# Public Key Cryptography 

Henning Seidler

July 12, 2023

## 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" or "Specialisation Large" cannot be examined alone (unless exchange student)
- exam at the end of the module (about PKC and another course)
- 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, 6 pm - 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!


## 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)

June/July 2023: FaustCTF
7.7.2023: CryptoCTF
17.6.2023 LNDW


## 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:


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


## What is Cryptography?

cryptography: (in strict sense) design of cryptosystems cryptoanalysis: breaking encryption
cryptology: both, but often "cryptography" used instead

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


## What is Cryptography?

cryptography: (in strict sense) design of cryptosystems cryptoanalysis: breaking encryption
cryptology: both, but often "cryptography" used instead

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

```
Kerckhoff's Principle (Open Design)
enc and dec are known,
only key }\mp@subsup{k}{\mathrm{ dec }}{}\mathrm{ secret (and }\mp@subsup{k}{\mathrm{ enc }}{}\mathrm{ if both same)
```


## Mathematical Model

## Definition (Cryptosystem)

A cryptosystem is a quintuple ( $P, C, K$, enc, 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
- enc : $P \times K \sim C$ is the encryption relation (not necessarily a map)
- dec : $C \times K \rightarrow P$ is the decryption function
- $\forall m \in P, k \in K$. dec $(\operatorname{enc}(m, k), k)=m$, or $\forall m \in P,\left(k_{\text {dec }}, k_{\text {enc }}\right) \in K . \operatorname{dec}\left(\operatorname{enc}\left(m, k_{\text {enc }}\right), k_{\text {dec }}\right)=m$
- enc, dec are efficiently computable


## Mathematical Model

## Definition (Cryptosystem)

A cryptosystem is a quintuple ( $P, C, K$, enc, 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
- enc : $P \times K \sim C$ is the encryption relation (not necessarily a map)
- dec : $C \times K \rightarrow P$ is the decryption function
- $\forall m \in P, k \in K$. dec $(\operatorname{enc}(m, k), k)=m$, or $\forall m \in P,\left(k_{\text {dec }}, k_{\text {enc }}\right) \in K . \operatorname{dec}\left(\operatorname{enc}\left(m, k_{\text {enc }}\right), k_{\text {dec }}\right)=m$
- enc, 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


## Caesar-Cipher


by Albert Uderzo,
taken from
https://asterix.fandom.com/de/wiki/Julius_C\�\�sar

## 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 $\sim$ 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

## Ring of integers modulo $n$

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


## Ring of integers modulo $n$

- 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


## Ring of integers modulo $n$

- 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 $a>0$


## Ring of integers modulo $n$

- 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 $\operatorname{gcd}(a, n)=1$
- if $n$ prime, works for all $0<a<n$


## Example Ring - CPU/ALU

- modern CPU uses 64 Bit $\leadsto$ 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 $\leadsto$ 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 $\bar{a}$ is a with all bits flipped,
- $a+\bar{a}=1 \ldots 1$ (in every bit add 0 and 1 )
- $a+\bar{a}+1=(1) 0 \ldots 0=0$ (overflow)
- hence $-a=\bar{a}+1$


## 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)
$\checkmark$ 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 $\sim \mathcal{O}(b)$

## Algorithms

Need algorithms for the following: ( $b$ bits input)
$\checkmark$ addition, subtraction,
$\checkmark$ 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 $\sim \mathcal{O}(b)$

## Multiplication

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


## Division <br> reduce to multiplication, same complexity

## Algorithms

Theorem (Extended Euclidian Algorithm)
For all $a, b \in \mathbb{Z}$ there are $\lambda, \mu \in \mathbb{Z}$ such that

$$
\lambda a+\mu b=\operatorname{gcd}(a, b)
$$

## Algorithms

Theorem (Extended Euclidian Algorithm)
For all $a, b \in \mathbb{Z}$ there are $\lambda, \mu \in \mathbb{Z}$ such that

$$
\lambda a+\mu b=\operatorname{gcd}(a, b)
$$

```
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 $\lambda, \mu \in \mathbb{Z}$ such that

$$
\lambda a+\mu b=\operatorname{gcd}(a, b)
$$

def $\operatorname{EEA}(\mathrm{a}, \mathrm{b})$ :
if $\mathrm{b}==0$ : return ( $\mathrm{a}, 1,0$ )
$\mathrm{d}, \mathrm{s}, \mathrm{t}=\operatorname{EEA}(\mathrm{b}, \mathrm{a} \% \mathrm{~b})$
return ( $d, \mathrm{t}, \mathrm{s}-(\mathrm{a} / / \mathrm{b}) * \mathrm{t})$
Modular Inverse in $\mathbb{Z}_{n}$

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


## Algorithms

Theorem (Extended Euclidian Algorithm)
For all $a, b \in \mathbb{Z}$ there are $\lambda, \mu \in \mathbb{Z}$ such that

$$
\lambda a+\mu b=\operatorname{gcd}(a, b)
$$

def $\operatorname{EEA}(\mathrm{a}, \mathrm{b})$ :
if $b==0:$ return ( $a, 1,0$ )
$\mathrm{d}, \mathrm{s}, \mathrm{t}=\operatorname{EEA}(\mathrm{b}, \mathrm{a} \% \mathrm{~b})$
return (d, t, s - (a//b) * t)
Modular Inverse in $\mathbb{Z}_{n}$

- assume $\operatorname{gcd}(a, n)=1$, (always works if $n$ prime and $0<a<n$ )
- compute $d, \lambda, \mu=\operatorname{EEA}(a, n)$, clearly $d=1$
- $1=\lambda \cdot a+\mu \cdot n$ means $\lambda \cdot a \equiv 1 \bmod n$
- so $\lambda=a^{-1}$ 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

$$
a_{i} \equiv x \quad \bmod n_{i} \quad i=1, \ldots, k
$$

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

$$
a_{i} \equiv x \quad \bmod n_{i} \quad i=1, \ldots, k
$$

has a unique solution $0 \leq x<\prod n_{i}$.
Algorithm for 2 congruences:

$$
\begin{array}{cl}
\operatorname{EEA}\left(n_{1}, n_{2}\right) \sim & 1=s \cdot n_{1}+t \cdot n_{2} \\
\quad \text { solution } & x:=a_{2} \cdot s \cdot n_{1}+a_{1} \cdot t \cdot n_{2}
\end{array}
$$

continue recursively with: $a^{\prime}=x$ and $n^{\prime}=n_{1} \cdot n_{2}$

## Example (CRT)

- Consider the system

$$
\begin{array}{ll}
x \equiv 1 & \bmod 3 \\
x \equiv 2 & \bmod 5 \\
x \equiv 3 & \bmod 7
\end{array}
$$

## Example (CRT)

- Consider the system

$$
\begin{array}{ll}
x \equiv 1 & \bmod 3 \\
x \equiv 2 & \bmod 5 \\
x \equiv 3 & \bmod 7
\end{array}
$$

- combine first two:

$$
\begin{array}{rlrl}
\operatorname{EEA}(3,5) & =(1,2,-1) & \text { as } 1=2 \cdot 3+(-1) \cdot 5 \\
& \sim a^{\prime} & =2 \cdot 2 \cdot 3+1 \cdot(-1) \cdot 5=7 &
\end{array}
$$

## Example (CRT)

- Consider the system

$$
\begin{array}{ll}
x \equiv 1 & \bmod 3 \\
x \equiv 2 & \bmod 5 \\
x \equiv 3 & \bmod 7
\end{array}
$$

- combine first two:

$$
\begin{array}{rlrl}
\operatorname{EEA}(3,5) & =(1,2,-1) & \text { as } 1=2 \cdot 3+(-1) \cdot 5 \\
& \sim a^{\prime} & =2 \cdot 2 \cdot 3+1 \cdot(-1) \cdot 5=7 &
\end{array}
$$

- reduced system, continue recursively

$$
\begin{array}{ll}
x \equiv 7 & \bmod 15 \\
x \equiv 3 & \bmod 7
\end{array}
$$

## Fermat's Little Theorem

## Theorem

Let $p$ prime, $a$ arbitrary, then $a^{p} \equiv a \bmod p$.

## Fermat's Little Theorem

Theorem
Let $p$ prime, $a$ arbitrary, then $a^{p} \equiv a \bmod p$.

Proof by induction on a:
Base: $a=0 \checkmark$

## Fermat's Little Theorem

## Theorem

Let $p$ prime, $a$ arbitrary, then $a^{p} \equiv a \bmod p$.
Proof by induction on $a$ :
Base: $a=0 \checkmark$
Step:

$$
\begin{aligned}
(a+1)^{p} & =\sum_{k=0}^{p}\binom{p}{k} a^{k}=a^{p}+1+\sum_{k=1}^{p-1} \underbrace{\frac{p(p-1) \ldots(p-k+1)}{1 \cdot 2 \cdot \ldots \cdot k}}_{p \text { divides this }} a^{k} \\
& \equiv a^{p}+1 \stackrel{\text { IH }}{=} a+1 \quad \bmod p
\end{aligned}
$$

## Fermat's Little Theorem

## Theorem

Let $p$ prime, $a$ arbitrary, then $a^{p} \equiv a \bmod p$.

Proof by induction on $a$ :
Base: $a=0 \checkmark$
Step:

$$
\begin{aligned}
(a+1)^{p} & =\sum_{k=0}^{p}\binom{p}{k} a^{k}=a^{p}+1+\sum_{k=1}^{p-1} \frac{p(p-1) \ldots(p-k+1)}{1 \cdot 2 \cdot \ldots \cdot k} a^{k} \\
& \equiv a^{p}+1 \stackrel{\text { IH }}{=} a+1 \quad \bmod p
\end{aligned}
$$

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

## Euler's Theorem

## Generalise Fermat's Little Theorem:

## Definition (Euler's Phi-Function)

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

## Euler's Theorem

## Generalise Fermat's Little Theorem:

Definition (Euler's Phi-Function)

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

Lemma (How to compute $\varphi(n)$ ?)
Let $n=\prod p_{i}^{e_{i}}$ factorisation. Then $\varphi(n)=\prod\left(p_{i}-1\right) \cdot p_{i}^{e_{i}-1}$.

## Euler's Theorem

Generalise Fermat's Little Theorem:
Definition (Euler's Phi-Function)

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

Lemma (How to compute $\varphi(n)$ ?)
Let $n=\prod p_{i}^{e_{i}}$ factorisation. Then $\varphi(n)=\prod\left(p_{i}-1\right) \cdot p_{i}^{e_{i}-1}$.

Theorem (Euler)
Let $n \geq 2$ and $a \in \mathbb{Z}_{n}^{*}$. Then $a^{\varphi(n)} \equiv 1 \bmod n$.

## Euler's Theorem

Generalise Fermat's Little Theorem:
Definition (Euler's Phi-Function)

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

Lemma (How to compute $\varphi(n)$ ?)
Let $n=\prod p_{i}^{e_{i}}$ factorisation. Then $\varphi(n)=\prod\left(p_{i}-1\right) \cdot p_{i}^{e_{i}-1}$.
Theorem (Euler)
Let $n \geq 2$ and $a \in \mathbb{Z}_{n}^{*}$. Then $a^{\varphi(n)} \equiv 1 \bmod n$.
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)

$$
\begin{aligned}
\pi(100) & = \\
\pi(10000) & =
\end{aligned}
$$

## 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)

$$
\begin{aligned}
\pi(100) & =25 \\
\pi(10000) & =1229
\end{aligned}
$$

$\sim 1 / 4$ of numbers
$\sim 1 / 8$ of numbers

RSA
$27 / 217$

- THE classic in Public Key Cryptography besides Diffie-Hellman key-exchange ( $\sim$ later section)
- published April, 1977
- simple design, scheme yet unbroken
- THE classic in Public Key Cryptography besides Diffie-Hellman key-exchange ( $\sim$ later section)
- published April, 1977
- simple design, scheme yet unbroken
- named after
- Ron Rivest
- Adi Shamir
- Leonard Adleman
- featured in Martin Gardner's "Mathematical Games", Aug 1977; including the first RSA-challenge (129 decimal digits, 100\$),
- THE classic in Public Key Cryptography besides Diffie-Hellman key-exchange ( $\sim$ later section)
- published April, 1977
- simple design, scheme yet unbroken
- named after
- Ron Rivest
- Adi Shamir
- Leonard Adleman
- 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
- $n:=p \cdot q \Longrightarrow \varphi(n)=(p-1)(q-1)$
- choose e coprime to $\varphi(n)$
- $d:=e^{-1} \bmod \varphi(n)$ (extended Euclidean Algorithm)


## RSA

## Setup:

- $p, q$ primes
- $n:=p \cdot q \Longrightarrow \varphi(n)=(p-1)(q-1)$
- 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
- $n:=p \cdot q \Longrightarrow \varphi(n)=(p-1)(q-1)$
- 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 $m=\left(m^{e}\right)^{d} \bmod n$.

## Correctness of RSA

## Theorem

For every message $0 \leq m<n$ we have $m=\left(m^{e}\right)^{d} \bmod n$. slightly wrong "proof".

$$
m^{e d} \equiv m^{1+k \varphi(n)} \equiv m \cdot\left(m^{\varphi(n)}\right)^{k} \equiv m \cdot 1^{k} \equiv m \quad \bmod n
$$

by Euler.

## Correctness of RSA

## Theorem

For every message $0 \leq m<n$ we have $m=\left(m^{e}\right)^{d} \bmod n$.
slightly wrong "proof".

$$
m^{e d} \equiv m^{1+k \varphi(n)} \equiv m \cdot\left(m^{\varphi(n)}\right)^{k} \equiv m \cdot 1^{k} \equiv m \quad \bmod n
$$

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

## Correctness of RSA

## Theorem

For every message $0 \leq m<n$ we have $m=\left(m^{e}\right)^{d} \bmod n$.
slightly wrong "proof".

$$
m^{e d} \equiv m^{1+k \varphi(n)} \equiv m \cdot\left(m^{\varphi(n)}\right)^{k} \equiv m \cdot 1^{k} \equiv m \quad \bmod n
$$

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

## Proof.

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

## Correctness of RSA

## Theorem

For every message $0 \leq m<n$ we have $m=\left(m^{e}\right)^{d} \bmod n$.
slightly wrong "proof".

$$
m^{e d} \equiv m^{1+k \varphi(n)} \equiv m \cdot\left(m^{\varphi(n)}\right)^{k} \equiv m \cdot 1^{k} \equiv m \quad \bmod n
$$

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

## Proof.

If $p \mid m$, then $m^{\text {ed }} \equiv 0 \equiv m \bmod p$. Else
$m^{e d} \equiv m^{1+k(q-1)(p-1)} \equiv m \cdot\left(m^{p-1}\right)^{k(q-1)} \equiv m \cdot 1^{k(q-1)} \equiv m \bmod p$
So $m^{e d} \equiv m \bmod p$.

## Correctness of RSA

## Theorem

For every message $0 \leq m<n$ we have $m=\left(m^{e}\right)^{d} \bmod n$.
slightly wrong "proof".

$$
m^{e d} \equiv m^{1+k \varphi(n)} \equiv m \cdot\left(m^{\varphi(n)}\right)^{k} \equiv m \cdot 1^{k} \equiv m \quad \bmod n
$$

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

## Proof.

If $p \mid m$, then $m^{\text {ed }} \equiv 0 \equiv m \bmod p$. Else
$m^{e d} \equiv m^{1+k(q-1)(p-1)} \equiv m \cdot\left(m^{p-1}\right)^{k(q-1)} \equiv m \cdot 1^{k(q-1)} \equiv m \bmod p$
So $m^{e d} \equiv m \bmod p$. Analogue for $q$.
Hence $m^{e d} \equiv m \bmod n$ by CRT.

## RSA Details

## Generating primes of $B$ bit

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


## RSA Details

## Generating primes of $B$ bit

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


## RSA Details

## Generating primes of $B$ bit

- generate random bit sequence $p=p_{B-1} \ldots p_{1} 1$ (last bit 1 ) (Random Number Generators $\sim$ "Cryptography for Security")
- test, whether $p$ is (probably) prime ( $\sim$ 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 $p=p_{B-1} \ldots p_{1} 1$ (last bit 1 ) (Random Number Generators $\sim$ "Cryptography for Security")
- test, whether $p$ is (probably) prime ( $\sim$ 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 $e=65537=2^{16}+1=0 \times 10001$
- does not work $\Longrightarrow$ new primes


## RSA Details

## Generating primes of $B$ bit

- generate random bit sequence $p=p_{B-1} \ldots p_{1} 1$ (last bit 1 ) (Random Number Generators $\sim$ "Cryptography for Security")
- test, whether $p$ is (probably) prime ( $\sim$ 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 $e=65537=2^{16}+1=0 \times 10001$
- does not work $\Longrightarrow$ new primes
- alternatively:

$$
e \in\{3,5,17,257,65537\}=\left\{2^{2^{k}}+1: k=0, \ldots, 4\right\}
$$

## RSA Details

## Generating primes of $B$ bit

- generate random bit sequence $p=p_{B-1} \ldots p_{1} 1$ (last bit 1 ) (Random Number Generators $\sim$ "Cryptography for Security")
- test, whether $p$ is (probably) prime ( $\sim$ 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 $e=65537=2^{16}+1=0 \times 10001$
- does not work $\Longrightarrow$ new primes
- alternatively:

$$
e \in\{3,5,17,257,65537\}=\left\{2^{2^{k}}+1: k=0, \ldots, 4\right\}
$$

Fermat primes: coprime iff $e \nmid \varphi(n)$,
$e=10 \ldots 01_{2}$, only $2^{k}+1 \leq 17$ multiplications $\sim$ fast

## Modular Exponentiation

Need to compute $a^{b} \bmod n$

## Modular Exponentiation

Need to compute $a^{b} \bmod n$
Naive approach

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


## Modular Exponentiation

Need to compute $a^{b} \bmod n$
Naive approach

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

Square-and-Multiply
1: function $\operatorname{Pow}(a, b, n)$
2: $\quad c \leftarrow 1$
3: $\quad$ for $i=\log b, \ldots, 0$ do
4: $\quad c \leftarrow c^{2} \bmod n$
5:
if $b_{i}=1$ then
$\triangleright i f(b \&(1 \ll i))$
6: $\quad c \leftarrow c \cdot a \bmod n$
7: return $c$

## Modular Exponentiation

Need to compute $a^{b} \bmod n$
Naive approach

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

Square-and-Multiply
1: function $\operatorname{Pow}(a, b, n)$
2: $\quad c \leftarrow 1$
3: $\quad$ for $i=\log b, \ldots, 0$ do
4: $\quad c \leftarrow c^{2} \bmod n$
5: $\quad$ if $b_{i}=1$ then
$\triangleright i f(b \&(1 \ll i))$
6: $\quad c \leftarrow c \cdot a \bmod n$
7: 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 $\mathbb{Z}_{n}$ :

$$
\begin{array}{ll}
c:=102 & \\
c:=102^{2} & =1286 \\
c:=1286^{2} & =3438 \\
c:=3438^{2} & =2916 \\
c:=2916^{2} & =521 \\
c:=521 \cdot 102 & =2993
\end{array}
$$

$$
m^{1} \bmod n
$$

$$
m^{2} \bmod n
$$

$$
m^{4} \bmod n
$$

$$
m^{8} \bmod n
$$

$$
m^{16} \bmod n
$$

$$
m^{17} \bmod n
$$

In total: 5 multiplications

## Further Optimisation

During setup also compute (once)

$$
d_{p}=d \bmod p-1 \quad d_{q}=d \bmod q-1 \quad q_{\mathrm{inv}}=q^{-1} \bmod p
$$

## Further Optimisation

During setup also compute (once)

$$
d_{p}=d \bmod p-1 \quad d_{q}=d \bmod q-1 \quad q_{\mathrm{inv}}=q^{-1} \bmod p
$$

decryption

$$
\begin{array}{cc}
c_{p}=c \bmod p & c_{q}=c \bmod q \\
m_{p}=c_{p}^{d_{p}} \bmod p & m_{q}=c_{q}^{d_{q}} \bmod q \\
h=q_{\mathrm{inv}}\left(m_{p}-m_{q}\right) \bmod p \\
m=m_{q}+h q \bmod n
\end{array}
$$

## Further Optimisation

During setup also compute (once)

$$
d_{p}=d \bmod p-1 \quad d_{q}=d \bmod q-1 \quad q_{\mathrm{inv}}=q^{-1} \bmod p
$$

decryption

$$
\begin{aligned}
& c_{p}=c \bmod p \\
& c_{q}=c \bmod q \\
& m_{p}=c_{p}^{d_{p}} \bmod p \\
& m_{q}=c_{q}^{d_{q}} \bmod q \\
& h=q_{\mathrm{inv}}\left(m_{p}-m_{q}\right) \bmod p \\
& m=m_{q}+h q \bmod n
\end{aligned}
$$

Proof of correctness.

$$
\begin{array}{ll}
c^{d} \equiv c_{q}^{k \cdot(q-1)+d_{q}} \equiv c_{q}^{d_{q}} \equiv m_{q} \equiv m_{q}+h q \equiv m & \bmod q \\
c^{d} \equiv c_{p}^{d_{p}} \equiv m_{p} \equiv m_{q}+q q^{-1}\left(m_{p}-m_{q}\right) \equiv m_{q}+h q \equiv m & \bmod p
\end{array}
$$

## Further Optimisation

During setup also compute (once)

$$
d_{p}=d \bmod p-1 \quad d_{q}=d \bmod q-1 \quad q_{\mathrm{inv}}=q^{-1} \bmod p
$$

decryption

$$
\begin{gathered}
c_{p}=c \bmod p \\
m_{p}=c_{p}^{d_{p}} \bmod p \\
m_{q}=c \bmod q \\
h=c_{q}^{d_{q}} \bmod q \\
m=m_{q}+h q \bmod n
\end{gathered}
$$

Proof of correctness.

$$
\begin{array}{ll}
c^{d} \equiv c_{q}^{k \cdot(q-1)+d_{q}} \equiv c_{q}^{d_{q}} \equiv m_{q} \equiv m_{q}+h q \equiv m & \bmod q \\
c^{d} \equiv c_{p}^{d_{p}} \equiv m_{p} \equiv m_{q}+q q^{-1}\left(m_{p}-m_{q}\right) \equiv m_{q}+h q \equiv m & \bmod p
\end{array}
$$

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

## Complexity Analysis

$$
\begin{gathered}
c_{p}=c \bmod p \\
m_{p}=c_{p}^{d_{p}} \bmod p \\
m_{q}=c \bmod q \\
h=c_{q}^{d_{q}} \bmod q \\
m=m_{q}+h q \bmod n
\end{gathered}
$$

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

## Complexity Analysis

$$
\begin{gathered}
c_{p}=c \bmod p \\
m_{p}=c_{p}^{d_{p}} \bmod p \\
m_{q}=c \bmod q \\
h=c_{q}^{d_{q}} \bmod q \\
m=m_{q}+h q \bmod n
\end{gathered}
$$

Assume $\log d=\log n=B$, and $\log p=\log q=\frac{B}{2}$, $d, d_{p}, d_{q}$ equally many 0 s and 1 s normal $\sim \frac{3}{2} B$ mult. of size $B$

## Complexity Analysis

$$
\begin{aligned}
c_{p} & =c \bmod p & c_{q} & =c \bmod q \\
m_{p} & =c_{p}^{d_{p}} \bmod p & m_{q} & =c_{q}^{d_{q}} \bmod q
\end{aligned}
$$

$$
\begin{aligned}
h & =q_{\mathrm{inv}}\left(m_{p}-m_{q}\right) \bmod p \\
m & =m_{q}+h q \bmod n
\end{aligned}
$$

Assume $\log d=\log n=B$, and $\log p=\log q=\frac{B}{2}$, $d, d_{p}, d_{q}$ equally many 0 s and 1 s
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 $\frac{B}{2}, 1 \bmod$ 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 $\sim$ another factor 2

## Example

- private key: $n=4559, d=3377, p=47, q=97$
- compute once: $d_{p}=19, d_{q}=17, q_{i n v}=16$


## Example

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

$$
\begin{gathered}
c_{p}=32 \\
m_{p}=c_{p}^{d_{p}} \bmod p=8 \quad c_{q}=83 \\
m_{q}=c_{q}^{d_{q}} \bmod q=5 \\
h=q_{\mathrm{inv}}\left(m_{p}-m_{q}\right) \bmod p=1 \\
m=m_{q}+h q \bmod n=102
\end{gathered}
$$

Note: These computations are nearly possible by hand.

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

An addition chain for integer $n$ of length $I$ 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 $/$ 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 $/$ 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: $15=1111_{2} \sim 7$ multiplications

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

An addition chain for integer $n$ of length $/$ 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: $15=1111_{2} \sim 7$ multiplications

$$
\begin{array}{ll}
x^{2}=x \cdot x \quad \bmod n & x^{12}=x^{6} \cdot x^{6} \quad \bmod n \\
x^{3}=x^{2} \cdot x \quad \bmod n & x^{15}=x^{12} \cdot x^{3} \quad \bmod n \\
x^{6}=x^{3} \cdot x^{3} \quad \bmod n &
\end{array}
$$

## Problem with Addition Chains

Let $I(n)$ denote length of smallest addition chain for $n$.
Finding $I(n)$ is hard
Let $|n|_{1}$ denote number of 1 s . Known bounds are:

$$
\begin{aligned}
\log n+\log |n|_{1}-2.13 \leq I(n) & \leq \log n+|n|_{1}-1 \\
I(n) & \in \log n+(1+o(1)) \cdot \frac{\log n}{\log \log n}
\end{aligned}
$$

## Problem with Addition Chains

Let $I(n)$ denote length of smallest addition chain for $n$.
Finding $I(n)$ is hard
Let $|n|_{1}$ denote number of 1 s . Known bounds are:

$$
\begin{aligned}
\log n+\log |n|_{1}-2.13 \leq I(n) & \leq \log n+|n|_{1}-1 \\
I(n) & \in \log n+(1+o(1)) \cdot \frac{\log n}{\log \log n}
\end{aligned}
$$

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 $I(n)$ denote length of smallest addition chain for $n$.
Finding $I(n)$ is hard
Let $|n|_{1}$ denote number of 1 s . Known bounds are:

$$
\begin{aligned}
\log n+\log |n|_{1}-2.13 \leq I(n) & \leq \log n+|n|_{1}-1 \\
I(n) & \in \log n+(1+o(1)) \cdot \frac{\log n}{\log \log n}
\end{aligned}
$$

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 (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 <br> If $d$ is known, it is not sufficient to just replace e and $d$.

## 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:

$$
\begin{aligned}
& a:=n-\varphi(n)=p+q-1 \\
& n=p \cdot q=p \cdot(a+1-p)
\end{aligned}
$$

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 $e d-1=x \cdot \varphi(n)$ for all $x \leq 2 e$; likely $x=\left\lceil\frac{e d-1}{n}\right\rceil$

Theorem of Secret Parameters - $d$ known
Assume $d$ is known.
$e$ small: Test ed $-1=x \cdot \varphi(n)$ for all $x \leq 2 e$; likely $x=\left\lceil\frac{e d-1}{n}\right\rceil$ else: Note: $\operatorname{gcd}(*, n) \in\{1, p, q, n\}$

Theorem of Secret Parameters - $d$ known
Assume $d$ is known.
$e$ small: Test ed $-1=x \cdot \varphi(n)$ for all $x \leq 2 e$; likely $x=\left\lceil\frac{e d-1}{n}\right\rceil$ else: Note: $\operatorname{gcd}(*, n) \in\{1, p, q, n\}$

1: function $\operatorname{FACtOR}(d, e, n)$
2: $\quad s \leftarrow \mathrm{M}_{2}(e d-1)$
3: $\quad k \leftarrow \frac{e d-1}{2^{s}}$
$\triangleright$ multiplicity of 2
$\triangleright$ "odd part"

Assume $d$ is known.
$e$ small: Test $e d-1=x \cdot \varphi(n)$ for all $x \leq 2 e$; likely $x=\left\lceil\frac{e d-1}{n}\right\rceil$ else: Note: $\operatorname{gcd}(*, n) \in\{1, p, q, n\}$

1: function $\operatorname{FACTOR}(d, e, n)$

