2 \rfloor, \ldots, 0, \ldots, \lfloor p/2 \rfloor\}$

Key Generation

- pick random $f, g \in R$ with small coefficients, $f \in \mathcal{L}_f, g \in \mathcal{L}_g$
- let $f_q = f^{-1} \bmod q$ and $f_p = f^{-1} \bmod p$
  solve linear equation systems; fail $\rightsquigarrow$ new f
- public key: $h := p \cdot f_q \cdot g \bmod q$
- secret key: f, f_p (g, f_q not needed any more)

Encryption

- encode message as polynomial with small coefficients, $m \in \mathcal{L}_m$
- pick random $r \in R$ with small coefficients, $r \in \mathcal{L}_r$
- cipher $c = r \cdot h + m \bmod q$

Decryption

- $a = f \cdot c \bmod q$
- $m = f_p \cdot a \bmod p$

Why/When NTRU works?

- In decryption

$$a = f \cdot c \bmod q = f(rh + m) \bmod q = f(rpf_q g + m) \bmod q$$
$$= p \cdot r \cdot g + f \cdot m \bmod q$$

all polynomials of small coefficients

Why/When NTRU works?

- In decryption

$$a = f \cdot c \bmod q = f(rh + m) \bmod q = f(rpf_q g + m) \bmod q$$
$$= p \cdot r \cdot g + f \cdot m \bmod q$$

  all polynomials of small coefficients
- want that mod q does not do anything, then

$$f_p \cdot a \bmod p = p \cdot * + f_p \cdot f \cdot m \bmod p = m \bmod p = m$$

Why/When NTRU works?

- In decryption

$$a = f \cdot c \bmod q = f(rh + m) \bmod q = f(rpf_q g + m) \bmod q$$
$$= p \cdot r \cdot g + f \cdot m \bmod q$$

  all polynomials of small coefficients
- want that $\bmod q$ does not do anything, then

$$f_p \cdot a \bmod p = p \cdot * + f_p \cdot f \cdot m \bmod p = m \bmod p = m$$

- assume coefficients of f, g, r, m in $\{-1, 0, 1\}$,
  then for coefficients a_i of a (via product formula) have

$$|a_i| \leq pn + n = n(p+1) \overset{!}{<} \frac{q}{2}$$

Why/When NTRU works?

- In decryption

$$a = f \cdot c \bmod q = f(rh + m) \bmod q = f(rpf_q g + m) \bmod q$$
$$= p \cdot r \cdot g + f \cdot m \bmod q$$

  all polynomials of small coefficients
- want that $\bmod q$ does not do anything, then

$$f_p \cdot a \bmod p = p \cdot * + f_p \cdot f \cdot m \bmod p = m \bmod p = m$$

- assume coefficients of f, g, r, m in $\{-1, 0, 1\}$,
  then for coefficients a_i of a (via product formula) have

$$|a_i| \leq pn + n = n(p+1) \overset{!}{<} \frac{q}{2}$$

- hence $q > 2n(p+1)$ is a "correct" choice

NTRU and Lattices

Calculation mod $x^n - 1$ allows for special translation.

Translate polynomials into lattices

- polynomial = vector of its coefficients, also as matrix

$$v = \sum_{k=0}^{n-1} v_k x^k \cong \begin{pmatrix} v_0 \\ v_1 \\ \vdots \end{pmatrix} \cong \begin{pmatrix} v_0 & v_{n-1} & \cdots & v_1 \\ v_1 & v_0 & \cdots & v_2 \\ & & \ddots & \\ v_{n-1} & v_{n-2} & \cdots & v_0 \end{pmatrix}$$

NTRU and Lattices

Calculation mod $x^n - 1$ allows for special translation.

Translate polynomials into lattices

- polynomial = vector of its coefficients, also as matrix

$$v = \sum_{k=0}^{n-1} v_k x^k \cong \begin{pmatrix} v_0 \\ v_1 \\ \vdots \end{pmatrix} \cong \begin{pmatrix} v_0 & v_{n-1} & \cdots & v_1 \\ v_1 & v_0 & \cdots & v_2 \\ & & \ddots & \\ v_{n-1} & v_{n-2} & \cdots & v_0 \end{pmatrix}$$

- adding polynomials = adding vectors = adding matrices

NTRU and Lattices

Calculation $\mod x^n - 1$ allows for special translation.