```
2: \(\quad s \leftarrow \mathrm{M}_{2}(e d-1)\)
\(\triangleright\) multiplicity of 2
```

3: $\quad k \leftarrow \frac{e d-1}{2^{s}}$
4: while True do
5: $\quad$ pick random $0<a<n$
6: $\quad$ if $\operatorname{gcd}(a, n)>1$ then return gcd
for $i=0, \ldots, s-1$ do
9:
10 : if $\operatorname{gcd}\left(\left(a^{k}\right)^{2^{i}}-1, n\right) \notin\{1, n\}$ then
return gcd

## Theorem of Secret Parameters $-d$ known

Assume $d$ is known.
$e$ small: Test $e d-1=x \cdot \varphi(n)$ for all $x \leq 2 e$; likely $x=\left\lceil\frac{e d-1}{n}\right\rceil$ else: Note: $\operatorname{gcd}(*, n) \in\{1, p, q, n\}$

1: function $\operatorname{FACTOR}(d, e, n)$

```
2: \(\quad s \leftarrow \mathrm{M}_{2}(e d-1)\)
\(\triangleright\) multiplicity of 2
```

3: $\quad k \leftarrow \frac{e d-1}{2^{s}}$
4: while True do
5: $\quad$ pick random $0<a<n$
6: if $\operatorname{gcd}(a, n)>1$ then $\quad \triangleright$ very low chance
7: return gcd - "odd part"

8: $\quad$ for $i=0, \ldots, s-1$ do
9:
10: if $\operatorname{gcd}\left(\left(a^{k}\right)^{2^{i}}-1, n\right) \notin\{1, n\}$ then return gcd
Chance of success $\geq \frac{1}{2}$ per loop, but the "why" is more complicatded

Notation: ed $-1=x \cdot \varphi(n)=k \cdot 2^{s}, k$ odd Interesting Code part:

$$
\text { for } i=0, \ldots, s-1 \text { do }
$$

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

## return gcd

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

Notation: ed $-1=x \cdot \varphi(n)=k \cdot 2^{s}, k$ odd Interesting Code part:

$$
\text { for } i=0, \ldots, s-1 \text { do }
$$

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

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

$$
\operatorname{gcd}(a, n)=1 \Longrightarrow\left(a^{k}\right)^{2^{s}}=a^{x \cdot \varphi(n)}=\left(a^{\varphi(n)}\right)^{x} \equiv 1 \quad \bmod n
$$

Notation: ed $-1=x \cdot \varphi(n)=k \cdot 2^{s}, k$ odd Interesting Code part:

$$
\text { for } i=0, \ldots, s-1 \text { do }
$$

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

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

$$
\begin{aligned}
\operatorname{gcd}(a, n)=1 & \Longrightarrow\left(a^{k}\right)^{2^{s}}=a^{x \cdot \varphi(n)}=\left(a^{\varphi(n)}\right)^{x} \equiv 1 \bmod n \\
& \Longrightarrow\left(a^{k}\right)^{2^{s}}-1 \equiv 0 \bmod n \\
& \Longrightarrow \operatorname{gcd}\left(\left(a^{k}\right)^{2^{s}}-1, n\right)=n
\end{aligned}
$$

Notation: ed $-1=x \cdot \varphi(n)=k \cdot 2^{s}, k$ odd Interesting Code part:

$$
\text { for } i=0, \ldots, s-1 \text { do }
$$

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

return gcd
Do not really need $d$, but just some multiple of $\varphi(n)$
Proof (beginning).

$$
\begin{aligned}
\operatorname{gcd}(a, n)=1 & \Longrightarrow\left(a^{k}\right)^{2^{s}}=a^{x \cdot \varphi(n)}=\left(a^{\varphi(n)}\right)^{x} \equiv 1 \bmod n \\
& \Longrightarrow\left(a^{k}\right)^{2^{s}}-1 \equiv 0 \bmod n \\
& \Longrightarrow \operatorname{gcd}\left(\left(a^{k}\right)^{2^{s}}-1, n\right)=n
\end{aligned}
$$

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

## Proof Idea.

- started with observation

$$
\left(a^{k}\right)^{2^{s}}-1 \equiv 0 \quad \bmod n
$$

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

$$
\left(a^{k}\right)^{2^{x}}-1 \equiv 0 \bmod p \quad\left(a^{k}\right)^{2^{y}}-1 \equiv 0 \bmod q
$$

- assume $x, y$ differ, wlog $x<y$

$$
\begin{aligned}
\left(a^{k}\right)^{2^{x}}-1 & \equiv 0 \bmod p \quad\left(a^{k}\right)^{2^{x}}-1 \not \equiv 0 \bmod q \\
& \Longrightarrow \operatorname{gcd}\left(\left(a^{k}\right)^{2^{x}}-1, n\right)=p
\end{aligned}
$$

- try out all $x \sim$ success


## Groups

## Definition

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

- o 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 $\mathcal{G}=\left(G, \circ, *^{-1}, 1\right)$ such that

- o 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 o explicitly.

## Groups

## Definition

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

- o 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 o explicitly.

## Example

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


## Groups

## Definition

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

- o 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 o explicitly.

## Example

- $(\mathbb{Z},+),\left(\mathbb{Z}_{n},+\right)$
- $\left(\mathbb{Z}_{n}^{*}, \cdot\right)$ all numbers coprime to $n$
- $S_{n}$ : the group of permutations of $n$ elements
- point addition on elliptic curves ( $\sim$ 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

- $G=\left(\mathbb{Z}_{7}^{*}, \cdot\right), g=2$, 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

- $G=\left(\mathbb{Z}_{7}^{*}, \cdot\right), g=2$, then $2^{3}=8=1$, but $2^{1}, 2^{2} \neq 1$ so $o(2)=3$
- $G=\left(\mathbb{Z}_{7},+\right), g=2$, 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

- $G=\left(\mathbb{Z}_{7}^{*}, \cdot\right), g=2$, then $2^{3}=8=1$, but $2^{1}, 2^{2} \neq 1$ so $o(2)=3$
- $G=\left(\mathbb{Z}_{7},+\right), g=2$, 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, $g \in G$

- $o(g)\left||G|\right.$ (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: $G=\langle g\rangle=\left\{g^{n}: n \in \mathbb{Z}\right\}$.

## Definition

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

## Example

- $\mathbb{Z}=(\mathbb{Z},+): \mathbb{Z}=\langle 1\rangle$
- $\mathbb{G}=\left(\mathbb{Z}_{7},+\right), g=2$ : multiples are $2,4,6,1,3,5,0$, so $\mathbb{Z}_{7}=\langle 2\rangle$
- $\mathbb{G}=\left(\mathbb{Z}_{7}^{*}, \cdot\right): g=3$, 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: $G=\langle g\rangle=\left\{g^{n}: n \in \mathbb{Z}\right\}$.

## Example

- $\mathbb{Z}=(\mathbb{Z},+): \mathbb{Z}=\langle 1\rangle$
- $\mathbb{G}=\left(\mathbb{Z}_{7},+\right), g=2$ : multiples are $2,4,6,1,3,5,0$, so $\mathbb{Z}_{7}=\langle 2\rangle$
- $\mathbb{G}=\left(\mathbb{Z}_{7}^{*}, \cdot\right): g=3$, 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.)

$$
\mathbb{Z}_{n} \cong \mathbb{Z}_{p} \times \mathbb{Z}_{q}
$$

Notation: ed $-1=x \cdot \varphi(n)=k \cdot 2^{s}$
Proof (cont.)

$$
\mathbb{Z}_{n}{ }^{*} \cong \mathbb{Z}_{p}{ }^{*} \times \mathbb{Z}_{q}{ }^{*}
$$

Notation: ed $-1=x \cdot \varphi(n)=k \cdot 2^{s}$
Proof (cont.)

$$
\mathbb{Z}_{n}{ }^{*} \cong \mathbb{Z}_{p}{ }^{*} \times \mathbb{Z}_{q}{ }^{*}
$$

## 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.)

$$
\mathbb{Z}_{n}{ }^{*} \cong \mathbb{Z}_{p}{ }^{*} \times \mathbb{Z}_{q}{ }^{*}
$$

## 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.)

$$
\mathbb{Z}_{n}{ }^{*} \cong \mathbb{Z}_{p}{ }^{*} \times \mathbb{Z}_{q}{ }^{*}
$$

## 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 $\left(a^{k}\right) \mid 2^{s}$

$$
\Longrightarrow o_{\mathbb{Z}_{p}^{*}}\left(a^{k}\right)=2^{l_{1}} \quad o_{\mathbb{Z}_{q}^{*}}\left(a^{k}\right)=2^{l_{2}} \quad \text { for some } I_{1}, l_{2} \leq s
$$

Notation: ed $-1=x \cdot \varphi(n)=k \cdot 2^{s}$
Proof (cont.)

$$
\mathbb{Z}_{n}{ }^{*} \cong \mathbb{Z}_{p}{ }^{*} \times \mathbb{Z}_{q}{ }^{*}
$$

## 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 $o\left(a^{k}\right) \mid 2^{s}$

$$
\Longrightarrow o_{\mathbb{Z}_{p}^{*}}\left(a^{k}\right)=2^{l_{1}} \quad o_{\mathbb{Z}_{q}^{*}}\left(a^{k}\right)=2^{l_{2}} \quad \text { for some } I_{1}, l_{2} \leq s
$$

Let $g, h$ be generators of $\mathbb{Z}_{p}^{*}, \mathbb{Z}_{q}^{*}$ (i.e. $\mathbb{Z}_{p}^{*}=\langle g\rangle=\left\{g^{n}: n \in \mathbb{N}\right\}$ ) random $a \sim a^{k} \cong\left(g^{y}, h^{z}\right)$ (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.)

$$
\mathbb{Z}_{n}{ }^{*} \cong \mathbb{Z}_{p}{ }^{*} \times \mathbb{Z}_{q}{ }^{*}
$$

## 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 $o\left(a^{k}\right) \mid 2^{s}$

$$
\Longrightarrow o_{\mathbb{Z}_{p}^{*}}\left(a^{k}\right)=2^{l_{1}} \quad o_{\mathbb{Z}_{q}^{*}}\left(a^{k}\right)=2^{l_{2}} \quad \text { for some } I_{1}, l_{2} \leq s
$$

Let $g, h$ be generators of $\mathbb{Z}_{p}^{*}, \mathbb{Z}_{q}^{*}$ (i.e. $\mathbb{Z}_{p}^{*}=\langle g\rangle=\left\{g^{n}: n \in \mathbb{N}\right\}$ ) random $a \sim a^{k} \cong\left(g^{y}, h^{z}\right)$ (a bit random) in $\mathbb{Z}_{p}^{*} \times \mathbb{Z}_{q}^{*}$ for some $y, z$ $o\left(g^{y}\right)=2^{/_{1}}, o\left(h^{z}\right)=2^{/_{2}}$ are 2-powers; if different $\sim$ success

Notation: ed $-1=x \cdot \varphi(n)=k \cdot 2^{s}$
Proof (cont.)

$$
\begin{equation*}
\mathbb{Z}_{n}{ }^{*} \cong \mathbb{Z}_{p}{ }^{*} \times \mathbb{Z}_{q}{ }^{*} \tag{CRT}
\end{equation*}
$$

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 $o\left(a^{k}\right) \mid 2^{s}$

$$
\Longrightarrow o_{\mathbb{Z}_{p}^{*}}\left(a^{k}\right)=2^{l_{1}} \quad o_{\mathbb{Z}_{q}^{*}}\left(a^{k}\right)=2^{l_{2}} \quad \text { for some } I_{1}, l_{2} \leq s
$$

Let $g, h$ be generators of $\mathbb{Z}_{p}^{*}, \mathbb{Z}_{q}^{*}$ (i.e. $\mathbb{Z}_{p}^{*}=\langle g\rangle=\left\{g^{n}: n \in \mathbb{N}\right\}$ ) random $a \sim a^{k} \cong\left(g^{y}, h^{z}\right)$ (a bit random) in $\mathbb{Z}_{p}^{*} \times \mathbb{Z}_{q}^{*}$ for some $y, z$ $o\left(g^{y}\right)=2^{1_{1}}, o\left(h^{z}\right)=2^{/_{2}}$ are 2-powers; if different $\sim$ success Assume wlog $l_{1}<l_{2}$, then

$$
\left(a^{k}\right)^{2_{1}} \equiv 1 \bmod p \quad\left(a^{k}\right)^{2_{1}} \not \equiv 1 \quad \bmod q
$$

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

## Proof (cont.)

$$
\begin{equation*}
\mathbb{Z}_{n}{ }^{*} \cong \mathbb{Z}_{p}{ }^{*} \times \mathbb{Z}_{q}{ }^{*} \tag{CRT}
\end{equation*}
$$

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 $o\left(a^{k}\right) \mid 2^{s}$

$$
\Longrightarrow o_{\mathbb{Z}_{p}^{*}}\left(a^{k}\right)=2^{l_{1}} \quad o_{\mathbb{Z}_{q}^{*}}\left(a^{k}\right)=2^{l_{2}} \quad \text { for some } I_{1}, l_{2} \leq s
$$

Let $g, h$ be generators of $\mathbb{Z}_{p}^{*}, \mathbb{Z}_{q}^{*}$ (ie. $\mathbb{Z}_{p}^{*}=\langle g\rangle=\left\{g^{n}: n \in \mathbb{N}\right\}$ ) random $a \leadsto a^{k} \cong\left(g^{y}, h^{z}\right)$ (a bit random) in $\mathbb{Z}_{p}^{*} \times \mathbb{Z}_{q}^{*}$ for some $y, z$ $o\left(g^{y}\right)=2^{1_{1}}, o\left(h^{z}\right)=2^{l_{2}}$ are 2-powers; if different $\leadsto$ success Assume wog $I_{1}<l_{2}$, then

$$
\begin{gathered}
\left(a^{k}\right)^{2_{1}} \equiv 1 \quad \bmod p \quad\left(a^{k}\right)^{2^{1}} \not \equiv 1 \quad \bmod q \\
p \mid\left(a^{k}\right)^{2^{1 /}}-1 \quad q \nmid\left(a^{k}\right)^{2^{1}}-1
\end{gathered} \Longrightarrow \operatorname{gcd}\left(\left(a^{k}\right)^{2^{1 /}}-1, n\right)=p
$$

## Proof (cont.)

Recall $a^{k} \cong\left(g^{y}, h^{z}\right) \in \mathbb{Z}_{p}^{*} \times \mathbb{Z}_{q}^{*}$
$\mathrm{M}_{2}\left(\left|\mathbb{Z}_{p}^{*}\right|\right)$ : How much squaring is irreversible in $\mathbb{Z}_{p}^{*}$ ?
$\mathrm{M}_{2}(y)$ : How much squaring did we already do?
$I_{1}$ : How much squaring do we still have to do?

## Proof (cont.)

Recall $a^{k} \cong\left(g^{y}, h^{z}\right) \in \mathbb{Z}_{p}^{*} \times \mathbb{Z}_{q}^{*}$
$\mathrm{M}_{2}\left(\left|\mathbb{Z}_{p}^{*}\right|\right)$ : How much squaring is irreversible in $\mathbb{Z}_{p}^{*}$ ?
$\mathrm{M}_{2}(y)$ : How much squaring did we already do?
$I_{1}$ : How much squaring do we still have to do?

$$
\mathrm{M}_{2}\left(\left|\mathbb{Z}_{p}^{*}\right|\right)=\mathrm{M}_{2}(p-1)=\mathrm{M}_{2}(y)+\iota_{1}
$$

## Proof (cont.)

Recall $a^{k} \cong\left(g^{y}, h^{z}\right) \in \mathbb{Z}_{p}^{*} \times \mathbb{Z}_{q}^{*}$
$\mathrm{M}_{2}\left(\left|\mathbb{Z}_{p}^{*}\right|\right)$ : How much squaring is irreversible in $\mathbb{Z}_{p}^{*}$ ?
$\mathrm{M}_{2}(y)$ : How much squaring did we already do?
$I_{1}$ : How much squaring do we still have to do?

$$
\mathrm{M}_{2}\left(\left|\mathbb{Z}_{p}^{*}\right|\right)=\mathrm{M}_{2}(p-1)=\mathrm{M}_{2}(y)+\iota_{1}
$$

Case distinction:
$\mathrm{M}_{2}(p-1)<\mathrm{M}_{2}(q-1)$ : if $z$ odd, then $\mathrm{M}_{2}(z)=0$, so

$$
I_{1} \leq \mathrm{M}_{2}(p-1)<\mathrm{M}_{2}(q-1)=I_{2}
$$

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

## Proof (cont.)

Recall $a^{k} \cong\left(g^{y}, h^{z}\right) \in \mathbb{Z}_{p}^{*} \times \mathbb{Z}_{q}^{*}$
$\mathrm{M}_{2}\left(\left|\mathbb{Z}_{p}^{*}\right|\right)$ : How much squaring is irreversible in $\mathbb{Z}_{p}^{*}$ ?
$\mathrm{M}_{2}(y)$ : How much squaring did we already do?
$I_{1}$ : How much squaring do we still have to do?

$$
\mathrm{M}_{2}\left(\left|\mathbb{Z}_{p}^{*}\right|\right)=\mathrm{M}_{2}(p-1)=\mathrm{M}_{2}(y)+\iota_{1}
$$

Case distinction:
$\mathrm{M}_{2}(p-1)<\mathrm{M}_{2}(q-1)$ : if $z$ odd, then $\mathrm{M}_{2}(z)=0$, so

$$
I_{1} \leq \mathrm{M}_{2}(p-1)<\mathrm{M}_{2}(q-1)=I_{2}
$$

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 $I_{1} \neq I_{2}$; 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

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
- 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
RSA-Problem $\leq_{p}$ finding $d \leq_{p}$ 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:

$$
\text { find } d \equiv B P P \text { FACTORING }
$$

i.e. equivalent in "bounded 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:

$$
\text { find } d \equiv_{B P P} \text { FACTORING }
$$

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

Theorem (Coron, May, 2004)
Using Coppersmith (see later): finding $d \equiv{ }_{p}$ 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:

$$
\text { find } d \equiv_{B P P} \text { FACTORING }
$$

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

Theorem (Coron, May, 2004)
Using Coppersmith (see later): finding $d \equiv{ }_{p}$ Factoring
For the first reduction, the converse is open.
finding $d \stackrel{?}{\leq} p$ 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 <br> 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 $m^{e}<n \Longrightarrow m^{e} \bmod n=m^{e}$
- 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 $m^{e}<n \Longrightarrow m^{e} \bmod n=m^{e}$
- 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 $m=102$

$$
m^{3} \bmod n=1061208 \bmod 10720441=1061208
$$

## Padding

## Padding

Artificially enlarge message to $m^{\prime}=\operatorname{Pad}(m)$, such that $m^{\prime} \approx n$.

## Padding

## Padding

Artificially enlarge message to $m^{\prime}=\operatorname{Pad}(m)$, such that $m^{\prime} \approx n$.
Toy Example 1: multiply with fixed, known number
Encrypt: $m^{\prime}:=m \cdot r$ for fixed $r \in \mathbb{Z}_{n}^{*}$; $c=\left(m^{\prime}\right)^{e} \bmod n$ Decrypt: $m=r^{-1} \cdot\left(m^{\prime}\right)^{d} \bmod n$

## Padding

## Padding

Artificially enlarge message to $m^{\prime}=\operatorname{Pad}(m)$, such that $m^{\prime} \approx n$.
Toy Example 1: multiply with fixed, known number
Encrypt: $m^{\prime}:=m \cdot r$ for fixed $r \in \mathbb{Z}_{n}^{*} ; c=\left(m^{\prime}\right)^{e} \bmod n$ Decrypt: $m=r^{-1} \cdot\left(m^{\prime}\right)^{d} \bmod n$

Break
RSA is multiplicative
If we know enc $(m)$, then we also know enc $(x \cdot m)$ for every $x$.

## Padding

## Padding

Artificially enlarge message to $m^{\prime}=\operatorname{Pad}(m)$, such that $m^{\prime} \approx n$.
Toy Example 1: multiply with fixed, known number
Encrypt: $m^{\prime}:=m \cdot r$ for fixed $r \in \mathbb{Z}_{n}^{*} ; c=\left(m^{\prime}\right)^{e} \bmod n$
Decrypt: $m=r^{-1} \cdot\left(m^{\prime}\right)^{d} \bmod n$
Break

## RSA is multiplicative

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

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

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

## Padding

Toy Example 2: Fill with 0s
Putting $m^{\prime}=m \| 0 \ldots 0$

## Padding

Toy Example 2: Fill with 0s
Putting $m^{\prime}=m \| 0 \ldots 0$ is just $m^{\prime}=m \cdot 2^{k}$, so same as version 1 .

## Padding

Toy Example 2: Fill with 0s
Putting $m^{\prime}=m \| 0 \ldots 0$ is just $m^{\prime}=m \cdot 2^{k}$, so same as version 1 .
Toy Example 3: Concatenate concatenate $m$ with itself: $m^{\prime}=m\|\ldots\| m$

## Padding

Toy Example 2: Fill with 0s
Putting $m^{\prime}=m \| 0 \ldots 0$ is just $m^{\prime}=m \cdot 2^{k}$, so same as version 1 .
Toy Example 3: Concatenate concatenate $m$ with itself: $m^{\prime}=m\|\ldots\| m$
But mathematically, that is just

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

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


## Coppersmith

Theorem (Coppersmith, 1996)
Let $f \in \mathbb{Z}[x]$ normalised, $e=\operatorname{deg} f$. Then we can compute all $x_{0} \in \mathbb{Z}$ with $f\left(x_{0}\right) \equiv 0 \bmod n$ and $\left|x_{0}\right| \leq \sqrt[e]{n}$ in polynomial time.

## Coppersmith

Theorem (Coppersmith, 1996)
Let $f \in \mathbb{Z}[x]$ normalised, $e=\operatorname{deg} f$. Then we can compute all $x_{0} \in \mathbb{Z}$ with $f\left(x_{0}\right) \equiv 0 \bmod n$ and $\left|x_{0}\right| \leq \sqrt[e]{n}$ in polynomial time.
polynomial time $\neq$ efficient

## Coppersmith

Theorem (Coppersmith, 1996)
Let $f \in \mathbb{Z}[x]$ normalised, $e=\operatorname{deg} f$. Then we can compute all $x_{0} \in \mathbb{Z}$ with $f\left(x_{0}\right) \equiv 0 \bmod n$ and $\left|x_{0}\right| \leq \sqrt[e]{n}$ in polynomial time.
polynomial time $\neq$ efficient
Theorem (from Nina Jekel, Bsc-thesis, 2017)
Let $f \in \mathbb{Z}[x]$ normalised, $\operatorname{deg} f=e$. 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 $m^{d} \bmod n$ ):

$$
m^{\prime}=00\|01\| F F \ldots F F\|00\| m
$$

## 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 $m^{d} \bmod n$ ):

$$
m^{\prime}=00\|01\| F F \ldots F F\|00\| m
$$

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


## 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 $m^{d} \bmod n$ ):

$$
m^{\prime}=00\|01\| F F \ldots F F\|00\| m
$$

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

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

Counting and adding correct 2-power:

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

## 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 $m^{d} \bmod n$ ):

$$
m^{\prime}=00\|01\| F F \ldots F F\|00\| m
$$

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

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

Counting and adding correct 2-power:

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

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

## Coppersmith Failure

Example (NSUCrypto 2019, Problem 3)
We know $p, q$ have 500 bits. Given $n=p q$ and

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

Find $p, q$.

## Coppersmith Failure

Example (NSUCrypto 2019, Problem 3)
We know $p, q$ have 500 bits. Given $n=p q$ and

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

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

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

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 $n=p q$ and

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

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

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

If we assume $p<q$, then $p^{2}<n$, so $p^{4}<n^{2}<N$.
But $\varepsilon$ too small $\sim$ 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 $\operatorname{gcd}\left(n_{i}, n_{j}\right) \neq 1$, we found a prime factor. $\checkmark$
So wlog system of congruences with coprime $n_{i}$

$$
c_{i} \equiv m^{e} \quad \bmod n_{i} \quad i=1, \ldots, e
$$

## 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 $\operatorname{gcd}\left(n_{i}, n_{j}\right) \neq 1$, we found a prime factor. $\checkmark$
So wlog system of congruences with coprime $n_{i}$

$$
c_{i} \equiv m^{e} \quad \bmod n_{i} \quad i=1, \ldots, e
$$

Put $x=m^{e}$ and solve via CRT. Unique solution $m^{e} \bmod \prod n_{i}$

$$
m<n_{i} \Longrightarrow m^{e}<\prod n_{i} \Longrightarrow m^{e} \bmod \prod n_{i}=m^{e}
$$

## 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 $\operatorname{gcd}\left(n_{i}, n_{j}\right) \neq 1$, we found a prime factor. $\checkmark$
So wlog system of congruences with coprime $n_{i}$

$$
c_{i} \equiv m^{e} \quad \bmod n_{i} \quad i=1, \ldots, e
$$

Put $x=m^{e}$ and solve via CRT. Unique solution $m^{e} \bmod \prod n_{i}$

$$
m<n_{i} \Longrightarrow m^{e}<\prod n_{i} \Longrightarrow m^{e} \bmod \prod n_{i}=m^{e}
$$

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

## Example

Same message $m, e=3, c_{i}=m^{e} \bmod n_{i}$ :

$$
\begin{array}{ll}
n_{1}=551 & c_{1}=533 \\
n_{2}=943 & c_{2}=333 \\
n_{3}=527 & c_{3}=357
\end{array}
$$

## Example

Same message $m, e=3, c_{i}=m^{e} \bmod n_{i}$ :

$$
\begin{array}{ll}
n_{1}=551 & c_{1}=533 \\
n_{2}=943 & c_{2}=333 \\
n_{3}=527 & c_{3}=357
\end{array}
$$

CRT yields

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

## 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

$$
k \geq e \cdot \max \left\{\operatorname{deg} f_{i}: i=1, \ldots, k\right\}
$$

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

$$
k \geq e \cdot \max \left\{\operatorname{deg} f_{i}: i=1, \ldots, k\right\}
$$

then we can recover $m$ from the $f_{i}$ and $c_{i}=m_{i}^{e} \bmod n_{i}$.

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


## 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

$$
k \geq e \cdot \max \left\{\operatorname{deg} f_{i}: i=1, \ldots, k\right\}
$$

then we can recover $m$ from the $f_{i}$ and $c_{i}=m_{i}^{e} \bmod n_{i}$.

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


## 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 $g_{i}(m) \equiv 0 \bmod n_{i}$, Note: $\operatorname{deg} g_{i}=e \cdot \operatorname{deg} f_{i} \leq k$


## Proof.

- put $g_{i}(x)=f_{i}(x)^{e}-c_{i}$, so all $g_{i}(m) \equiv 0 \bmod n_{i}$, Note: $\operatorname{deg} g_{i}=e \cdot \operatorname{deg} f_{i} \leq k$
- with CRT compute $T_{i}$ with $T_{i} \equiv 1 \bmod n_{i}$ and $T_{i} \equiv 0 \bmod n_{j}$ for $i \neq j$ and put

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

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

## Proof.

- put $g_{i}(x)=f_{i}(x)^{e}-c_{i}$, so all $g_{i}(m) \equiv 0 \bmod n_{i}$, Note: $\operatorname{deg} g_{i}=e \cdot \operatorname{deg} f_{i} \leq k$
- with CRT compute $T_{i}$ with $T_{i} \equiv 1 \bmod n_{i}$ and $T_{i} \equiv 0 \bmod n_{j}$ for $i \neq j$ and put

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

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

- $g(m) \equiv 0 \bmod n_{i}$ 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 $g_{i}(m) \equiv 0 \bmod n_{i}$, Note: $\operatorname{deg} g_{i}=e \cdot \operatorname{deg} f_{i} \leq k$
- with CRT compute $T_{i}$ with $T_{i} \equiv 1 \bmod n_{i}$ and $T_{i} \equiv 0 \bmod n_{j}$ for $i \neq j$ and put

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

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

- $g(m) \equiv 0 \bmod n_{i}$ for all $i$ :
- summands $j \neq i$ vanish because of $T_{j}$
- summand $i$ because of definition of $g_{i}$
- by CRT $g(m) \equiv 0 \bmod \prod n_{i}$

$$
m<\min _{i} n_{i}<\left(\prod n_{i}\right)^{\frac{1}{k}} \leq\left(\prod n_{i}\right)^{\frac{1}{\operatorname{deg} g}}
$$

- 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 $a_{k}$ ).
$\sim$ 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 $a_{k}$ ).
$\sim$ 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 $a_{k}$ ).
$\sim$ 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 $\sim$ What holds over $\mathbb{Z}_{n}$ ?

## Polynomial Division

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

$$
g=\sum_{k=0}^{D_{1}} g_{k} x^{k}
$$

$$
h=\sum_{k=0}^{D_{2}} h_{k} x^{k}
$$

with $D_{1} \geq D_{2}$.

## Polynomial Division

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

$$
g=\sum_{k=0}^{D_{1}} g_{k} x^{k}
$$

$$
h=\sum_{k=0}^{D_{2}} h_{k} x^{k}
$$

with $D_{1} \geq D_{2}$.
Then first step of division is

$$
g=\left(g_{D_{1}} \cdot h_{D_{2}}^{-1} \cdot x^{D_{1}-D_{2}}\right) \cdot h+\operatorname{Rem}_{1}
$$

where $\operatorname{deg}\left(\operatorname{Rem}_{1}\right)<\operatorname{deg} g$.

## Polynomial Division

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

$$
g=\sum_{k=0}^{D_{1}} g_{k} x^{k}
$$

$$
h=\sum_{k=0}^{D_{2}} h_{k} x^{k}
$$

with $D_{1} \geq D_{2}$.
Then first step of division is

$$
g=\left(g_{D_{1}} \cdot h_{D_{2}}^{-1} \cdot x^{D_{1}-D_{2}}\right) \cdot h+\operatorname{Rem}_{1}
$$

where $\operatorname{deg}\left(\operatorname{Rem}_{1}\right)<\operatorname{deg} g$. Continue with $\operatorname{Rem}_{1}$ and $h$.

## Polynomial Division

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

$$
g=\sum_{k=0}^{D_{1}} g_{k} x^{k}
$$

$$
h=\sum_{k=0}^{D_{2}} h_{k} x^{k}
$$

with $D_{1} \geq D_{2}$.
Then first step of division is

$$
g=\left(g_{D_{1}} \cdot h_{D_{2}}^{-1} \cdot x^{D_{1}-D_{2}}\right) \cdot h+\operatorname{Rem}_{1}
$$

where $\operatorname{deg}\left(\operatorname{Rem}_{1}\right)<\operatorname{deg} g$. Continue with $\operatorname{Rem}_{1}$ 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 $f\left(x_{0}\right)=0$. Then there is $g \in \mathbb{Z}_{n}[x]$ with $f=\left(x-x_{0}\right) g$, which we can compute via polynomial division.

## Franklin-Reiter-Related-Message-Attack

## Theorem

If two messages are related via $m_{2}=f\left(m_{1}\right)$ for some known polynomial $f$, we often can recover them from $c_{i}=m_{i}^{e} \bmod n$. The time is $\mathcal{O}\left((e \cdot \operatorname{deg} f)^{2}\right)$ 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 $m_{2}=f\left(m_{1}\right)$ for some known polynomial $f$, we often can recover them from $c_{i}=m_{i}^{e} \bmod n$. The time is $\mathcal{O}\left((e \cdot \operatorname{deg} f)^{2}\right)$ arithmetic operations. If $f$ is linear and $e=3$ the attack is guaranteed to work.

## Proof.

Define polynomials

$$
\begin{aligned}
& g(x)=x^{e}-c_{1} \quad h(x)=f(x)^{e}-c_{2} \\
\Longrightarrow & g\left(m_{1}\right)=h\left(m_{1}\right)=0 \\
\Longrightarrow & \left(x-m_{1}\right) \mid \operatorname{gcd}(g, h)
\end{aligned}
$$

Mostly gcd is linear, if $e=3$ and $\operatorname{deg} f=1$, 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)

$$
g(x)=x^{3}-c_{1} \quad h(x)=\left(x+2^{296}\right)^{3}-c_{2}
$$

## 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)

$$
g(x)=x^{3}-c_{1} \quad h(x)=\left(x+2^{296}\right)^{3}-c_{2}
$$

Calling Euclidean Algo:

$$
\begin{array}{lr}
r_{1}=h-g=3 \cdot 2^{592} x^{2}+3 \cdot 2^{296} x+c_{1}-c_{2} & \text { cancel } x^{3} \\
r_{2}=g-\left(3 \cdot 2^{592}\right)^{-1} x r_{1}-* \cdot r_{1}=k\left(x-m_{1}\right) & \text { cancel } x^{3}, x^{2}
\end{array}
$$

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)

$$
g(x)=x^{3}-c_{1} \quad h(x)=\left(x+2^{296}\right)^{3}-c_{2}
$$

Calling Euclidean Algo:

$$
\begin{array}{lr}
r_{1}=h-g=3 \cdot 2^{592} x^{2}+3 \cdot 2^{296} x+c_{1}-c_{2} & \text { cancel } x^{3} \\
r_{2}=g-\left(3 \cdot 2^{592}\right)^{-1} x r_{1}-* \cdot r_{1}=k\left(x-m_{1}\right) & \text { cancel } x^{3}, x^{2}
\end{array}
$$

for some $k \in \mathbb{Z}_{n}$.
If inverting $3 \cdot 2^{592}$ or $k$ fails, then $\operatorname{gcd}(*, n) \in\{p, q\}$.
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

$$
g(x, y)=x^{e}-c_{1} \quad h(x, y)=(x+y)^{e}-c_{2}
$$

- See $y$ as parameter and $x$ as actual variable.
- If $y=r_{2}-r_{1}$, then common root $g\left(2^{R} m+r_{1}, y\right)=h\left(2^{R} m+r_{1}, y\right)=0$.

Starting to break it
Define polynomials

$$
g(x, y)=x^{e}-c_{1} \quad h(x, y)=(x+y)^{e}-c_{2}
$$

- See $y$ as parameter and $x$ as actual variable.
- If $y=r_{2}-r_{1}$, then common root $g\left(2^{R} m+r_{1}, y\right)=h\left(2^{R} m+r_{1}, y\right)=0$.

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

## Lemma

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

## Resultants

For $e=3$ we have

$$
\operatorname{res}(g, h)=\operatorname{det}\left(\begin{array}{cccccc}
1 & 0 & 0 & -c_{1} & 0 & 0 \\
0 & 1 & 0 & 0 & -c_{1} & 0 \\
0 & 0 & 1 & 0 & 0 & -c_{1} \\
1 & 3 y & 3 y^{2} & y^{3}-c_{2} & 0 & 0 \\
0 & 1 & 3 y & 3 y^{2} & y^{3}-c_{2} & 0 \\
0 & 0 & 1 & 3 y & 3 y^{2} & y^{3}-c_{2}
\end{array}\right)
$$

## Resultants

For $e=3$ we have

$$
\operatorname{res}(g, h)=\operatorname{det}\left(\begin{array}{cccccc}
1 & 0 & 0 & -c_{1} & 0 & 0 \\
0 & 1 & 0 & 0 & -c_{1} & 0 \\
0 & 0 & 1 & 0 & 0 & -c_{1} \\
1 & 3 y & 3 y^{2} & y^{3}-c_{2} & 0 & 0 \\
0 & 1 & 3 y & 3 y^{2} & y^{3}-c_{2} & 0 \\
0 & 0 & 1 & 3 y & 3 y^{2} & y^{3}-c_{2}
\end{array}\right)
$$

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


## Coppersmith + Franklin-Reiter

- $\operatorname{res}(g, h) \in \mathbb{Z}_{n}[y]$ 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

- $\operatorname{res}(g, h) \in \mathbb{Z}_{n}[y]$ 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: $m_{2}=m_{1}+y=: f\left(m_{1}\right)$
- (Try to) Recover via

$$
x-m_{1}=\operatorname{gcd}\left(x^{e}-c_{1},(x+y)^{e}-c_{2}\right)
$$

- original message $m=m_{1} / / 2^{R}=\left(m_{1} \gg R\right)$


## Coppersmith + Franklin-Reiter

- $\operatorname{res}(g, h) \in \mathbb{Z}_{n}[y]$ 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: $m_{2}=m_{1}+y=: f\left(m_{1}\right)$
- (Try to) Recover via

$$
x-m_{1}=\operatorname{gcd}\left(x^{e}-c_{1},(x+y)^{e}-c_{2}\right)
$$

- original message $m=m_{1} / / 2^{R}=\left(m_{1} \gg R\right)$

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:

$$
c=m^{e} \quad \bmod n \Longrightarrow c m_{1}^{-e} \equiv m_{2}^{e} \quad \bmod n
$$

## 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:

$$
c=m^{e} \quad \bmod n \Longrightarrow c m_{1}^{-e} \equiv m_{2}^{e} \quad \bmod n
$$

Strategy:

- List $c m_{1}^{-e} \bmod n$ for all $m_{1} \leq 2^{b_{1}} \sim$ (parallel write, not trivial)
- Look up $m_{2}^{e} \bmod n$ for all $m_{2} \leq 2^{b_{2}} \sim$ (parallel, only read)
- Search for collision


## Meet in the Middle



## Meet in the Middle



Meet in the Middle