Translate polynomials into lattices

- polynomial = vector of its coefficients, also as matrix

$$v = \sum_{k=0}^{n-1} v_k x^k \cong \begin{pmatrix} v_0 \\ v_1 \\ \vdots \end{pmatrix} \cong \begin{pmatrix} v_0 & v_{n-1} & \cdots & v_1 \\ v_1 & v_0 & \cdots & v_2 \\ & & \ddots & \\ v_{n-1} & v_{n-2} & \cdots & v_0 \end{pmatrix}$$

- adding polynomials = adding vectors = adding matrices
- multiplication of polynomials f, g:

$$\text{Matrix}(f) \cdot \text{Vector}(g) = \text{Vector}(f \cdot g)$$

Hence, we are in the realm of lattices.

NTRU and Lattice Problems

Break Key

f, g only have small entries

$$(f, g) \in \mathcal{L}\left(\begin{pmatrix} I_n & 0 \\ p^{-1}h & qI_n \end{pmatrix}\right) \subseteq \mathbb{Z}^{2n}$$

$f, g \in \{-1, 0, 1\}^n$, so we look for short vectors $\rightsquigarrow$ SVP

NTRU and Lattice Problems

Break Key

f, g only have small entries

$$(f, g) \in \mathcal{L}\left(\begin{pmatrix} I_n & 0 \\ p^{-1}h & qI_n \end{pmatrix}\right) \subseteq \mathbb{Z}^{2n}$$

$f, g \in \{-1, 0, 1\}^n$, so we look for short vectors $\rightsquigarrow$ SVP

Find Message

r, m only have small entries

$$(r, c - m) \in \mathcal{L}\left(\begin{pmatrix} I_n & 0 \\ h & qI_n \end{pmatrix}\right) \subseteq \mathbb{Z}^{2n}$$

$r, m \in \{-1, 0, 1\}^n$, so we look for a vector close to $(0, c)$ $\rightsquigarrow$ CVP

NTRU – Improvements

Selecting Polynomials

- additionally restrict polynomials,
  - $\mathcal{T}$ ternary polynomial, coefficients $\{-1, 0, 1\}$, degree $\leq n - 2$
  - $\mathcal{T}(d)$: additionally $\frac{d}{2}$ coeff.s 1, $\frac{d}{2}$ coeff.s -1, else 0
- let $f, r \in \mathcal{T}$ and $g, m \in \mathcal{T}(q/8 - 2)$ with $p = 3$, then

$$|a_i| \leq p \cdot (q/8 - 2) + q/8 - 2 = q/2 - 8 < \frac{q}{2}$$

NTRU – Improvements

Selecting Polynomials

- additionally restrict polynomials,
  - $\mathcal{T}$ ternary polynomial, coefficients $\{-1, 0, 1\}$, degree $\leq n - 2$
  - $\mathcal{T}(d)$: additionally $\frac{d}{2}$ coeff.s 1, $\frac{d}{2}$ coeff.s -1, else 0
- let $f, r \in \mathcal{T}$ and $g, m \in \mathcal{T}(q/8 - 2)$ with $p = 3$, then

$$|a_i| \leq p \cdot (q/8 - 2) + q/8 - 2 = q/2 - 8 < \frac{q}{2}$$

NTRU-HPS – Recommended Values

- $n = 501$ and $q = 2048$
- $n = 677$ and $q = 2048$
- $n = 821$ and $q = 4096$

good speed with high security, keys and cipher 900-1600 byte

NTRU – Summary

- basic form: public key cryptosystem (i.e. en-/decrypt)
- submitted version generates session keys
- based on other mathematical problem
  - shortest vector: break key
  - closest vector: find message
- believed to be quantum resistant
- faster than RSA/ECDH
- public keys larger than RSA
- only recently greater focus $\rightsquigarrow$ less researched

skip multivariate crypto

Multivariate Cryptography

Problem MQ – Multivariate Quadratic

Given: finite field K, polynomials $f_i \in K[x_1, \ldots, x_n]$ of degree 2

Task: find $x \in K^n$ with $f_i(x) = 0$ for all i

Multivariate Cryptography

Problem MQ – Multivariate Quadratic

Given: finite field K, polynomials $f_i \in K[x_1, \ldots, x_n]$ of degree 2

Task: find $x \in K^n$ with $f_i(x) = 0$ for all i

Hardness

- can encode SAT, easiest for $K = \mathbb{Z}_2$, via $x \wedge y = x \cdot y$ and $x \vee y = x + y - xy$ and auxiliary variables $\implies$ NP-hard