## Very Small Message $m$

- List $c m_{1}^{-e} \bmod n$ for all $m_{1} \leq 2^{b_{1}} \sim$ store
- Look up $m_{2}^{e} \bmod n$ for all $m_{2} \leq 2^{b_{2}} \sim$ don't store
- Search for collision should be $\mathcal{O}(1)$


## Very Small Message $m$

- List $c m_{1}^{-e} \bmod n$ for all $m_{1} \leq 2^{b_{1}} \sim$ store
- Look up $m_{2}^{e} \bmod n$ for all $m_{2} \leq 2^{b_{2}} \sim$ don't store
- Search for collision should be $\mathcal{O}(1)$

Analysis (assume $b_{1}<b_{2}$ ):
Time: $\mathcal{O}\left(2^{b_{1}}+2^{b_{2}}\right)=\mathcal{O}\left(2^{b_{2}}\right)$ exp. and $\mathcal{O}\left(2^{b_{1}}\right)$ inv.
Memory: $\mathcal{O}\left(2^{b_{1}} \log n\right)$
Total: if all goes well $b_{1}=b_{2}$, so $2^{b_{1}} \approx \sqrt{m}$
$\mathcal{O}(\sqrt{m} \cdot$ poly $)$ time and space
Compare: Brute Force $\mathcal{O}(m \cdot$ poly $)$ time, but $\mathcal{O}(\log n)$ space
$\sim$ Space-Time-Tradeoff

## Very Small Message $m$

- List $c m_{1}^{-e} \bmod n$ for all $m_{1} \leq 2^{b_{1}} \sim$ store
- Look up $m_{2}^{e} \bmod n$ for all $m_{2} \leq 2^{b_{2}} \sim$ don't store
- Search for collision should be $\mathcal{O}(1)$

Analysis (assume $b_{1}<b_{2}$ ):
Time: $\mathcal{O}\left(2^{b_{1}}+2^{b_{2}}\right)=\mathcal{O}\left(2^{b_{2}}\right)$ exp. and $\mathcal{O}\left(2^{b_{1}}\right)$ inv.
Memory: $\mathcal{O}\left(2^{b_{1}} \log n\right)$
Total: if all goes well $b_{1}=b_{2}$, so $2^{b_{1}} \approx \sqrt{m}$
$\mathcal{O}(\sqrt{m} \cdot$ poly $)$ time and space
Compare: Brute Force $\mathcal{O}(m \cdot$ poly $)$ time, but $\mathcal{O}(\log n)$ space $\sim$ 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 $\left(n, e_{1}\right)$ and $\left(n, e_{2}\right)$ with $\operatorname{gcd}\left(e_{1}, e_{2}\right)=1$, 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 $\left(n, e_{1}\right)$ and ( $n, e_{2}$ ) with $\operatorname{gcd}\left(e_{1}, e_{2}\right)=1$, 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 $\left(n, e_{1}\right)$ and ( $n, e_{2}$ ) with $\operatorname{gcd}\left(e_{1}, e_{2}\right)=1$, 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: $s e_{1}+t e_{2}=1$, 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 $\left(n, e_{1}\right)$ and $\left(n, e_{2}\right)$ with $\operatorname{gcd}\left(e_{1}, e_{2}\right)=1$, 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: $s e_{1}+t e_{2}=1$, with $s<0$ and $t>0$
- compute $c_{1}^{-1} \bmod n$ (if it fails, we found a factor of $n$ )
- Compute $m$ via

$$
\left(c_{1}^{-1}\right)^{|s|} c_{2}^{t} \equiv\left(\left(m^{e_{1}}\right)^{-1}\right)^{|s|}\left(m^{e_{2}}\right)^{t} \equiv m^{s e_{1}+t e_{2}} \equiv m \quad \bmod n
$$

Recap - What can go wrong?
In general, RSA is secure but take care if:
$\checkmark e$ is small: several attacks, find $m$
$\checkmark m$ is very small: find $m$
$\checkmark$ same $n$, but different $e$ : find $m$

Recap - What can go wrong?
In general, RSA is secure but take care if:
$\checkmark e$ is small: several attacks, find $m$
$\checkmark m$ is very small: find $m$
$\checkmark$ 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:
$\checkmark e$ is small: several attacks, find $m$
$\checkmark m$ is very small: find $m$
$\checkmark$ 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:
$\checkmark e$ is small: several attacks, find $m$
$\checkmark m$ is very small: find $m$
$\checkmark$ 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:
$\checkmark e$ is small: several attacks, find $m$
$\checkmark m$ is very small: find $m$
$\checkmark$ 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:
$\checkmark e$ is small: several attacks, find $m$
$\checkmark m$ is very small: find $m$
$\checkmark$ 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 $q<p<2 q, e<\varphi(n)$ 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 $q<p<2 q, e<\varphi(n)$ 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 $q<p<2 q, e<\varphi(n)$ 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 $e=d^{-1} \bmod \varphi(n)$. $\sim$ already unsafe

## Consequences

- Decrypting "always" takes rather long


## Small Private Exponent $d$

Theorem (Wiener, 1989)
Assume $q<p<2 q, e<\varphi(n)$ 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 $e=d^{-1} \bmod \varphi(n)$. $\sim$ 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 $(67,24)$ )

$$
\begin{array}{ll}
67=2 \cdot 24+19 & 5=1 \cdot 4+1 \\
24=1 \cdot 19+5 & 4=4 \cdot 1+0 \\
19=3 \cdot 5+4 &
\end{array}
$$

## Continued Fractions

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

## Example (Euclidean Algo gcd $(67,24)$ )

$$
\begin{array}{ll}
67=2 \cdot 24+19 & 5=1 \cdot 4+1 \\
24=1 \cdot 19+5 & 4=4 \cdot 1+0 \\
19=3 \cdot 5+4 &
\end{array}
$$

yields representation

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

divide - swap - repeat

## Continued Fractions

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

## Example (Euclidean Algo gcd $(67,24)$ )

$$
\begin{array}{ll}
67=2 \cdot 24+19 & 5=1 \cdot 4+1 \\
24=1 \cdot 19+5 & 4=4 \cdot 1+0 \\
19=3 \cdot 5+4 &
\end{array}
$$

yields representation

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

divide - swap - repeat

## Example (cont.)

What if we stop at some intermediate step?

## Example (cont.)

What if we stop at some intermediate step?

$$
\left.[2 ; 1,3,1,4]=2+\frac{1}{1+\frac{1}{3+\frac{1}{1+\frac{1}{4}}}}=\frac{67}{24} \right\rvert\, \Delta=0
$$

## Example (cont.)

What if we stop at some intermediate step?

$$
\begin{aligned}
& {[2 ; 1,3,1,4]=2+\frac{1}{1+\frac{1}{3+\frac{1}{1+\frac{1}{4}}}}=\frac{67}{24}} \\
& {[2 ; 1,3,1]=2+\frac{1}{1+\frac{1}{3+\frac{1}{1}}}=\frac{14}{5}}
\end{aligned} \Delta=0 \quad \frac{1}{120}
$$

## Example (cont.)

What if we stop at some intermediate step?

$$
\begin{aligned}
& {[2 ; 1,3,1,4]=2+\frac{1}{1+\frac{1}{3+\frac{1}{1+\frac{1}{4}}}}=\frac{67}{24}} \\
& {[2 ; 1,3,1]=2+\frac{14}{1+\frac{1}{3+\frac{1}{1}}}=\frac{14}{5}} \\
& \Delta=2+\frac{1}{1+\frac{1}{3}}=\frac{11}{4} \\
& {[2 ; 1,3]}
\end{aligned}=2=-\frac{1}{24}
$$

## Example (cont.)

What if we stop at some intermediate step?

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

## Example (cont.)

What if we stop at some intermediate step?

$$
\begin{array}{llll}
{[2 ; 1,3,1,4]=2+\frac{1}{1+\frac{1}{3+\frac{1}{1+\frac{1}{4}}}}=\frac{67}{24}} & \Delta= & 0  \tag{0}\\
{[2 ; 1,3,1]=2+\frac{1}{1+\frac{1}{3+\frac{1}{1}}}} & =\frac{14}{5} & \Delta= & \frac{1}{120} \\
{[2 ; 1,3]} & =2+\frac{1}{1+\frac{1}{3}} & =\frac{11}{4} & \Delta= \\
{[2 ; 1]} & =2+\frac{1}{1} & =3 & \Delta=\frac{1}{24} \\
{[2]} & & =2 & \Delta= \\
{\left[\begin{array}{l}
24 \\
24
\end{array}\right.}
\end{array}
$$

## Example (cont.)

What if we stop at some intermediate step?

$$
\begin{align*}
& {[2 ; 1,3,1,4]=2+\frac{1}{1+\frac{1}{3+\frac{1}{1+\frac{1}{4}}}}=\frac{67}{24}}  \tag{0}\\
& {[2 ; 1,3,1]=2+\frac{1}{1+\frac{1}{3+\frac{1}{T}}}=\frac{14}{5}} \\
& {[2 ; 1,3]=2+\frac{1}{1+\frac{1}{3}}=\frac{11}{4}} \\
& {[2 ; 1] \quad=2+\frac{1}{1}=3} \\
& \text { [2] } \\
& =2 \\
& \Delta= \\
& \Delta=\frac{1}{120} \\
& \Delta=-\frac{1}{24} \\
& \Delta=\frac{5}{24} \\
& \Delta=-\frac{19}{24}
\end{align*}
$$

## Observations

- difference alternates sign
- absolute value of difference decreases $\uparrow$
- enumerator and denominator increase $\uparrow$


## 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: $\left[z_{0} ; z_{1}, \ldots, z_{n}\right]=\frac{p_{n}}{q_{n}}$

$$
\begin{array}{rlrl}
p_{k} & =z_{k} p_{k-1}+p_{k-2} & p_{-1}=1 & p_{-2}=0 \\
q_{k}=z_{k} q_{k-1}+q_{k-2} & q_{-1}=0 & q_{-2}=1
\end{array}
$$

## 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: $\left[z_{0} ; z_{1}, \ldots, z_{n}\right]=\frac{p_{n}}{q_{n}}$

$$
\begin{array}{lll}
p_{k}=z_{k} p_{k-1}+p_{k-2} & p_{-1}=1 & p_{-2}=0 \\
q_{k}=z_{k} q_{k-1}+q_{k-2} & q_{-1}=0 & q_{-2}=1
\end{array}
$$

## Remark

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

$$
x_{0}=x \quad z_{k}=\left\lfloor x_{k}\right\rfloor \quad x_{k+1}=\frac{1}{x_{k}-\left\lfloor x_{k}\right\rfloor}
$$

## 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: $\left[z_{0} ; z_{1}, \ldots, z_{n}\right]=\frac{p_{n}}{q_{n}}$

$$
\begin{array}{lll}
p_{k}=z_{k} p_{k-1}+p_{k-2} & p_{-1}=1 & p_{-2}=0 \\
q_{k}=z_{k} q_{k-1}+q_{k-2} & q_{-1}=0 & q_{-2}=1
\end{array}
$$

## Remark

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

$$
x_{0}=x \quad z_{k}=\left\lfloor x_{k}\right\rfloor \quad x_{k+1}=\frac{1}{x_{k}-\left\lfloor x_{k}\right\rfloor}
$$

But we are mainly interested in Rationals and finite fractions.

## Bonus Slide

Can be infinite

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

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

## Bonus Slide

Can be infinite

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

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

## Bonus Slide

Can be infinite

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

yields $x=1+\frac{1}{x}$, hence $x$ is golden ratio $\varphi$, 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

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

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

## Theorem

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

- $\varphi$ has $n$-th convergent $\frac{F_{n+2}}{F_{n+1}}$, with Fibonacci numbers $F_{0}=0$
- $\varphi$ 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

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

## Properties of Continued Fractions

Improving: each step improves approximation
Alternating: even $\rightarrow$ smaller, odd $\rightarrow$ larger value

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

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

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

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

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

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: $\left|x-\frac{p}{q}\right|<\frac{1}{2 q^{2}} \Longrightarrow \frac{p}{q}$ is a cont.frac.

## Small Private Exponent $d$

Theorem (Wiener, 1989, slightly generalised)
Assume $q<p<a q, e<\varphi(n)$ 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 $q<p<a q, e<\varphi(n)$ 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 $\left|\frac{e}{n}-\frac{k}{d}\right|<\ldots<\frac{1}{2 d^{2}}$


## Small Private Exponent $d$

Theorem (Wiener, 1989, slightly generalised)
Assume $q<p<a q, e<\varphi(n)$ 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 $\left|\frac{e}{n}-\frac{k}{d}\right|<\ldots<\frac{1}{2 d^{2}}$
- $\Longrightarrow \frac{k}{d}$ is a cont.frac. of $\frac{e}{n}$


## Small Private Exponent $d$

Theorem (Wiener, 1989, slightly generalised)
Assume $q<p<a q, e<\varphi(n)$ 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 $\left|\frac{e}{n}-\frac{k}{d}\right|<\ldots<\frac{1}{2 d^{2}}$
- $\Longrightarrow \frac{k}{d}$ is a cont.frac. of $\frac{e}{n}$
- compute all continued fractions $\sim$ list of $\log n$ candidates
a) check decoding: $2^{\text {ed }} \bmod n \stackrel{?}{=} 2$
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:

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

## 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:

$$
\begin{aligned}
&\left|\frac{e}{n}-\frac{k}{d}\right|=\left|\frac{e d-k \varphi(n)-k n+k \varphi(n)}{n d}\right| \\
&=\left|\frac{1-k(n-\varphi(n))}{n d}\right|<\frac{(a+1) k \sqrt{n}}{n d}=\frac{(a+1) k}{d \sqrt{n}} \\
& \Longrightarrow\left|\frac{e}{n}-\frac{k}{d}\right|<\frac{a+1}{\sqrt{n}} \leq \frac{a+1}{2(a+1) d^{2}}=\frac{1}{2 d^{2}}
\end{aligned}
$$

## 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:

$$
\begin{aligned}
&\left|\frac{e}{n}-\frac{k}{d}\right|=\left|\frac{e d-k \varphi(n)-k n+k \varphi(n)}{n d}\right| \\
&=\left|\frac{1-k(n-\varphi(n))}{n d}\right|<\frac{(a+1) k \sqrt{n}}{n d}=\frac{(a+1) k}{d \sqrt{n}} \\
& \Longrightarrow\left|\frac{e}{n}-\frac{k}{d}\right|<\frac{a+1}{\sqrt{n}} \leq \frac{a+1}{2(a+1) d^{2}}=\frac{1}{2 d^{2}}
\end{aligned}
$$

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

## Example (Wiener Attack)

- assume given public key

$$
n=389033 \quad e=332383
$$

- calculate continued fractions

$$
\begin{array}{rlrl}
\frac{332383}{389033} & =\frac{1}{1+\frac{56560}{332383}} & & \sim 1 \\
& =\frac{1}{1+\frac{1}{5+\frac{49133}{56550}}} & & \sim \frac{5}{6} \\
& =\frac{1}{1+\frac{1}{5+\frac{1}{451 T}}} & \sim \frac{6}{7}
\end{array}
$$

- 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 $e^{\prime}=e+* \cdot \varphi(n)$, destroys assumption $e<\varphi(n)$
- optimised decryption: make $d_{p}=d \bmod p-1$ and

$$
d_{q}=d \bmod q-1 \text { small-ish }
$$

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

$$
\operatorname{enc}(\operatorname{dec}(m))=\operatorname{dec}(\operatorname{enc}(m))
$$

## Desired Property

often Encryption/Decryption commute

$$
\operatorname{enc}(\operatorname{dec}(m))=\operatorname{dec}(\operatorname{enc}(m))
$$

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?


## Desired Property

often Encryption/Decryption commute

$$
\operatorname{enc}(\operatorname{dec}(m))=\operatorname{dec}(\operatorname{enc}(m))
$$

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?


## Desired Property

often Encryption/Decryption commute

$$
\operatorname{enc}(\operatorname{dec}(m))=\operatorname{dec}(\operatorname{enc}(m))
$$

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?


## Problem

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


## Mathematical Model

## Definition (Signature System)

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

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

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

- sign, vrfy are efficiently computable


## Observations

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


## Observations

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

Improved Plain-RSA signature

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


## Observations

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

Improved Plain-RSA signature

- Alice computes $s:=\operatorname{sign}(m,(n, d))=m^{d} \bmod n$
- Send $(m, s)$ to Bob
- Bob gets $\left(m^{\prime}, s^{\prime}\right)$, checks $m^{\prime}=s^{\prime e} \bmod n$. 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 $\left(m_{1}, s_{1}\right)$ and $\left(m_{2}, s_{2}\right)$ are valid pairs, then $\left(m_{1} m_{2}, s_{1} s_{2}\right)$ also is valid.

$$
m_{1}=s_{1}^{e} \quad m_{2}=s_{2}^{e} \Longrightarrow m_{1} m_{2}=\left(s_{1} s_{2}\right)^{e}
$$

## 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 $m^{\prime} \neq m$ (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 $m^{\prime} \neq m$ (or some restricted subset)


## Attack

- factor $m=m_{1} \ldots m_{k} \bmod n$ 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 $i=1, \ldots, k$
- have signature $s=\prod s_{i}$ for $m$


## Example (Signature Oracle Attack)

- oracle accepts messages of printable characters
- checking with

```
s = input('signature: ')
m = long_to_bytes(pow(s, e, n))
if !strcmp(m,'flag') { print(flag); }
```


## Example (Signature Oracle Attack)

- oracle accepts messages of printable characters
- checking with

```
s = input('signature: ')
m = long_to_bytes(pow(s, e, n))
if !strcmp(m,'flag') { print(flag); }
```

- factor flag: 5•499•688729
- does not work,


## Example (Signature Oracle Attack)

- oracle accepts messages of printable characters
- checking with

```
s = input('signature: ')
m = long_to_bytes(pow(s, e, n))
if !strcmp(m,'flag') { print(flag); }
```

- factor flag: 5•499•688729
- does not work, but we can add 0-Bytes
- factoring flag $\backslash x 00 \backslash x 00$ yields
$2^{6}$
$499 \cdot 2^{5}$
$5 \cdot 2^{3}$
$688729 \cdot 2^{2}$
©
$>{ }^{\prime}$
(
* $\backslash$ td
$\sim 4$ valid messages, get 4 signatures
- then send product of these 4 signatures


## Forgery for Small e

- Assume $e$ is small (e.g. $e=3$ ) and checking is done in $C$ with strcmp.
- Originally: find $s$ with $s^{e} \equiv m \bmod n$.


## Forgery for Small e

- Assume $e$ is small (e.g. $e=3$ ) and checking is done in $C$ with strcmp.
- Originally: find $s$ with $s^{e} \equiv m \bmod n$.
- Simplified Forgery: find $s$ such that $s^{e} \bmod n$ starts with $\mathrm{msg}+\mathrm{b}$ ' $\backslash \mathrm{x} 00$ '.
while $\|m s g\|<\log n$ do

$$
\mathrm{msg}+=\mathrm{b} \backslash \mathrm{x} 00 \text { ' }
$$

$$
m=\operatorname{int}(\mathrm{msg})
$$

$$
s=\lceil\sqrt[e]{m}\rceil
$$

if not $\operatorname{strcmp}\left(m s g, \operatorname{str}\left(s^{e}\right)\right)$ then return $s$

## Forgery for Small e

- Assume $e$ is small (e.g. $e=3$ ) and checking is done in $C$ with strcmp.
- Originally: find $s$ with $s^{e} \equiv m \bmod n$.
- Simplified Forgery: find $s$ such that $s^{e} \bmod n$ starts with $\mathrm{msg}+\mathrm{b}$ ' $\backslash \mathrm{x} 00$ '.
while $\|m s g\|<\log n$ do

$$
\mathrm{msg}+=\mathrm{b} \text { ' } \mathrm{x} 00 \text { ' }
$$

$$
m=\operatorname{int}(\mathrm{msg})
$$

$$
s=\lceil\sqrt[e]{m}\rceil
$$

if not $\operatorname{strcmp}\left(m s g, \operatorname{str}\left(s^{e}\right)\right)$ then return $s$

- works for $m<\sqrt[e]{n} / 256$
- Similar attacks for other types of checking, e.g. compare only beginning, check for occurrence
- also check length of message


## 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 $m \notin\left\{m_{1}, \ldots, m_{k}\right\}$


## 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 $m \notin\left\{m_{1}, \ldots, m_{k}\right\}$
sEUF-CMA strong EUF-CMA
- Eve may request signatures $s_{i}=\operatorname{sign}\left(m_{i}\right)$
- forges pair $(m, s) \notin\left\{\left(m_{i}, s_{i}\right): i=1, \ldots\right\}$;
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 $m \notin\left\{m_{1}, \ldots, m_{k}\right\}$
sEUF-CMA strong EUF-CMA
- Eve may request signatures $s_{i}=\operatorname{sign}\left(m_{i}\right)$
- forges pair $(m, s) \notin\left\{\left(m_{i}, s_{i}\right): i=1, \ldots\right\}$;
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 $h:$ Byte $^{*} \rightarrow$ 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 $h:$ Byte $^{*} \rightarrow$ 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 $M$ : (seed, $\ell) \mapsto$ Byte $^{\ell}$
$T \leftarrow$ empty string for $c=0$ to $\lceil\ell / h L e n\rceil-1$ do
$T \leftarrow T \| h(\operatorname{seed} \| c)$


## OAEP-Encryption

## Encryption

- ( $n, e$ ) public RSA key
- $m$ message, $\|m\| \leq\|n\|-2 h$ Len -2
- L label (optional), default empty


## OAEP-Encryption

## Encryption

- ( $n, e$ ) public RSA key
- $m$ message, $\|m\| \leq\|n\|-2 h$ Len -2
- L label (optional), default empty
function $\operatorname{ENCRYPT}(m, L)$
DB $\leftarrow h(L)\|00 \ldots 0001\| m$
Seed $\leftarrow$ random seed of length hLen
$\mathrm{mDB} \leftarrow M($ Seed $) \oplus \mathrm{DB}$
$\mathrm{mSeed} \leftarrow$ Seed $\oplus M(\mathrm{mDB})$
EM $\leftarrow 00 \|$ mSeed $|\mid$ mDB
return $\mathrm{EM}^{e} \bmod n$


## OAEP-Encryption

## Encryption

- ( $n, e$ ) public RSA key
- $m$ message, $\|m\| \leq\|n\|-2 h$ Len -2
- L label (optional), default empty
function $\operatorname{ENCRYPT}(m, L)$
$\mathrm{DB} \leftarrow h(L)\|00 \ldots 0001\| m$
Seed $\leftarrow$ random seed of length hLen
$\mathrm{mDB} \leftarrow M($ Seed $) \oplus \mathrm{DB}$
$\mathrm{mSeed} \leftarrow$ Seed $\oplus M(\mathrm{mDB})$
EM $\leftarrow 00 \|$ mSeed $\| \mathrm{mDB}$
return $\mathrm{EM}^{e} \bmod n$
- Payload m: 50\%-89\% of cipher, $\geq 1000$ Bit
- more than enough for AES key
- continue with symmetric encryption



## 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: EM = 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: EM = encode( $m$ )
- apply RSA primitive/plain-RSA: $s=\mathrm{EM}^{d} \bmod n$


## Verify

- apply RSA primitive/plain-RSA: 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: EM = encode( $m$ )
- apply RSA primitive/plain-RSA: $s=\mathrm{EM}^{d} \bmod n$


## Verify

- apply RSA primitive/plain-RSA: EM $=s^{e} \bmod n$
- check consistency with EMSA-PSS-VERIFY
- apply scheme on hash $(m)$ instead of $m$
- can sign arbitrarily long message (document)


## RSASSA-PSS - Details

Arguments

- $m$ message to be signed
- $h$ hash function
- $M$ mask generation function
- sLen salt length (bytes), mostly hash length or 0
- $L$ desired output length, $\geq\|h(*)\|+$ sLen +2


## RSASSA-PSS - Details

Arguments

- $m$ message to be signed
- $h$ hash function
- $M$ mask generation function
- sLen salt length (bytes), mostly hash length or 0
- $L$ desired output length, $\geq\|h(*)\|+$ sLen +2


## Encode

- $m^{\prime}=00 \ldots 00\|h(m)\|$ salt with 8 Zero-bytes
- $\mathrm{DB}=00 \ldots 0001 \|$ salt of length $L-\|h(*)\|-1$
- mask DB with $M\left(h\left(m^{\prime}\right)\right)$
- output $\mathrm{EM}=$ maskedDB $\left\|h\left(m^{\prime}\right)\right\| 0 \times b c$


## 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, $\geq\|h(*)\|+$ sLen +2


## Decode

- split $E M$ by length to get the parts maskedDB ${ }^{\prime}, H^{\prime}$
- with $H^{\prime}$ unmask to get $D B^{\prime}$
- know salt length, so get salt ${ }^{\prime}$
- construct $m^{\prime}=00 \ldots 00\|h(m)\|$ salt $^{\prime}$
- check $H^{\prime}=h\left(m^{\prime}\right)$,
- if yes (and all hardcoded bytes correct), accept



## 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 $\|n\| \geq\|m\|+2 h$ Len +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 $\|n\| \geq\|m\|+2 h$ Len +2
- maximal payload


## RSASSA-PSS/Signing

- $L$ desired output length, so $L=\|n\|$
- one hash, one salt, two fixed bytes
- only restriction $\|n\| \geq$ hLen + sLen +2
- no restriction on $m$, as hashed anyway


## Why so complicated

- PKCS \#1 v1.5 was easier, for encryption we have

$$
00|\mid 02 \text { || random || } 00| \mid m_{0}
$$

- 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

$$
00 \text { || } 02 \text { || random || } 00 \text { || } m_{0}
$$

- 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

$$
00|\mid 02 \text { || random || } 00| \mid m_{0}
$$

- 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

$$
00 \text { || } 02 \text { || random || } 00 \text { || } m_{0}
$$

- 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.

## Bleichenbacher's Attack - Decrypt

- let $B$ bound on $m:=\left(\right.$ random $\left.\|00\| m_{0}\right)$
- input $c$, get the information whether $2 B \leq c^{d} \bmod n \leq 3 B$
- given $c_{0}=m^{e} \bmod n$, find $s_{i}$ such that $c_{0}\left(s_{i}\right)^{e}$ is accepted
- $M_{i}$ set of intervals, one contains $m, M_{0}=\{[2 B, 3 B]\}$ finished if $M_{*}=\{[m, m]\}$, i.e. one singleton


## Bleichenbacher's Attack - Decrypt

- let $B$ bound on $m:=\left(\right.$ random $\left.\|00\| m_{0}\right)$
- input $c$, get the information whether $2 B \leq c^{d} \bmod n \leq 3 B$
- given $c_{0}=m^{e} \bmod n$, find $s_{i}$ such that $c_{0}\left(s_{i}\right)^{e}$ is accepted
- $M_{i}$ set of intervals, one contains $m, M_{0}=\{[2 B, 3 B]\}$ finished if $M_{*}=\{[m, m]\}$, i.e. one singleton
- $s_{i}$ is conform if $2 B \leq m s_{i} \bmod n<3 B$
- assume $m \in[a, b]$

$$
\begin{array}{rr}
2 B \leq m s_{i}-r n \leq 3 B-1 & \text { for some } r \in \mathbb{N} \\
\xrightarrow{a \leq m \leq b} a s_{i}-(3 B-1) \leq r n \leq b s_{i}-2 B \quad \text { inductive bounds }
\end{array}
$$

some candidates for $r$, for each

$$
\frac{2 B+r n}{s_{i}} \leq m \leq \frac{3 B-1+r n}{s_{i}}
$$

$i=1$ : smallest $s_{1} \geq n /(3 B)$ s.t. conform
$\left|M_{i-1}\right|>1$ : smallest $s_{i}>s_{i-1}$ s.t. conform
$\left|M_{i-1}\right|=1$ : find smallest $r_{i}$, then $s_{i}$ with

$$
r_{i} \geq 2 \frac{b s_{i-1}-2 B}{n}
$$

$$
\frac{2 B+r_{i} n}{b} \leq s_{i} \leq \frac{3 B+r_{i} n}{a}
$$

$i=1$ : smallest $s_{1} \geq n /(3 B)$ s.t. conform
$\left|M_{i-1}\right|>1$ : smallest $s_{i}>s_{i-1}$ s.t. conform
$\left|M_{i-1}\right|=1$ : find smallest $r_{i}$, then $s_{i}$ with

$$
r_{i} \geq 2 \frac{b s_{i-1}-2 B}{n} \quad \frac{2 B+r_{i} n}{b} \leq s_{i} \leq \frac{3 B+r_{i} n}{a}
$$

- combine both old and new bounds

$$
\begin{aligned}
M_{i}=\{ & {\left[\max \left(a, \frac{2 B+r n}{s_{i}}\right), \min \left(b, \frac{3 B-1+r n}{s_{i}}\right)\right] } \\
& \left.:[a, b] \in M_{i}, \frac{a s_{i}-3 B+1}{n} \leq r \leq \frac{b s_{i}-2 B}{n}\right\}
\end{aligned}
$$

$i=1$ : smallest $s_{1} \geq n /(3 B)$ s.t. conform
$\left|M_{i-1}\right|>1$ : smallest $s_{i}>s_{i-1}$ s.t. conform
$\left|M_{i-1}\right|=1$ : find smallest $r_{i}$, then $s_{i}$ with

$$
r_{i} \geq 2 \frac{b s_{i-1}-2 B}{n} \quad \frac{2 B+r_{i} n}{b} \leq s_{i} \leq \frac{3 B+r_{i} n}{a}
$$

- combine both old and new bounds

$$
\begin{aligned}
M_{i}=\{ & {\left[\max \left(a, \frac{2 B+r n}{s_{i}}\right), \min \left(b, \frac{3 B-1+r n}{s_{i}}\right)\right] } \\
& \left.:[a, b] \in M_{i}, \frac{a s_{i}-3 B+1}{n} \leq r \leq \frac{b s_{i}-2 B}{n}\right\}
\end{aligned}
$$

- 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 $\operatorname{Prime}(n)$
for $i=2, \ldots,\lfloor\sqrt{n}\rfloor$ do if $n \bmod i=0$ then return False
return True

## Naive Test

## Naive Test

function $\operatorname{Prime}(n)$
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 $\operatorname{gcd}(a, p)=1$, then $a^{p-1} \equiv 1 \bmod p$.
Only implication! No "if and only if"!

## Fermat Test

Lemma
If $p$ is prime, and $\operatorname{gcd}(a, p)=1$, then $a^{p-1} \equiv 1 \bmod p$.
Only implication! No "if and only if"!

## Fermat Test

- pick some random $a<n$
- optional: check $\operatorname{gcd}(a, n)=1$
- check whether $n$ satisfies this lemma


## Fermat Test

## Lemma

If $p$ is prime, and $\operatorname{gcd}(a, p)=1$, then $a^{p-1} \equiv 1 \bmod p$.
Only implication! No "if and only if"!

## Fermat Test

- pick some random $a<n$
- optional: check $\operatorname{gcd}(a, n)=1$
- check whether $n$ satisfies this lemma


## Example

- Let $n=97, a=68$. Have $a^{n-1} \bmod n=1$, so $n$ passes.


## Fermat Test

## Lemma

If $p$ is prime, and $\operatorname{gcd}(a, p)=1$, then $a^{p-1} \equiv 1 \bmod p$.
Only implication! No "if and only if"!

## Fermat Test

- pick some random $a<n$
- optional: check $\operatorname{gcd}(a, n)=1$
- check whether $n$ satisfies this lemma


## Example

- Let $n=97, a=68$. Have $a^{n-1} \bmod n=1$, so $n$ passes.
- Let $n=91, a=23$ yields $a^{n-1} \bmod n=1$, so $n$ passes.


## Fermat Test

## Lemma

If $p$ is prime, and $\operatorname{gcd}(a, p)=1$, then $a^{p-1} \equiv 1 \bmod p$.
Only implication! No "if and only if"!

## Fermat Test

- pick some random $a<n$
- optional: check $\operatorname{gcd}(a, n)=1$
- check whether $n$ satisfies this lemma


## Example

- Let $n=97, a=68$. Have $a^{n-1} \bmod n=1$, so $n$ passes.
- Let $n=91, a=23$ yields $a^{n-1} \bmod n=1$, so $n$ passes. But $a=19 \Longrightarrow a^{n-1} \bmod n=64 \Longrightarrow n$ is not prime.


## Fermat Test

## Lemma

If $p$ is prime, and $\operatorname{gcd}(a, p)=1$, then $a^{p-1} \equiv 1 \bmod p$.
Only implication! No "if and only if"!

## Fermat Test

- pick some random $a<n$
- optional: check $\operatorname{gcd}(a, n)=1$
- check whether $n$ satisfies this lemma


## Example

- Let $n=97, a=68$. Have $a^{n-1} \bmod n=1$, so $n$ passes.
- Let $n=91, a=23$ yields $a^{n-1} \bmod n=1$, so $n$ passes. But $a=19 \Longrightarrow a^{n-1} \bmod n=64 \Longrightarrow n$ 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 $x^{2} \equiv 1 \bmod n$ 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 $\operatorname{gcd}(a, n) \neq 1$ then return False
write $n-1=2^{s} \cdot k$, for $k$ odd
if $a^{k} \equiv 1 \bmod n$ then return True
for $i=0, \ldots, s-1$ do
if $\left(a^{k}\right)^{2^{i}} \equiv-1 \bmod n$ then return True
return False

## Miller-Rabin Test

## Miller-Rabin-Test

function Miller-Rabin(n)
pick random $a<n$
if $\operatorname{gcd}(a, n) \neq 1$ then return False
write $n-1=2^{s} \cdot k$, for $k$ odd
if $a^{k} \equiv 1 \bmod n$ then return True
for $i=0, \ldots, s-1$ do
if $\left(a^{k}\right)^{2^{i}} \equiv-1 \bmod n$ 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
- $\sim$ 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 $a=38$
- $a^{45} \equiv 90 \equiv-1 \bmod 91$
- hence accepted (for now)


## Example Miller-Rabin

## Example

- Let $n=91$
- write $n-1=90=45 \cdot 2^{1}$
- pick base $a=38$
- $a^{45} \equiv 90 \equiv-1 \bmod 91$
- hence accepted (for now)
- pick base $a=23$
- $a^{45} \bmod 91=64 \neq \pm 1$
- $\left(a^{45}\right)^{2} \bmod 91=1$
- hence composite


## Prime and Prejudice

- 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

- 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

$$
S(n)=\left(\prod \operatorname{gcd}\left(k, k_{i}\right)\right)\left(\frac{2^{\min \left(s_{i}\right) \cdot m}-1}{2^{m}-1}+1\right)
$$

$$
\begin{aligned}
& n=2^{s} \cdot k+1=\prod{ }_{i=1}^{m} p_{i}^{e_{i}} \quad p_{i}=2^{s_{i}} k_{i}+1 \\
& S(n)=\left(\prod \operatorname{gcd}\left(k, k_{i}\right)\right)\left(\frac{2^{\min \left(s_{i}\right) \cdot m}-1}{2^{m}-1}+1\right)
\end{aligned}
$$

## Corollary

Let $x$ odd with with $2 x+1$ and $4 x+1$ prime. Then $n=(2 x+1)(4 x+1)$ achieves the worst error chance for Miller-Rabin.

$$
\begin{aligned}
& n=2^{s} \cdot k+1=\prod{ }_{i=1}^{m} p_{i}^{e_{i}} \quad p_{i}=2^{s_{i}} k_{i}+1 \\
& S(n)=\left(\prod \operatorname{gcd}\left(k, k_{i}\right)\right)\left(\frac{2^{\min \left(s_{i}\right) \cdot m}-1}{2^{m}-1}+1\right)
\end{aligned}
$$

## Corollary

Let $x$ odd with with $2 x+1$ and $4 x+1$ prime. Then $n=(2 x+1)(4 x+1)$ achieves the worst error chance for Miller-Rabin.
apply formula.

- $p_{1}=2 x+1, p_{2}=4 x+1$, so $k_{1}=k_{2}=x, s_{1}=1, s_{2}=2$
- $n=8 x^{2}+6 x+1$, so $s=1$ and $k=4 x^{2}+3 x$
- $\operatorname{gcd}\left(k, k_{i}\right)=x$, hence $S(n)=2 x^{2}=\frac{\varphi(n)}{4}$

$$
\begin{aligned}
& n=2^{s} \cdot k+1=\prod{ }_{i=1}^{m} p_{i}^{e_{i}} \quad p_{i}=2^{s_{i}} k_{i}+1 \\
& S(n)=\left(\prod \operatorname{gcd}\left(k, k_{i}\right)\right)\left(\frac{2^{\min \left(s_{i}\right) \cdot m}-1}{2^{m}-1}+1\right)
\end{aligned}
$$

## Corollary

Let $x$ odd with with $2 x+1$ and $4 x+1$ prime. Then $n=(2 x+1)(4 x+1)$ achieves the worst error chance for Miller-Rabin.
apply formula.

- $p_{1}=2 x+1, p_{2}=4 x+1$, so $k_{1}=k_{2}=x, s_{1}=1, s_{2}=2$
- $n=8 x^{2}+6 x+1$, so $s=1$ and $k=4 x^{2}+3 x$
- $\operatorname{gcd}\left(k, k_{i}\right)=x$, hence $S(n)=2 x^{2}=\frac{\varphi(n)}{4}$

Construction: guess $x$ and check primality

## Miller-Rabin - Wrap-Up

## Do's

- stick to the pseudo-code
- use random bases
- $t$ rounds give $2 t$ bit security level


## Miller-Rabin - Wrap-Up

Do's

- stick to the pseudo-code
- use random bases
- $t$ rounds give $2 t$ 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
return $\left\{\begin{aligned} a_{i \cdot j} & \leftarrow 0 \\ : a_{p} & =1\}\end{aligned}\right.$


## 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

$$
\begin{gathered}
\text { for } j=2, \ldots,\lfloor n / i\rfloor \text { do } \\
a_{i \cdot j}^{\leftarrow} \leftarrow 0 \\
\text { return }\left\{p: a_{p}=1\right\}
\end{gathered}
$$

- 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 $\sim$ "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 $p=k M+\left(65537^{a} \bmod M\right)$


## 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 $p=k M+\left(65537^{a} \bmod M\right)$
- $p$ 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 Icm


## 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 Icm but it gets worse


## ROCA - Return of Coppersmith Attack

## Theorem (Coppersmith)

Let $p=\sum a_{i j} x^{i} y^{j} \in \mathbb{Z}[x, y]$ irreducible; $X, Y$ bounds for solutions. Put $W:=\max \left\{\left|a_{i j}\right| \cdot X^{i} Y^{j}: i, j\right\}$ and $\delta=\max \left(\operatorname{deg}_{x}(p), \operatorname{deg}_{y}(p)\right)$. Assume $X Y<W^{2 /(3 \delta)}$. Then we can find integer root $\left(x_{0}, y_{0}\right)$ with $\left|x_{0}\right|<X,\left|y_{0}\right|<Y$, if it exists.

Find "small" integer roots in 2 variables.

## ROCA - Return of Coppersmith Attack

## Theorem (Coppersmith)

Let $p=\sum a_{i j} x^{i} y^{j} \in \mathbb{Z}[x, y]$ irreducible; $X, Y$ bounds for solutions. Put $W:=\max \left\{\left|a_{i j}\right| \cdot X^{i} Y^{j}: i, j\right\}$ and $\delta=\max \left(\operatorname{deg}_{x}(p), \operatorname{deg}_{y}(p)\right)$. Assume $X Y<W^{2 /(3 \delta)}$. Then we can find integer root $\left(x_{0}, y_{0}\right)$ with $\left|x_{0}\right|<X,\left|y_{0}\right|<Y$, if it exists.

Find "small" integer roots in 2 variables.

$$
\begin{gathered}
p=k M+\left(65537^{a} \bmod M\right) \quad q=\ell M+\left(65537^{b} \bmod M\right) \\
n=p q=* \cdot M+\left(65537^{a+b} \bmod M\right)
\end{gathered}
$$

## ROCA - Return of Coppersmith Attack

## Theorem (Coppersmith)

Let $p=\sum a_{i j} x^{i} y^{j} \in \mathbb{Z}[x, y]$ irreducible; $X, Y$ bounds for solutions. Put $W:=\max \left\{\left|a_{i j}\right| \cdot X^{i} Y^{j}: i, j\right\}$ and $\delta=\max \left(\operatorname{deg}_{x}(p), \operatorname{deg}_{y}(p)\right)$. Assume $X Y<W^{2 /(3 \delta)}$. Then we can find integer root $\left(x_{0}, y_{0}\right)$ with $\left|x_{0}\right|<X,\left|y_{0}\right|<Y$, if it exists.

Find "small" integer roots in 2 variables.

$$
\begin{gathered}
p=k M+\left(65537^{a} \bmod M\right) \quad q=\ell M+\left(65537^{b} \bmod M\right) \\
n=p q=* \cdot M+\left(65537^{a+b} \bmod M\right)
\end{gathered}
$$

- $a+b=\log _{65537}(n \bmod M)$,
easy to compute, since $M$ has only small primes
(compute for each prime, compose with CRT)
- guess $a$, yields $b$, then compute $k, \ell$ via Coppersmith


## ROCA - Application

Assume 2048 bit key $\sim 1024$ bit prime

$$
s=126 \quad M \sim 971 \mathrm{bit} \quad k<2^{53}
$$

## ROCA - Application

Assume 2048 bit key $\sim 1024$ bit prime

$$
s=126 \quad M \sim 971 \text { bit } \quad k<2^{53}
$$

For some given guess of $a$ (and $b$ ) we have

$$
\begin{gathered}
A:=65537^{a} \bmod M \quad B:=65537^{b} \bmod M \\
p(x, y)=M x y+A y+B x+(A B-n) / M
\end{gathered}
$$

## ROCA - Application

Assume 2048 bit key $\sim 1024$ bit prime

$$
s=126 \quad M \sim 971 \text { bit } \quad k<2^{53}
$$

For some given guess of $a$ (and $b$ ) we have

$$
\begin{gathered}
A:=65537^{a} \bmod M \quad B:=65537^{b} \bmod M \\
p(x, y)=M x y+A y+B x+(A B-n) / M \\
\delta=1 \quad X=Y=2^{53}
\end{gathered}
$$

$$
W=\max (M X Y, A Y, B X,(A B-n) / M)=M X Y
$$

## ROCA - Application

Assume 2048 bit key $\sim 1024$ bit prime

$$
s=126 \quad M \sim 971 \mathrm{bit} \quad k<2^{53}
$$

For some given guess of $a$ (and $b$ ) we have

$$
\begin{gathered}
A:=65537^{a} \bmod M \quad B:=65537^{b} \bmod M \\
p(x, y)=M x y+A y+B x+(A B-n) / M \\
\delta=1 \quad X=Y=2^{53} \\
W=\max (M X Y, A Y, B X,(A B-n) / M)=M X Y
\end{gathered}
$$

Then check $X Y<W^{2 / 3}$, which holds (by far). Hence, Coppersmith finds solution $k, \ell$, i.e. the primes

## Refining ROCA

## Problem <br> guessing a still has $\sim 255$ bits in our example $\sim$ too much

## Refining ROCA

## Problem

guessing a still has $\sim 255$ bits in our example $\sim$ too much

- But we are far from the limit in Coppersmith.

$$
\text { need: } X Y<W^{2 / 3} \quad \text { have: } W=M X Y
$$

## Refining ROCA

## Problem

guessing a still has $\sim 255$ bits in our example $\sim$ too much

- But we are far from the limit in Coppersmith.

$$
\text { need: } X Y<W^{2 / 3} \quad \text { have: } W=M X Y
$$

- use divisor $M^{\prime} \mid M$, sufficient to have $M^{\prime}>X=Y$ note: $X, Y$ increase for smaller $M^{\prime}$


## Refining ROCA

## Problem

guessing a still has $\sim 255$ bits in our example $\sim$ too much

- But we are far from the limit in Coppersmith.

$$
\text { need: } X Y<W^{2 / 3} \quad \text { have: } W=M X Y
$$

- use divisor $M^{\prime} \mid M$, sufficient to have $M^{\prime}>X=Y$ note: $X, Y$ increase for smaller $M^{\prime}$
- less unknown bits for $a$, but more unknown bits for $k, \ell$
- Coppersmith takes longer, but much less attempts


## Refining ROCA

## Problem

guessing a still has $\sim 255$ bits in our example $\sim$ too much

- But we are far from the limit in Coppersmith.

$$
\text { need: } X Y<W^{2 / 3} \quad \text { have: } W=M X Y
$$

- use divisor $M^{\prime} \mid M$, sufficient to have $M^{\prime}>X=Y$ note: $X, Y$ increase for smaller $M^{\prime}$
- less unknown bits for $a$, but more unknown bits for $k, \ell$
- Coppersmith takes longer, but much less attempts
- for each key-length find optimal trade-off
- 2048 bit takes ca. 35 CPU-years, cost $\sim 1240 €$ (rough guess)
$\sim$ 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

$$
x \equiv a_{i} \quad \bmod p_{i} \quad \text { for } i=1,2, \ldots, s
$$

- increasing $s \sim$ 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

$$
x \equiv a_{i} \quad \bmod p_{i} \quad \text { for } i=1,2, \ldots, s
$$

- increasing $s \sim$ 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 $k M+a$ 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 $\leq 12$

$$
\frac{M}{\varphi(M)}=\prod \frac{p_{i}}{p_{i}-1}=\prod\left(1+\frac{1}{p_{i}-1}\right)>\sum \frac{1}{p_{i}} \sim \ln \ln p_{s}
$$

## Theory

- as before use primorial $M:=\prod_{i \leq s} p_{i}$
- prime candidate $k M+a$ 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 $\leq 12$

$$
\frac{M}{\varphi(M)}=\prod \frac{p_{i}}{p_{i}-1}=\prod\left(1+\frac{1}{p_{i}-1}\right)>\sum \frac{1}{p_{i}} \sim \ln \ln p_{s}
$$

## 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 $k M+a$ 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 $\leq 12$

$$
\frac{M}{\varphi(M)}=\prod \frac{p_{i}}{p_{i}-1}=\prod\left(1+\frac{1}{p_{i}-1}\right)>\sum \frac{1}{p_{i}} \sim \ln \ln p_{s}
$$

## 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
$\leadsto$ 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?


## 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 $n=p q$. If $p, q$ (not necessarily prime) are close, we can factor $n$.

## Fermat Factorisation

Method
Let $n=p q$. If $p, q$ (not necessarily prime) are close, we can factor $n$.

- Key Observation

$$
n=p q=\left(\frac{p+q}{2}\right)^{2}-\left(\frac{p-q}{2}\right)^{2}
$$

## Fermat Factorisation

Method
Let $n=p q$. If $p, q$ (not necessarily prime) are close, we can factor $n$.

- Key Observation

$$
n=p q=\left(\frac{p+q}{2}\right)^{2}-\left(\frac{p-q}{2}\right)^{2}
$$

Pseudocode