Multivariate Cryptography

Problem MQ – Multivariate Quadratic

Given: finite field K, polynomials $f_i \in K[x_1, \ldots, x_n]$ of degree 2

Task: find $x \in K^n$ with $f_i(x) = 0$ for all i

Hardness

- can encode SAT, easiest for $K = \mathbb{Z}_2$, via $x \wedge y = x \cdot y$ and $x \vee y = x + y - xy$ and auxiliary variables $\implies$ NP-hard

Example

take formula $\varphi = (x_1 \wedge x_2 \wedge \neg x_3) \vee (\neg x_2 \wedge x_3)$
to find satisfying assignment, solve system

$$y_1 = x_1 \cdot x_2 \qquad y_2 = y_1 \cdot (1 - x_3)$$
$$y_3 = (1 - x_2) \cdot x_3 \qquad 1 = y_2 + y_3 - y_2 \cdot y_3$$

Turning MQ into Cryptography

Basic Idea

additional secret information allows to solve hard problem

Turning MQ into Cryptography

Basic Idea

additional secret information allows to solve hard problem

Reformulation

finding root equivalent to

Given: $y_i \in K$, $f_i \in K[x_1, \ldots, x_n]$

Task: find $\boldsymbol{x} \in K^n$ with $y_i = f_i(\boldsymbol{x})$ for all i

Turning MQ into Cryptography

Basic Idea

additional secret information allows to solve hard problem

Reformulation

finding root equivalent to

Given: $y_i \in K$, $f_i \in K[x_1, \ldots, x_n]$

Task: find $\boldsymbol{x} \in K^n$ with $y_i = f_i(\boldsymbol{x})$ for all i

translate into cryptography

Keys: $P = (f_1, \ldots, f_m)$ – public key, P^{-1} – secret key

Encryption: y – cipher, x – message

Signing y – message, x – signature

Key Generation

- pick easily "invertible" polynomial system F
- pick two invertible affine (linear + shift) maps S, T
- public key $P = T \circ F \circ S$ (meaning $x \to T \to F \to S \rightsquigarrow P(x)$)
- secret key S, F, T, owner can compute

$$P^{-1} = S^{-1} \circ F^{-1} \circ T^{-1}$$

Key Generation

- pick easily "invertible" polynomial system F
- pick two invertible affine (linear + shift) maps S, T
- public key $P = T \circ F \circ S$ (meaning $x \to T \to F \to S \rightsquigarrow P(x)$)
- secret key S, F, T, owner can compute

$$P^{-1} = S^{-1} \circ F^{-1} \circ T^{-1}$$

Signatures

Sign: message m, signature $s = P^{-1}(m)$

Verify: check $m = P(s)$

Key Generation

- pick easily "invertible" polynomial system F
- pick two invertible affine (linear + shift) maps S, T
- public key $P = T \circ F \circ S$ (meaning $x \to T \to F \to S \rightsquigarrow P(x)$)
- secret key S, F, T, owner can compute

$$P^{-1} = S^{-1} \circ F^{-1} \circ T^{-1}$$

Signatures

Sign: message m, signature $s = P^{-1}(m)$

Verify: check $m = P(s)$

Encryption – several schemes outdated/broken!

Encrypt: message m, cipher $c = P(m)$

Decrypt: retrieve $m = P^{-1}(c)$

Multivariate Signatures

Key Observation

Signature just has to be some valid preimage under P.

Multivariate Signatures

Key Observation

Signature just has to be some valid preimage under P.

New Idea – Oil and Vinegar to construct F

- use $K^m \xrightarrow{T} K^m \xrightarrow{F} K^n \xrightarrow{S} K^n$

Multivariate Signatures

Key Observation

Signature just has to be some valid preimage under P.

New Idea – Oil and Vinegar to construct F

- use $K^m \xrightarrow{T} K^m \xrightarrow{F} K^n \xrightarrow{S} K^n$
- have n "oil" variables $\boldsymbol{x}$ and v "vinegar" variables $\boldsymbol{a}$, $m = n + v$
- never mix (multiply) oil with oil, then structure

$$y_i = \sum_{j,k} \gamma_{ijk} x_j a_k + \sum_{j,k} \lambda_{ijk} a_j a_k + \sum_j \xi_{ij} x_j + \sum_j \xi'_{ij} a_j + \delta_i$$

Multivariate Signatures

Key Observation

Signature just has to be some valid preimage under P.