```
m\leftarrow\lceil\sqrt{}{n}\rceil
for i}\in\mathbb{N}\mathrm{ do
    \Deltai}=\sqrt{}{(m+i\mp@subsup{)}{}{2}-n
    if }\mp@subsup{\Delta}{i}{}\in\mathbb{N}\mathrm{ then
return }p\leftarrowm+i-\mp@subsup{\Delta}{i}{
```


## Fermat Factorisation - Analysis

$$
\begin{aligned}
\Delta_{i} & =\sqrt{(\lceil\sqrt{n}\rceil+i)^{2}-n} \\
n=p q & =\left(\frac{p+q}{2}\right)^{2}-\left(\frac{p-q}{2}\right)^{2}
\end{aligned}
$$

- 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

$$
\begin{aligned}
\Delta_{i} & =\sqrt{(\lceil\sqrt{n}\rceil+i)^{2}-n} \\
n=p q & =\left(\frac{p+q}{2}\right)^{2}-\left(\frac{p-q}{2}\right)^{2}
\end{aligned}
$$

- wlog $p>q$, then $\Delta_{i}=\frac{p-q}{2}$ in the end
- reach this when $m+i=\frac{p+q}{2}$, i.e.

$$
\begin{aligned}
i & \approx \frac{p+q}{2}-\sqrt{n}=\frac{(\sqrt{p}-\sqrt{q})^{2}+2 \sqrt{p q}}{2}-\sqrt{n} \\
& =\frac{(\sqrt{p}-\sqrt{q})^{2}}{2}=\frac{(\sqrt{p}-\sqrt{q})^{2} \cdot \sqrt{q^{2}}}{2 q}=\frac{(\sqrt{n}-q)^{2}}{2 q}
\end{aligned}
$$

## Fermat Factorisation - Analysis

$$
\begin{aligned}
\Delta_{i} & =\sqrt{(\lceil\sqrt{n}\rceil+i)^{2}-n} \\
n=p q & =\left(\frac{p+q}{2}\right)^{2}-\left(\frac{p-q}{2}\right)^{2}
\end{aligned}
$$

- wlog $p>q$, then $\Delta_{i}=\frac{p-q}{2}$ in the end
- reach this when $m+i=\frac{p+q}{2}$, i.e.

$$
\begin{aligned}
i & \approx \frac{p+q}{2}-\sqrt{n}=\frac{(\sqrt{p}-\sqrt{q})^{2}+2 \sqrt{p q}}{2}-\sqrt{n} \\
& =\frac{(\sqrt{p}-\sqrt{q})^{2}}{2}=\frac{(\sqrt{p}-\sqrt{q})^{2} \cdot \sqrt{q}^{2}}{2 q}=\frac{(\sqrt{n}-q)^{2}}{2 q}
\end{aligned}
$$

- in total: works well if $p, q$ nearly same in upper half


## Fermat Factorisation - Example

## Example

Let $n=583$, thus $m=25$

$$
\begin{array}{ll}
i=0 & \Delta_{i}^{2}=2 \cdot 3 \cdot 7 \\
i=1 & \Delta_{i}^{2}=3 \cdot 31 \\
i=2 & \Delta_{i}^{2}=2 \cdot 73 \\
i=3 & \Delta_{i}^{2}=3 \cdot 67 \\
i=4 & \Delta_{i}^{2}=2 \cdot 3 \cdot 43 \\
i=5 & \Delta_{i}^{2}=317 \\
i=6 & \Delta_{i}^{2}=2 \cdot 3^{3} \cdot 7 \\
i=7 & \Delta_{i}^{2}=3^{2} \cdot 7^{2}
\end{array}
$$

## Fermat Factorisation - Example

## Example

Let $n=583$, thus $m=25$

$$
\begin{array}{ll}
i=0 & \Delta_{i}^{2}=2 \cdot 3 \cdot 7 \\
i=1 & \Delta_{i}^{2}=3 \cdot 31 \\
i=2 & \Delta_{i}^{2}=2 \cdot 73 \\
i=3 & \Delta_{i}^{2}=3 \cdot 67 \\
i=4 & \Delta_{i}^{2}=2 \cdot 3 \cdot 43 \\
i=5 & \Delta_{i}^{2}=317 \\
i=6 & \Delta_{i}^{2}=2 \cdot 3^{3} \cdot 7 \\
i=7 & \Delta_{i}^{2}=3^{2} \cdot 7^{2}
\end{array}
$$

$\Delta_{7}=21 \sim p=25+7-21=11$ 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
- $\operatorname{gcd}(a \pm b, n)$ 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
- $\operatorname{gcd}(a \pm b, n)$ 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
- $\operatorname{gcd}(a \pm b, n)$ 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

$$
\mathcal{O}(\exp ((1+o(1)) \sqrt{\log n \cdot \log \log n}))
$$

## Quadratic Sieve - Example

Example
Let $n=583=11 \cdot 53$, use Fermat

$$
\begin{array}{lll}
i=0 & m+i=25 & \Delta_{i}^{2}=2 \cdot 3 \cdot 7 \\
i=6 & m+i=31 & \Delta_{i}^{2}=2 \cdot 3^{3} \cdot 7
\end{array}
$$

## Quadratic Sieve - Example

## Example

Let $n=583=11 \cdot 53$, use Fermat

$$
\begin{array}{lll}
i=0 & m+i=25 & \Delta_{i}^{2}=2 \cdot 3 \cdot 7 \\
i=6 & m+i=31 & \Delta_{i}^{2}=2 \cdot 3^{3} \cdot 7
\end{array}
$$

in Fermat factorisation, we have

$$
\Delta_{i}^{2}=(m+i)^{2}-n
$$

## Quadratic Sieve - Example

## Example

Let $n=583=11 \cdot 53$, use Fermat

$$
\begin{array}{lll}
i=0 & m+i=25 & \Delta_{i}^{2}=2 \cdot 3 \cdot 7 \\
i=6 & m+i=31 & \Delta_{i}^{2}=2 \cdot 3^{3} \cdot 7
\end{array}
$$

in Fermat factorisation, we have

$$
\Delta_{i}^{2}=(m+i)^{2}-n
$$

combine $i=0$ and $i=6$ to get square

$$
(25 \cdot 31)^{2} \equiv\left(2 \cdot 3^{2} \cdot 7\right)^{2} \quad \bmod 583
$$

## Quadratic Sieve - Example

## Example

Let $n=583=11 \cdot 53$, use Fermat

$$
\begin{array}{lll}
i=0 & m+i=25 & \Delta_{i}^{2}=2 \cdot 3 \cdot 7 \\
i=6 & m+i=31 & \Delta_{i}^{2}=2 \cdot 3^{3} \cdot 7
\end{array}
$$

in Fermat factorisation, we have

$$
\Delta_{i}^{2}=(m+i)^{2}-n
$$

combine $i=0$ and $i=6$ to get square

$$
(25 \cdot 31)^{2} \equiv\left(2 \cdot 3^{2} \cdot 7\right)^{2} \quad \bmod 583
$$

obtain $\operatorname{gcd}(25 \cdot 31-2 \cdot 9 \cdot 7,583)=11$

## 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 $\leadsto$ 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 $\leadsto$ exponential number of steps

Solution: factor into small primes

- more general approach
- try to factor the $\Delta_{i}$ 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 $\left(a^{2} \bmod n\right)$ is $p_{k}$-smooth
- construct $x, y$ with $x^{2} \equiv y^{2} \bmod n$
- 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 $\left(a^{2} \bmod n\right)$ is $p_{k}$-smooth
- construct $x, y$ with $x^{2} \equiv y^{2} \bmod n$
- 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

$$
\left(a_{j}^{2} \bmod n\right)=\prod_{i=1}^{k} p_{i}^{d_{i j}}
$$

- define the matrix $\boldsymbol{D} \in\{0,1\}^{(k+1) \times k}$ via $D_{i j}:=d_{i j} \bmod 2$. regard it as row vectors
- non-trivial solution $\boldsymbol{t} \cdot \boldsymbol{D}=\mathbf{0} \bmod 2 \boldsymbol{t} \in\{0,1\}^{k+1}$
- assume we have $k+1$ such numbers $a_{0}, \ldots, a_{k}$ with

$$
\left(a_{j}^{2} \bmod n\right)=\prod_{i=1}^{k} p_{i}^{d_{i j}}
$$

- define the matrix $\boldsymbol{D} \in\{0,1\}^{(k+1) \times k}$ via $D_{i j}:=d_{i j} \bmod 2$. regard it as row vectors
- non-trivial solution $\boldsymbol{t} \cdot \boldsymbol{D}=\mathbf{0} \bmod 2 \boldsymbol{t} \in\{0,1\}^{k+1}$
- note: $\frac{1}{2} \boldsymbol{t} \boldsymbol{D}$ is thus integer. put

$$
x:=\prod_{j: t_{j}=1} a_{j} \quad y:=\prod_{i=1}^{k} p_{i}^{\frac{1}{2} \sum_{j=0}^{k} t_{j} d_{i j}}
$$

- assume we have $k+1$ such numbers $a_{0}, \ldots, a_{k}$ with

$$
\left(a_{j}^{2} \bmod n\right)=\prod_{i=1}^{k} p_{i}^{d_{i j}}
$$

- define the matrix $\boldsymbol{D} \in\{0,1\}^{(k+1) \times k}$ via $D_{i j}:=d_{i j} \bmod 2$. regard it as row vectors
- non-trivial solution $\boldsymbol{t} \cdot \boldsymbol{D}=\mathbf{0} \bmod 2 \boldsymbol{t} \in\{0,1\}^{k+1}$
- note: $\frac{1}{2} \boldsymbol{t} \boldsymbol{D}$ is thus integer. put

$$
x:=\prod_{j: t_{j}=1} a_{j} \quad y:=\prod_{i=1}^{k} p_{i}^{\frac{1}{2} \sum_{j=0}^{k} t_{j} d_{i j}}
$$

- have $x^{2} \equiv y^{2} \bmod n\left(\right.$ mult. the entries with $\left.t_{j}=1\right)$
- $50 \%$ chance: $x \equiv y \bmod p$ and $x \equiv-y \bmod q$ (or vice versa)
- get $p, q$ from $\operatorname{gcd}(x+y, n)$ or $\operatorname{gcd}(x-y, n)$


## Quadratic Sieve - Algorithm

- put $m=\lceil\sqrt{n}\rceil$, empty matrix $\boldsymbol{D}$
- for $j=0,1, \ldots$ try to factor

$$
(m+j)^{2}-n=\prod p_{i}^{d_{i j}} \cdot \text { remainder }
$$

- if remainder $=1$ :
store $m+j$ and append $d_{*, j} \bmod 2$ 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

$$
(m+j)^{2}-n=\prod p_{i}^{d_{i j}} \cdot \text { remainder }
$$

- if remainder $=1$ :
store $m+j$ and append $d_{*, j} \bmod 2$ to matrix $\boldsymbol{D}$
- do Gaussian elimination on a copy $\boldsymbol{D}^{\prime}$
- break if $\boldsymbol{D}^{\prime}$ has zero-row
- construct $x^{2} \equiv y^{2} \bmod n$ as above
- $p:=\operatorname{gcd}(x \pm y, n)$, 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 | $\checkmark$ |
| 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 | $\checkmark$ |

## 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 | $\checkmark$ |
| 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 | $\checkmark$ |

already have matrix of lower rank $\sim$ break

$$
\boldsymbol{D}=\left(\begin{array}{llll}
1 & 1 & 0 & 1 \\
1 & 1 & 0 & 1
\end{array}\right)
$$

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

$$
\boldsymbol{D}=\left(\begin{array}{llll}
1 & 1 & 0 & 1 \\
1 & 1 & 0 & 1
\end{array}\right) \quad \text { steps }=\binom{0}{6}
$$

- 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

$$
\boldsymbol{D}=\left(\begin{array}{llll}
1 & 1 & 0 & 1 \\
1 & 1 & 0 & 1
\end{array}\right) \quad \text { steps }=\binom{0}{6}
$$

- element $(1,1)$ from kernel tells us, which steps to combine
- hence, we get

$$
(25 \cdot 31)^{2} \equiv\left(2 \cdot 3^{2} \cdot 7\right)^{2} \quad \bmod 583
$$

## 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 $2^{[\|n\| / e\rceil}$
- 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 $\|n\|$
$\sim$ 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 $G=\langle g\rangle=\left\{g^{n}: n \in \mathbb{N}\right\}$, finite
- our groups are commutative


## Reminder Groups

What is a group?

- set with a single operation
- have neutral element and inverse
- we use $G=\langle g\rangle=\left\{g^{n}: n \in \mathbb{N}\right\}$, 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 $n=p q$
- mathematical problem is $e$-th root modulo $n$


## Motivation

- RSA works in $\mathbb{Z}_{n}$ for $n=p q$
- 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): given $g, g^{x} \in G$, find $x \in \mathbb{N}$.


## Motivation

- RSA works in $\mathbb{Z}_{n}$ for $n=p q$
- 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):

$$
\text { given } g, g^{x} \in G \text {, find } x \in \mathbb{N} \text {. }
$$

- framework for cryptosystem, until we decide which group comparable to abstract classes in programming
- keywords: Diffie-Hellman, Elliptic Curves, DSA, EIGamal


## Motivation

- RSA works in $\mathbb{Z}_{n}$ for $n=p q$
- 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):

$$
\text { given } g, g^{x} \in G \text {, find } x \in \mathbb{N} \text {. }
$$

- framework for cryptosystem, until we decide which group comparable to abstract classes in programming
- keywords: Diffie-Hellman, Elliptic Curves, DSA, EIGamal


## 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 $\left(g^{a}\right)^{b}=g^{a b}=g^{b a}=\left(g^{b}\right)^{a}$


## 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 $K_{B}:=A^{b}=5997$


## 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 $m=c \cdot\left(B^{a}\right)^{-1}$


## 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 $m=c \cdot\left(B^{a}\right)^{-1}$
- 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 $c=m A^{b}=20948$
- 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 $c=m A^{b}=20948$
- full cipher $(B, c)$

Alice decrypts:

- original message via $\left(B^{a}\right)^{-1} \cdot c=102$


## 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

Public Knowledge: $g, G=\langle g\rangle$


## Diffie-Hellman from Eve's view

Public Knowledge: $g, G=\langle g\rangle$


## 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 $g^{a b}$
find session key
Decisional DH (DDH): given $g, g^{a}, g^{b}, h$, decide $g^{a b}=h$
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 $g^{a b}$
find session key
Decisional DH (DDH): given $g, g^{a}, g^{b}, h$, decide $g^{a b}=h$
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 $g^{a b}$ find session key
Decisional DH (DDH): given $g, g^{a}, g^{b}, h$, decide $g^{a b}=h$
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<br>\section*{Input:}<br>$g$ - generator of group<br>$y=g^{x}$ for unknown $x$<br>Output: $x$-discrete log function $\operatorname{DLP}(\mathrm{g}, \mathrm{y})$<br>for $x=0$ to $n$ do if $g^{x}=y$ then<br>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 $\operatorname{DLP}(\mathrm{g}, \mathrm{y})$
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

Group Based Cryptography Generic Attacks on DLP

## Shanks Baby-Step-Giant-Step - Picture



## Shanks Baby-Step-Giant-Step - Picture



## Shanks Baby-Step-Giant-Step - Picture



## Shanks Baby-Step-Giant-Step - Picture


found match $\rightarrow$ Stop

## Shanks Baby-Step-Giant-Step - Picture


found match $\rightarrow$ Stop

- 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 $x=k i+j$ 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 $x=k i+j$ with $j<k$
- Meet-in-the-middle:
- List all $\left(g^{x}\right) \cdot g^{-k i}$ for $0 \leq i \leq\left\lfloor\frac{n}{k}\right\rfloor$
- for $g^{j}$ check whether in list
- if match $g^{j}=g^{x} \cdot g^{-k i}$, we found $x=k i+j$


## 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 $x=k i+j$ with $j<k$
- Meet-in-the-middle:
- List all $\left(g^{x}\right) \cdot g^{-k i}$ for $0 \leq i \leq\left\lfloor\frac{n}{k}\right\rfloor$
- for $g^{j}$ check whether in list
- if match $g^{j}=g^{x} \cdot g^{-k i}$, we found $x=k i+j$
- $k \approx \sqrt{n}$ 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 $x=k i+j$ with $j<k$
- Meet-in-the-middle:
- List all $\left(g^{-x}\right) \cdot g^{-k i}$ for $0 \leq i \leq\left\lfloor\frac{n}{k}\right\rfloor$
- for $g^{j}$ check whether in list
- if match $g^{j}=g^{x} \cdot g^{-k i}$, we found $x=k i+j$
- $k \approx \sqrt{n}$ 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

$$
\mathcal{O}(\operatorname{poly}(\|n\|) \cdot \sqrt{p})
$$

- 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

$$
\mathcal{O}(\operatorname{poly}(\|n\|) \cdot \sqrt{p})
$$

- 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 $n=2 \cdot p^{\prime}$, 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 $i=1, \ldots, e-1$

$$
\left(y \cdot g^{-\left(x_{0}+\ldots+x_{i-1} p^{i-1}\right)}\right)^{p^{e-i-1}}=g^{x_{i} p^{e-1}+p^{e} \cdot *}=h^{x_{i}}
$$

## 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

$$
n_{i}:=n / p_{i}^{e_{i}} \quad g_{i}:=g^{n_{i}} \quad y_{i}:=y^{n_{i}}
$$

## 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

$$
n_{i}:=n / p_{i}^{e_{i}} \quad g_{i}:=g^{n_{i}} \quad y_{i}:=y^{n_{i}}
$$

- compute $x_{i}=\log _{g_{i}}\left(y_{i}\right)$ 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

$$
n_{i}:=n / p_{i}^{e_{i}} \quad g_{i}:=g^{n_{i}} \quad y_{i}:=y^{n_{i}}
$$

- compute $x_{i}=\log _{g_{i}}\left(y_{i}\right)$ as before, easy as group has order $p_{i}^{e_{i}}$
- via CRT solve the system

$$
x \equiv x_{i} \quad \bmod p_{i}^{e_{i}} \quad \text { for } i=1, \ldots, \ell
$$

## 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

$$
n_{i}:=n / p_{i}^{e_{i}} \quad g_{i}:=g^{n_{i}} \quad y_{i}:=y^{n_{i}}
$$

- compute $x_{i}=\log _{g_{i}}\left(y_{i}\right)$ as before, easy as group has order $p_{i}^{e_{i}}$
- via CRT solve the system