New Idea – Oil and Vinegar to construct F

- use $K^m \xrightarrow{T} K^m \xrightarrow{F} K^n \xrightarrow{S} K^n$
- have n "oil" variables $\boldsymbol{x}$ and v "vinegar" variables $\boldsymbol{a}$, $m = n + v$
- never mix (multiply) oil with oil, then structure

$$y_i = \sum_{j,k} \gamma_{ijk} x_j a_k + \sum_{j,k} \lambda_{ijk} a_j a_k + \sum_j \xi_{ij} x_j + \sum_j \xi'_{ij} a_j + \delta_i$$

- fix random values for vinegar a_j
- solve linear equation system to get x_j
- yields preimage $(\boldsymbol{x}, \boldsymbol{a})$ for $\boldsymbol{y} = (y_1, \ldots, y_n)$

Selecting Parameters

Broken Cases

- initially $n = v$ (balanced), broken by Kipnis and Shamir in 1998 also works for $v \approx n$
- for $v \geq n^2$ and char $K = 2$, finding solution is feasible

Selecting Parameters

Broken Cases

- initially $n = v$ (balanced), broken by Kipnis and Shamir in 1998 also works for $v \approx n$
- for $v \geq n^2$ and char $K = 2$, finding solution is feasible

Unbalanced Oil and Vinegar (UOV)

- choose $v \geq 2n$
- parameters $\gamma, \lambda, \xi, \xi', \delta$ for F randomly
- $T \in K^m \to K^m$ affine, random,

Selecting Parameters

Broken Cases

- initially $n = v$ (balanced), broken by Kipnis and Shamir in 1998 also works for $v \approx n$
- for $v \geq n^2$ and char $K = 2$, finding solution is feasible

Unbalanced Oil and Vinegar (UOV)

- choose $v \geq 2n$
- parameters $\gamma, \lambda, \xi, \xi', \delta$ for F randomly
- $T \in K^m \to K^m$ affine, random, chance to be invertible

$$\frac{|\operatorname{GL}_m(q)|}{q^{m \cdot m}} = \prod_{k=0}^{m-1} \left(1 - q^{k-m}\right) \xrightarrow{m \to \infty} 28.8\ldots\% \qquad \text{for } q = 2$$

- similar chance for S and for computing x_j for random a_j

Selecting Parameters

Broken Cases

- initially $n = v$ (balanced), broken by Kipnis and Shamir in 1998 also works for $v \approx n$
- for $v \geq n^2$ and char $K = 2$, finding solution is feasible

Unbalanced Oil and Vinegar (UOV)

- choose $v \geq 2n$
- parameters $\gamma, \lambda, \xi, \xi', \delta$ for F randomly
- $T \in K^m \to K^m$ affine, random, chance to be invertible

$$\frac{|\operatorname{GL}_m(q)|}{q^{m \cdot m}} = \prod_{k=0}^{m-1} \left(1 - q^{k-m}\right) \xrightarrow{m \to \infty} 28.8\ldots\% \qquad \text{for } q = 2$$

- similar chance for S and for computing x_j for random a_j
- problem: key size $\mathcal{O}(m^3 \log q)$

Example Scheme – Rainbow

Rainbow

- Finalist in NIST competition for post-quantum signature
- uses multivariante quadratic polynomials
- map F has cascading structure, instead of 1 lin.eq.sys. solve several smaller ones, block-diagonal structure

Example Scheme – Rainbow

Rainbow

- Finalist in NIST competition for post-quantum signature
- uses multivariante quadratic polynomials
- map F has cascading structure, instead of 1 lin.eq.sys. solve several smaller ones, block-diagonal structure
- in highest security:
  - 1.38 MB private key
  - 1.89 MB public key
  - 212 B signature
  - sign/verify extremely fast, key generation moderate

Example Scheme – Rainbow

Rainbow

- Finalist in NIST competition for post-quantum signature
- uses multivariante quadratic polynomials
- map F has cascading structure, instead of 1 lin.eq.sys. solve several smaller ones, block-diagonal structure
- in highest security:
  - 1.38 MB private key
  - 1.89 MB public key
  - 212 B signature
  - sign/verify extremely fast, key generation moderate
- variant: 60 B private key, 523 kB public key, but sign/verify much longer (more than x100)