$$
x \equiv x_{i} \quad \bmod p_{i}^{e_{i}} \quad \text { for } i=1, \ldots, \ell
$$

- running time $\mathcal{O}\left(\sum_{i} e_{i}\left(\log n+\sqrt{p_{i}}\right)\right)$ 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 $\varphi(M)=\prod_{i}^{s}\left(p_{i}-1\right)$
- 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 $\mathcal{O}(\operatorname{poly}(\|n\|) \cdot \sqrt{p})$

## 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 $\mathcal{O}(\operatorname{poly}(\|n\|) \cdot \sqrt{p})$

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 $\left(\mathbb{Z}_{n},+\right)$
- multiplicative group $\left(\mathbb{Z}_{n}^{*}, \cdot\right)$
- symmetric group $S_{n}$ (permutations)
- invertible matrices $G L\left(n, p^{k}\right)=\left\{M \in G F\left(p^{k}\right)^{n \times n}: \operatorname{det} M \neq 0\right\}$


## DLP in Different Groups

Diffie-Hellman handshake is a template
Common Examples of (Finite) Groups

- additive group $\left(\mathbb{Z}_{n},+\right)$
- multiplicative group $\left(\mathbb{Z}_{n}^{*}, \cdot\right)$
- symmetric group $S_{n}$ (permutations)
- invertible matrices $G L\left(n, p^{k}\right)=\left\{M \in G F\left(p^{k}\right)^{n \times n}: \operatorname{det} M \neq 0\right\}$

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
- $\sim$ new kind of group (actually since end of 19th cent.)
- make everything discrete and finite


## Additive Groups

Additive Group $\left(\mathbb{Z}_{n},+\right)$

- 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 $\left(\mathbb{Z}_{n},+\right)$

- 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 $\varphi:\langle g\rangle \cong\left(\mathbb{Z}_{o(g)},+\right)$, with $\varphi(g)=1$, in general $\varphi\left(g^{k}\right)=k$ no matter which choice of $g$ and $G$


## Additive Groups

Additive Group $\left(\mathbb{Z}_{n},+\right)$

- 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 $\varphi:\langle g\rangle \cong\left(\mathbb{Z}_{o(g)},+\right)$, with $\varphi(g)=1$, in general $\varphi\left(g^{k}\right)=k$ no matter which choice of $g$ and $G$
- translate from any group into $\left(\mathbb{Z}_{n},+\right)$ ?
- 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

$$
\log _{(1,5,3,2,4)}((1,2,5,4,3)) \widehat{=}(1 \mapsto 5 \mapsto 3 \mapsto 2) \sim 3
$$

## 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

$$
\log _{(1,5,3,2,4)}((1,2,5,4,3)) \widehat{=}(1 \mapsto 5 \mapsto 3 \mapsto 2) \sim 3
$$

- let $\ell_{i}$ be cycle lengths, then have $x \bmod \ell_{i}$ for all $i$
- compose with (generalised) CRT modulo the Icm


## 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

$$
\log _{(1,5,3,2,4)}((1,2,5,4,3)) \widehat{=}(1 \mapsto 5 \mapsto 3 \mapsto 2) \sim 3
$$

- let $\ell_{i}$ be cycle lengths, then have $x \bmod \ell_{i}$ for all $i$
- compose with (generalised) CRT modulo the Icm
- result is bounded by

$$
o(g) \leq \ell_{1} \cdot \ldots \cdot \ell_{k} \stackrel{\text { AM-GM }}{\leq}\left(\frac{\ell_{1}+\ldots+\ell_{k}}{k}\right)^{k}=\left(\frac{n}{k}\right)^{k} \leq e^{n / e}
$$

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

$$
a \equiv 1 \quad \bmod 2 \quad a \equiv 2 \bmod 3 \quad a \equiv 2 \bmod 5
$$

with solution $a=17$

## Invertible Matrices

## General Linear Group

$$
\operatorname{GL}\left(n, p^{k}\right)=\left\{M \in \operatorname{GF}\left(p^{k}\right)^{n \times n}: \operatorname{det} M \neq 0\right\}
$$

$\operatorname{GF}(q)$ field with $q$ elements (not $\mathbb{Z}_{q}$ if $q$ is proper prime power)

## Invertible Matrices

## General Linear Group

$$
\operatorname{GL}\left(n, p^{k}\right)=\left\{M \in \operatorname{GF}\left(p^{k}\right)^{n \times n}: \operatorname{det} M \neq 0\right\}
$$

$\mathrm{GF}(q)$ field with $q$ elements (not $\mathbb{Z}_{q}$ if $q$ is proper prime power)

Theorem (Menezes, Wu, 1997)
We can transfer the DLP in $\mathrm{GL}\left(n, p^{k}\right)$ to the $D L P$ in $\operatorname{GF}\left(p^{k n}\right)$.

## Invertible Matrices

## General Linear Group

$$
\operatorname{GL}\left(n, p^{k}\right)=\left\{M \in \operatorname{GF}\left(p^{k}\right)^{n \times n}: \operatorname{det} M \neq 0\right\}
$$

$\operatorname{GF}(q)$ field with $q$ elements (not $\mathbb{Z}_{q}$ if $q$ is proper prime power)

Theorem (Menezes, Wu, 1997)
We can transfer the DLP in $\mathrm{GL}\left(n, p^{k}\right)$ to the $D L P$ in $\operatorname{GF}\left(p^{k n}\right)$.

- transfer t group with $p^{n k}-1$ elements
- attack that one like $\mathbb{Z}_{p}$
- computation with matrices more expensive
$\Longrightarrow$ matrix has no advantage over $\mathrm{GF}\left(p^{n k}\right)$


## DLP in $\mathbb{Z}_{p}$

Task: given $g, n=o(g), y=g^{x} \bmod p$, 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 $p_{i}=g^{*} \bmod p$
- find powers $g^{r}$ with

$$
g^{r} \cdot y \equiv p_{1}^{e_{1}} \ldots p_{k}^{e_{k}} \quad \bmod p
$$

## DLP in $\mathbb{Z}_{p}$

Task: given $g, n=o(g), y=g^{x} \bmod p$, 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 $p_{i}=g^{*} \bmod p$
- find powers $g^{r}$ with

$$
g^{r} \cdot y \equiv p_{1}^{e_{1}} \ldots p_{k}^{e_{k}} \quad \bmod p
$$

- enough of them give linear equation system

$$
\log _{g} y \equiv-r+e_{1} \log _{g} p_{1}+\ldots+e_{k} \log _{g} p_{k} \quad \bmod n
$$

variables $\log _{g} y$ and the $\log _{g} p_{i} \leadsto$ solve

## DLP in $\mathbb{Z}_{p}$

Task: given $g, n=o(g), y=g^{x} \bmod p$, 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 $p_{i}=g^{*} \bmod p$
- find powers $g^{r}$ with

$$
g^{r} \cdot y \equiv p_{1}^{e_{1}} \ldots p_{k}^{e_{k}} \quad \bmod p
$$

- enough of them give linear equation system

$$
\log _{g} y \equiv-r+e_{1} \log _{g} p_{1}+\ldots+e_{k} \log _{g} p_{k} \quad \bmod n
$$

variables $\log _{g} y$ and the $\log _{g} p_{i} \leadsto$ solve
running time like factoring

## DLP in $\mathbb{Z}_{p}$

## Setup

- prime $n$ such that $p=2 n+1$ is prime (Sophie-Germain prime)
- work in $\mathbb{Z}_{p}$, has order $\varphi(p)=p-1=2 n$
- 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 $p=2 n+1$ is prime (Sophie-Germain prime)
- work in $\mathbb{Z}_{p}$, has order $\varphi(p)=p-1=2 n$
- 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
- ~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 $4 a^{3}+27 b^{2} \neq 0$ (discriminant), needed to avoid degenerate case (curve behaves nicely)
- then the elliptic curve over $K$ (in Weierstrass form) is

$$
E(K):=\left\{(x, y) \in K^{2}: y^{2}=x^{3}+a x+b\right\} \cup\{\infty\}
$$

## 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 $4 a^{3}+27 b^{2} \neq 0$ (discriminant), needed to avoid degenerate case (curve behaves nicely)
- then the elliptic curve over $K$ (in Weierstrass form) is

$$
E(K):=\left\{(x, y) \in K^{2}: y^{2}=x^{3}+a x+b\right\} \cup\{\infty\}
$$

## Remark

- often used in projective coordinates, i.e. in $K^{3}$ no inversion in addition $\sim$ 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



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: $R=P \cdot Q$

$$
\begin{aligned}
\lambda & = \begin{cases}\frac{Q_{y}-P_{y}}{Q_{x}-P_{x}} & : P \neq Q \\
\frac{3 P_{x}^{2}+a}{2 P_{y}} & : P=Q\end{cases} \\
R_{x} & =\lambda^{2}-P_{x}-Q_{x} \\
R_{y} & =\lambda\left(P_{x}-R_{x}\right)-P_{y}
\end{aligned}
$$

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: $R=P \cdot Q$

$$
\begin{aligned}
\lambda & = \begin{cases}\frac{Q_{y}-P_{y}}{Q_{x}-P_{x}} & : P \neq Q \\
\frac{3 P_{x}^{2}+a}{2 P_{y}} & : P=Q\end{cases} \\
R_{x} & =\lambda^{2}-P_{x}-Q_{x} \\
R_{y} & =\lambda\left(P_{x}-R_{x}\right)-P_{y}
\end{aligned}
$$

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 EIGamal
Condition

- Chosen Cipher Attack
- Alice does not check whether $B \in E$
- work in curve $y^{2}=x^{3}+a x+b$
- computations don't use $b$
- works for $P, Q \notin E$


## Faulty Curve Injection

get Alice's secret key in EIGamal
Condition

- Chosen Cipher Attack
- Alice does not check whether $B \in E$
- work in curve $y^{2}=x^{3}+a x+b$
- 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 $B^{\prime} \in K^{2}$
- gives point on new curve $E^{\prime}: y^{2}=x^{3}+a x+b^{\prime}$
- try until order $O\left(B^{\prime}\right)$ 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 $B^{\prime} \in K^{2}$
- gives point on new curve $E^{\prime}: y^{2}=x^{3}+a x+b^{\prime}$
- try until order $o\left(B^{\prime}\right)$ has only small prime divisors
chance is good enough, offline search
- send $\left(B^{\prime-1}, \infty\right)$ (recall: $\infty$ is neutral element) usually would send $(B, c)=\left(g^{b}, m \cdot A^{b}\right)$
- decryption: $m=\infty \cdot\left(B^{\prime-1}\right)^{-a}=B^{\prime a}$ 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 $B^{\prime} \in K^{2}$
- gives point on new curve $E^{\prime}: y^{2}=x^{3}+a x+b^{\prime}$
- try until order $O\left(B^{\prime}\right)$ has only small prime divisors chance is good enough, offline search
- send $\left(B^{\prime-1}, \infty\right)$ (recall: $\infty$ is neutral element) usually would send $(B, c)=\left(g^{b}, m \cdot A^{b}\right)$
- decryption: $m=\infty \cdot\left(B^{\prime-1}\right)^{-a}=B^{\prime a}$ 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 $B^{\prime a}$


## 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 $a, b \in \mathbb{Z}_{p} \leadsto$ 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 $a, b \in \mathbb{Z}_{p} \leadsto$ defines curve
- compute size of group, want prime order
- then find generator $g$

Do once $\sim$ 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 $\mathcal{O}\left(\|p\|^{2} M\left(\|p\|^{3}\right) / \log \|p\|\right) \approx \mathcal{O}\left(\|p\|^{5}\right)$ Schoof-Elkies-Atkin: time $\mathcal{O}\left(\|p\|^{2} M\left(\|p\|^{2}\right) / \log \|p\|\right) \approx \mathcal{O}\left(\|p\|^{4}\right)$ significant improvement: $p^{\frac{1}{4}} \leadsto \operatorname{poly}(\log p)$ slow, but feasible

## Ideas Behind Methods

## Baby-Step-Giant-Step

- $|E|$ lies in interval $[p+1 \pm 2 \sqrt{p}]$ of size $4 \sqrt{p}$
- pick random $P \in E$ : pick random $x$, until $x^{3}+a x+b$ 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 $[p+1 \pm 2 \sqrt{p}]$ of size $4 \sqrt{p}$
- pick random $P \in E$ : pick random $x$, until $x^{3}+a x+b$ 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 $|E| \bmod q_{i}$ for some primes $q_{i}$
- until $\Pi q_{i}>4 \sqrt{p}$
- 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 $\sim$ long time
- factoring: only trial division $\sim$ fast
- find generator: good chance $\sim$ 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 $\sim$ long time
- factoring: only trial division $\sim$ fast
- find generator: good chance $\sim$ 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 $\sim$ long time
- factoring: only trial division $\sim$ fast
- find generator: good chance $\sim$ 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 $g^{r} \leadsto$ 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 $2 k$ )


## 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 $2 k$ )

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 $\operatorname{gcd}(\ell, p)=1, k$ minimal with $\ell \mid p^{k}-1$
- can transfer DLP to $\left(\operatorname{GF}\left(p^{k}\right)^{*}, \cdot\right)$, subexponential solutions


## Other Insecure Special Cases

Standard curves are also tested against other attacks.

## Multiplicative Transfer

- let $\ell=o(g)$ with $\operatorname{gcd}(\ell, p)=1, k$ minimal with $\ell \mid p^{k}-1$
- can transfer DLP to $\left(\operatorname{GF}\left(p^{k}\right)^{*}, \cdot\right)$, subexponential solutions


## Additive Transfer

- anomalous curve: $|E|=p$
- can transfer DLP to $\left(\mathbb{Z}_{p},+\right)$, easy to solve


## Other Insecure Special Cases

Standard curves are also tested against other attacks.

## Multiplicative Transfer

- let $\ell=o(g)$ with $\operatorname{gcd}(\ell, p)=1, k$ minimal with $\ell \mid p^{k}-1$
- can transfer DLP to $\left(\operatorname{GF}\left(p^{k}\right)^{*}, \cdot\right)$, subexponential solutions


## Additive Transfer

- anomalous curve: $|E|=p$
- can transfer DLP to $\left(\mathbb{Z}_{p},+\right)$, 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 $\operatorname{gcd}(\ell, p)=1, k$ minimal with $\ell \mid p^{k}-1$
- can transfer DLP to $\left(\operatorname{GF}\left(p^{k}\right)^{*}, \cdot\right)$, subexponential solutions


## Additive Transfer

- anomalous curve: $|E|=p$
- can transfer DLP to $\left(\mathbb{Z}_{p},+\right)$, 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 $\sim$ 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 $\sim$ 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 EIGamal or continue with AES


## Encryption with Elliptic Curves - Overview

## Encryption

- choose one of the standard curves
- all in every standard library $\sim$ 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 EIGamal 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 $(x, y)=g^{k}$
- $r=x \bmod n$
- $s=k^{-1}(\operatorname{hash}(\mathrm{msg})+r \cdot a) \bmod n$
- signature $(r, s)$
- $(x, y)=g^{k}$ and $r=x \bmod n$
- $s=k^{-1}($ hash $(\mathrm{msg})+r \cdot a) \bmod n$
- $(x, y)=g^{k}$ and $r=x \bmod n$
- $s=k^{-1}($ hash $(\mathrm{msg})+r \cdot a) \bmod n$


## Verification

- receive signature ( $r, s$ ) and message msg
- compute $u=h a s h(m s g) \cdot s^{-1} \bmod n$ and $v=r s^{-1} \bmod n$
- compute $\left(x^{\prime}, y^{\prime}\right)=g^{u} \cdot A^{v}$ in the curve
- accept if $r \equiv x^{\prime} \bmod n$
- $(x, y)=g^{k}$ and $r=x \bmod n$
- $s=k^{-1}($ hash $(\mathrm{msg})+r \cdot a) \bmod n$


## Verification

- receive signature ( $r, s$ ) and message msg
- compute $u=\operatorname{hash}(\mathrm{msg}) \cdot s^{-1} \bmod n$ and $v=r s^{-1} \bmod n$
- compute $\left(x^{\prime}, y^{\prime}\right)=g^{u} \cdot A^{v}$ in the curve
- accept if $r \equiv x^{\prime} \bmod n$


## Correctness

plugging in the supposed values:

$$
\begin{aligned}
\left(x^{\prime}, y^{\prime}\right) & =g^{u} \cdot A^{v} \\
& =\left(g^{\text {hash }(\mathrm{msg})} \cdot g^{r a}\right)^{s^{-1}} \\
& =g^{k}=(x, y)
\end{aligned}
$$

## Psychic Paper

- $(x, y)=g^{k}$ and $r=x \bmod n$
- $s=k^{-1}($ hash $(\mathrm{msg})+r \cdot a) \bmod n$

Why edge cases are important