Example Scheme – Rainbow

Rainbow

- Finalist in NIST competition for post-quantum signature
- uses multivariante quadratic polynomials
- map F has cascading structure, instead of 1 lin.eq.sys. solve several smaller ones, block-diagonal structure
- in highest security:
  - 1.38 MB private key
  - 1.89 MB public key
  - 212 B signature
  - sign/verify extremely fast, key generation moderate
- variant: 60 B private key, 523 kB public key, but sign/verify much longer (more than x100)

recently broken

Further Post-Quantum Candidates

- open competition by NIST
- 3rd round finished in 2022

Further Post-Quantum Candidates

- open competition by NIST
- 3rd round finished in 2022
- key exchange (KEM: key encapsulation method)
  - lattice: NTRU, Kyber, Saber
  - code: Classic McEliece

Further Post-Quantum Candidates

- open competition by NIST
- 3rd round finished in 2022
- key exchange (KEM: key encapsulation method)
  - lattice: NTRU, Kyber, Saber
  - code: Classic McEliece
- signatures
  - lattice: Dilithium, Falcon
  - MQ: ~~Rainbow~~
- some alternative candidates (in part of other classes)
  worse in: security/ time/ communication size
- trade-off between sizes of public key, secret key, signature/cipher
  but also time and power consumption

Security Enhancements

What is lacking?

- many crypto primitives focus on OW-CPA
- preferred security IND-CCA2
- PKCS#1 already does that, but RSA-specific
- general transformation of weaker scheme into IND-CCA2

Security Enhancements

What is lacking?

- many crypto primitives focus on OW-CPA
- preferred security IND-CCA2
- PKCS#1 already does that, but RSA-specific
- general transformation of weaker scheme into IND-CCA2

Solution – Fujisaki-Okamoto-Transformation

- generic transformation
- essentially a hybrid system (PKC + AES)
- need hash and symmetric encryption
- transform OW-CPA into IND-CCA2

Perfect Forward Secrecy

Scenario

- attacker captures your traffic over some time
- at later point gets access to private key

Perfect Forward Secrecy

Scenario

- attacker captures your traffic over some time
- at later point gets access to private key

Definition (Perfect Forward Secrecy)

compromise of long-term keys does not compromise past session keys

Perfect Forward Secrecy

Scenario

- attacker captures your traffic over some time
- at later point gets access to private key

Definition (Perfect Forward Secrecy)

compromise of long-term keys does not compromise past session keys

Method

- each session DH-handshake, forgotten afterwards
- long term key for signatures
- sign your part of handshake, to avoid man-in-the-middle

Perfect Forward Secrecy

Scenario

- attacker captures your traffic over some time
- at later point gets access to private key

Definition (Perfect Forward Secrecy)

compromise of long-term keys does not compromise past session keys

Method

- each session DH-handshake, forgotten afterwards
- long term key for signatures
- sign your part of handshake, to avoid man-in-the-middle

TLS 1.3 does this (TLS $\leq$1.2: optionally), Signal-protocol as well

Outlook

Some topics we left out:

Outlook

Some topics we left out:

Secret sharing

- split one secret across n people
- secret can be recovered if $\geq k$ people pair up
- e.g. via polynomial interpolation

Outlook

Some topics we left out:

Secret sharing

- split one secret across n people
- secret can be recovered if $\geq k$ people pair up
- e.g. via polynomial interpolation

Public Key Infrastructure

- If Eve controls network, how to avoid man-in-the-middle?
- How does Bob know, Alice's key was not changed?
- part of "Grundlagen der Rechnersicherheit"

Outlook

Some topics we left out:

Secret sharing

- split one secret across n people
- secret can be recovered if $\geq k$ people pair up
- e.g. via polynomial interpolation

Public Key Infrastructure

- If Eve controls network, how to avoid man-in-the-middle?
- How does Bob know, Alice's key was not changed?
- part of "Grundlagen der Rechnersicherheit"

Zero Knowledge Proofs

- Alice shows, she knows secret, without revealing secret

Larger Practical Use Cases

Signal-Protocol

- double ratchet method
- handling asynchronous communications and group chats

Larger Practical Use Cases

Signal-Protocol

- double ratchet method
- handling asynchronous communications and group chats

SSH

- authentication via public key (RFC 4252)
- get random token, have to sign

Larger Practical Use Cases

Signal-Protocol

- double ratchet method
- handling asynchronous communications and group chats

SSH

- authentication via public key (RFC 4252)
- get random token, have to sign

Telegram

- own protocol

Larger Practical Use Cases

Signal-Protocol

- double ratchet method
- handling asynchronous communications and group chats

SSH

- authentication via public key (RFC 4252)
- get random token, have to sign

Telegram

- own protocol

Generally, field with lot of ongoing research...

The End ...

Happy Hacking!

I hope you had fun.
Maybe see you at some CTF ;)