- above pseudo code is vulnerable
- some implementations say $0^{-1} \bmod n=0$
- also $\infty$ not treated correctly, but as with zero
- then signature $(0,0)$ always accepted


## Psychic Paper

- $(x, y)=g^{k}$ and $r=x \bmod n$
- $s=k^{-1}($ hash $(\mathrm{msg})+r \cdot a) \bmod n$

Why edge cases are important

- above pseudo code is vulnerable
- some implementations say $0^{-1} \bmod n=0$
- also $\infty$ 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 $r=x \bmod n$
- $s=k^{-1}($ hash $(\mathrm{msg})+r \cdot a) \bmod n$

Why edge cases are important

- above pseudo code is vulnerable
- some implementations say $0^{-1} \bmod n=0$
- also $\infty$ 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 $m, m^{\prime}$ get signatures $(r, s)$ and $\left(r, s^{\prime}\right)$

$$
\begin{aligned}
s-s^{\prime} & =k^{-1}\left(\operatorname{hash}(m)+r a-\operatorname{hash}\left(m^{\prime}\right)-r a\right) \\
\Longrightarrow k & =\frac{\operatorname{hash}(m)-\operatorname{hash}\left(m^{\prime}\right)}{s-s^{\prime}}
\end{aligned}
$$

## Sony's failure with the PS3

- fixed value $k$ (instead of random)
- for two messages $m, m^{\prime}$ get signatures $(r, s)$ and $\left(r, s^{\prime}\right)$

$$
\begin{aligned}
s-s^{\prime} & =k^{-1}\left(\operatorname{hash}(m)+r a-\operatorname{hash}\left(m^{\prime}\right)-r a\right) \\
\Longrightarrow k & =\frac{\operatorname{hash}(m)-\operatorname{hash}\left(m^{\prime}\right)}{s-s^{\prime}}
\end{aligned}
$$

- also get secret key $a=(s k-h a s h(m)) r^{-1} \bmod n$


## Sony's failure with the PS3

- fixed value $k$ (instead of random)
- for two messages $m, m^{\prime}$ get signatures $(r, s)$ and $\left(r, s^{\prime}\right)$

$$
\begin{aligned}
s-s^{\prime} & =k^{-1}\left(\operatorname{hash}(m)+r a-\operatorname{hash}\left(m^{\prime}\right)-r a\right) \\
\Longrightarrow k & =\frac{\operatorname{hash}(m)-\operatorname{hash}\left(m^{\prime}\right)}{s-s^{\prime}}
\end{aligned}
$$

- also get secret key $a=(s k-h a s h(m)) r^{-1} \bmod n$

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.

## 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.

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 $B=\left\{v_{1}, \ldots, v_{n}\right\}$, their lattice is

$$
L(B)=\operatorname{span}_{\mathbb{Z}}(B)=B \mathbb{Z}^{n}=\left\{\sum_{i=1}^{n} a_{i} v_{i}: a_{i} \in \mathbb{Z}\right\}
$$

- for simplicity $v_{i} \in \mathbb{Z}^{n}$


## Lattice

## Definition

Given a base of vectors $B=\left\{v_{1}, \ldots, v_{n}\right\}$, their lattice is

$$
L(B)=\operatorname{span}_{\mathbb{Z}}(B)=B \mathbb{Z}^{n}=\left\{\sum_{i=1}^{n} a_{i} v_{i}: a_{i} \in \mathbb{Z}\right\}
$$

- for simplicity $v_{i} \in \mathbb{Z}^{n}$


## Properties

- bases $A, B$ create same lattice if $A=U B$ for some $U \in \mathbb{Z}^{n \times n}$ with $\operatorname{det} U= \pm 1$ ( $U$ 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 $v \in L \backslash\{\mathbf{0}\}$ with $\|v\|$ minimal


## Lattice Problems

Shortest Vector Problem (SVP)

- given $L$, find a vector $v \in L \backslash\{\mathbf{0}\}$ 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 $v \in L \backslash\{\mathbf{0}\}$ 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 $B^{\prime}=\left\{v_{1}^{\prime}, \ldots, v_{n}^{\prime}\right\}$ with $B \mathbb{Z}^{n}=B^{\prime} \mathbb{Z}^{n}$ such that $\prod\left\|v_{i}^{\prime}\right\|$ minimal


## Lattice Problems

Shortest Vector Problem (SVP)

- given $L$, find a vector $v \in L \backslash\{\mathbf{0}\}$ 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 $B^{\prime}=\left\{v_{1}^{\prime}, \ldots, v_{n}^{\prime}\right\}$ with $B \mathbb{Z}^{n}=B^{\prime} \mathbb{Z}^{n}$ such that $\prod\left\|v_{i}^{\prime}\right\|$ minimal


## Hardness

- SVP NP-hard under randomised reduction
- with oracle for CVP, can solve SVP
- 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 $B x=u$ in $\mathbb{Q}$
- round entries of $x$ to get $v \in L$ via $v=B \cdot \operatorname{round}(x)$


## 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 $B x=u$ in $\mathbb{Q}$
- round entries of $x$ to get $v \in L$ via $v=B \cdot \operatorname{round}(x)$

$$
B=\left(\begin{array}{ll}
6 & 10 \\
7 & 12
\end{array}\right) \quad u=\binom{3.8}{4.1} \quad \Longrightarrow x=\binom{2.3}{-1}
$$

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 $B x=u$ in $\mathbb{Q}$
- round entries of $x$ to get $v \in L$ via $v=B \cdot \operatorname{round}(x)$

$$
B=\left(\begin{array}{ll}
6 & 10 \\
7 & 12
\end{array}\right) \quad u=\binom{3.8}{4.1} \quad \Longrightarrow x=\binom{2.3}{-1}
$$

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 $\bmod p$ ?
- 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 $\bmod p$ ?
- 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 $R=\mathbb{Z}[x] /\left(x^{n}-1\right)$,
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 $R=\mathbb{Z}[x] /\left(x^{n}-1\right)$,
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 $R=\mathbb{Z}[x] /\left(x^{n}-1\right)$,
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 modular intervals $a \bmod p$ as $a \in[-p / 2, p / 2) \cap \mathbb{Z}$
see modular intervals $a \bmod p$ as $a \in[-p / 2, p / 2) \cap \mathbb{Z}$


## 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 $\leadsto$ new $f$
- public key: $h:=p \cdot f_{q} \cdot g \bmod q$
- secret key: $f, f_{p}\left(g, f_{q}\right.$ not needed any more)
see modular intervals $a \bmod p$ as $a \in[-p / 2, p / 2) \cap \mathbb{Z}$


## 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 $\leadsto$ new $f$
- public key: $h:=p \cdot f_{q} \cdot g \bmod q$
- secret key: $f, f_{p}\left(g, f_{q}\right.$ 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 modular intervals $a \bmod p$ as $a \in[-p / 2, p / 2) \cap \mathbb{Z}$


## 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 $\leadsto$ 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

$$
\begin{aligned}
a & =f \cdot c \bmod q=f(r h+m) \bmod q=f\left(r p f_{q} g+m\right) \bmod q \\
& =p \cdot r \cdot g+f \cdot m \bmod q
\end{aligned}
$$

all polynomials of small coefficients

## Why/When NTRU works?

- In decryption

$$
\begin{aligned}
a & =f \cdot c \bmod q=f(r h+m) \bmod q=f\left(r p f_{q} g+m\right) \bmod q \\
& =p \cdot r \cdot g+f \cdot m \bmod q
\end{aligned}
$$

all polynomials of small coefficients

- want that $\bmod q$ does not do anything, then

$$
f_{p} \cdot a \bmod p=p \cdot *+f_{p} \cdot f \cdot m \bmod p=m \bmod p=m
$$

## Why/When NTRU works?

- In decryption

$$
\begin{aligned}
a & =f \cdot c \bmod q=f(r h+m) \bmod q=f\left(r p f_{q} g+m\right) \bmod q \\
& =p \cdot r \cdot g+f \cdot m \bmod q
\end{aligned}
$$

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

$$
\left|a_{i}\right| \leq p n+n=n(p+1) \stackrel{!}{<} \frac{q}{2}
$$

## Why/When NTRU works?

- In decryption

$$
\begin{aligned}
a & =f \cdot c \bmod q=f(r h+m) \bmod q=f\left(r p f_{q} g+m\right) \bmod q \\
& =p \cdot r \cdot g+f \cdot m \bmod q
\end{aligned}
$$

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

$$
\left|a_{i}\right| \leq p n+n=n(p+1) \stackrel{!}{<} \frac{q}{2}
$$

- hence $q>2 n(p+1)$ is a correct choice


## NTRU and Lattices

Calculation $\bmod x^{n}-1$ allows for special translation.
Translate polynomials into lattices

- polynomial $=$ vector of its coefficients, also as matrix

$$
v=\sum_{k=0}^{n-1} v_{k} x^{k} \cong\left(\begin{array}{c}
v_{0} \\
v_{1} \\
\vdots
\end{array}\right) \cong\left(\begin{array}{cccc}
v_{0} & v_{n-1} & \ldots & v_{1} \\
v_{1} & v_{0} & \ldots & v_{2} \\
& & \ddots & \\
v_{n-1} & v_{n-2} & \ldots & v_{0}
\end{array}\right)
$$

## NTRU and Lattices

Calculation $\bmod x^{n}-1$ allows for special translation.
Translate polynomials into lattices

- polynomial $=$ vector of its coefficients, also as matrix

$$
v=\sum_{k=0}^{n-1} v_{k} x^{k} \cong\left(\begin{array}{c}
v_{0} \\
v_{1} \\
\vdots
\end{array}\right) \cong\left(\begin{array}{cccc}
v_{0} & v_{n-1} & \ldots & v_{1} \\
v_{1} & v_{0} & \ldots & v_{2} \\
& & \ddots & \\
v_{n-1} & v_{n-2} & \ldots & v_{0}
\end{array}\right)
$$

- adding polynomials $=$ adding vectors $=$ adding matrices


## NTRU and Lattices

Calculation $\bmod x^{n}-1$ allows for special translation.
Translate polynomials into lattices

- polynomial $=$ vector of its coefficients, also as matrix

$$
v=\sum_{k=0}^{n-1} v_{k} x^{k} \cong\left(\begin{array}{c}
v_{0} \\
v_{1} \\
\vdots
\end{array}\right) \cong\left(\begin{array}{cccc}
v_{0} & v_{n-1} & \ldots & v_{1} \\
v_{1} & v_{0} & \ldots & v_{2} \\
& & \ddots & \\
v_{n-1} & v_{n-2} & \ldots & v_{0}
\end{array}\right)
$$

- adding polynomials $=$ adding vectors $=$ adding matrices
- multiplication of polynomials $f, g$ :

$$
\operatorname{Matrix}(f) \cdot \operatorname{Vector}(g)=\operatorname{Vector}(f \cdot g)
$$

Hence, we are in the realm of lattices.

## NTRU and Lattice Problems

## Break Key

$f, g$ only have small entries

$$
(f, g) \in \mathcal{L}\left(\left(\begin{array}{cc}
I_{n} & 0 \\
p^{-1} h & q I_{n}
\end{array}\right)\right) \subseteq \mathbb{Z}^{2 n}
$$

$f, g \in\{-1,0,1\}^{n}$, so we look for short vectors $\sim$ SVP

## NTRU and Lattice Problems

## Break Key

$f, g$ only have small entries

$$
(f, g) \in \mathcal{L}\left(\left(\begin{array}{cc}
I_{n} & 0 \\
p^{-1} h & q I_{n}
\end{array}\right)\right) \subseteq \mathbb{Z}^{2 n}
$$

$f, g \in\{-1,0,1\}^{n}$, so we look for short vectors $\sim$ SVP

## Find Message

$r, m$ only have small entries

$$
(r, c-m) \in \mathcal{L}\left(\left(\begin{array}{cc}
I_{n} & 0 \\
h & q I_{n}
\end{array}\right)\right) \subseteq \mathbb{Z}^{2 n}
$$

$r, m \in\{-1,0,1\}^{n}$, so we look for a vector close to $(0, c) \sim$ 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

$$
\left|a_{i}\right| \leq p \cdot(q / 8-2)+q / 8-2=q / 2-8<\frac{q}{2}
$$

## 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

$$
\left|a_{i}\right| \leq p \cdot(q / 8-2)+q / 8-2=q / 2-8<\frac{q}{2}
$$

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 $\sim$ less researched


## Multivariate Cryptography

Problem MQ - Multivariate Quadratic
Given: finite field $K$, polynomials $f_{i} \in K\left[x_{1}, \ldots, x_{n}\right]$ 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 $f_{i} \in K\left[x_{1}, \ldots, x_{n}\right]$ 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 $x \vee y=x+y-x y$ and auxiliary variables $\Longrightarrow$ NP-hard


## Multivariate Cryptography

## Problem MQ - Multivariate Quadratic

Given: finite field $K$, polynomials $f_{i} \in K\left[x_{1}, \ldots, x_{n}\right]$ 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 $x \vee y=x+y-x y$ and auxiliary variables $\Longrightarrow$ NP-hard


## Example

take formula $\varphi=\left(x_{1} \wedge x_{2} \wedge \neg x_{3}\right) \vee\left(\neg x_{2} \wedge x_{3}\right)$
to find satisfying assignment, solve system

$$
\begin{aligned}
& y_{1}=x_{1} \cdot x_{2} \\
& y_{2}=y_{1} \cdot\left(1-x_{3}\right) \\
& y_{3}=\left(1-x_{2}\right) \cdot x_{3} \\
& 1=y_{2}+y_{3}-y_{2} \cdot y_{3}
\end{aligned}
$$

## 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: $y_{i} \in K, f_{i} \in K\left[x_{1}, \ldots, x_{n}\right]$
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: $y_{i} \in K, f_{i} \in K\left[x_{1}, \ldots, x_{n}\right]$
Task: find $\boldsymbol{x} \in K^{n}$ with $y_{i}=f_{i}(\boldsymbol{x})$ for all $i$
translate into cryptography
Keys: $P=\left(f_{1}, \ldots, f_{m}\right)$ - 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 $x \rightarrow T \rightarrow F \rightarrow S \sim P(x)$ )
- 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 $x \rightarrow T \rightarrow F \rightarrow S \sim P(x)$ )
- 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 $x \rightarrow T \rightarrow F \rightarrow S \sim P(x)$ )
- 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

$$
y_{i}=\sum_{j, k} \gamma_{i j k} x_{j} a_{k}+\sum_{j, k} \lambda_{i j k} a_{j} a_{k}+\sum_{j} \xi_{i j} x_{j}+\sum_{j} \xi_{i j}^{\prime} a_{j}+\delta_{i}
$$

## 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

$$
y_{i}=\sum_{j, k} \gamma_{i j k} x_{j} a_{k}+\sum_{j, k} \lambda_{i j k} a_{j} a_{k}+\sum_{j} \xi_{i j} x_{j}+\sum_{j} \xi_{i j}^{\prime} a_{j}+\delta_{i}
$$

- fix random values for vinegar $a_{j}$
- solve linear equation system to get $x_{j}$
- yields preimage $(\boldsymbol{x}, \boldsymbol{a})$ for $\boldsymbol{y}=\left(y_{1}, \ldots, y_{n}\right)$


## 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 $v \geq 2 n$
- parameters $\gamma, \lambda, \xi, \xi^{\prime}, \delta$ for $F$ randomly
- $S, T \in K^{m} \rightarrow K^{m}$ 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 $v \geq 2 n$
- parameters $\gamma, \lambda, \xi, \xi^{\prime}, \delta$ for $F$ randomly
- $S, T \in K^{m} \rightarrow K^{m}$ affine, random, chance to be invertible

$$
\frac{\left|\mathrm{GL}_{m}(q)\right|}{q^{m \cdot m}}=\prod_{k=0}^{m-1}\left(1-q^{k-m}\right) \xrightarrow{m \rightarrow \infty} 28.8 \ldots \% \quad \text { for } q=2
$$

- similar chance 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 $v \geq 2 n$
- parameters $\gamma, \lambda, \xi, \xi^{\prime}, \delta$ for $F$ randomly
- $S, T \in K^{m} \rightarrow K^{m}$ affine, random, chance to be invertible

$$
\frac{\left|\mathrm{GL}_{m}(q)\right|}{q^{m \cdot m}}=\prod_{k=0}^{m-1}\left(1-q^{k-m}\right) \xrightarrow{m \rightarrow \infty} 28.8 \ldots \% \quad \text { for } q=2
$$

- similar chance for computing $x_{j}$ for random $a_{j}$
- problem: key size $\mathcal{O}\left(m^{3} \log q\right)$


## 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 $\times 100$ )


## 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 $\times 100$ )
recently broken


## Further Post-Quantum Candidates

- open competition by NIST
- 3rd round just finished


## Further Post-Quantum Candidates

- open competition by NIST
- 3rd round just finished
- key exchange (KEM: key encapsulation method)
- lattice: NTRU, Kyber, Saber
- code: Classic McEliece


## Further Post-Quantum Candidates

- open competition by NIST
- 3rd round just finished
- 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...

# Happy Hacking! 

I hope you had fun.<br>Maybe see you at some CTF ;)

