Cryptography 101: From Theory to Practice

Chapter 5 – One-Way Functions

Rolf Oppliger

March 16, 2022

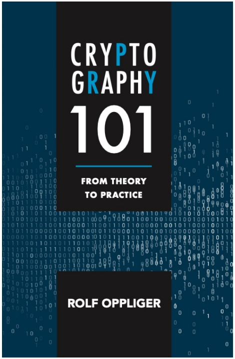
Terms of Use

whoami

rolf-oppliger.ch
rolf-oppliger.com

- Swiss National Cyber Security Centre NCSC (scientific employee)
- eSECURITY Technologies Rolf Oppliger (founder and owner)
- University of Zurich (adjunct professor)
- Artech House (author and series editor for information security and privacy)

Reference Book



© Artech House, 2021
ISBN 978-1-63081-846-3

https://books.esecurity.ch/crypto101.html

Challenge Me

Outline

5. One-Way Functions

5. One-Way Functions

5. One-Way Functions

5.1 Introduction

- According to Definition 2.3, a function $f : X \rightarrow Y$ is one way, if $f(x)$ can be computed efficiently for all $x \in X$, but $f^{-1}(f(x))$ cannot be computed efficiently for all $x \in X$, i.e., $f^{-1}(y)$ cannot be computed efficiently for $y \in_R Y$
- In a complexity-theoretic setting, an "efficient computation" stands for a computation that runs in polynomial time
- A probabilistic algorithm that runs in polynomial time is called **probabilistic polynomial-time (PPT)**

5. One-Way Functions

5.1 Introduction

Definition 5.1 (One-way function)

A function $f : X \rightarrow Y$ for which the following two conditions are fulfilled:

- The function f is easy to compute, meaning that it is known how to efficiently compute $f(x)$ for all $x \in X$ (i.e., there is a PPT algorithm A that outputs $A(x) = f(x)$ for all $x \in X$)
- The function f is hard to invert, meaning that it is not known how to efficiently compute $f^{-1}(f(x))$ for $x \in_R X$ (i.e., there is no known PPT algorithm A that can output $A(f(x)) = f^{-1}(f(x))$ for $x \in_R X$)

5. One-Way Functions

5.1 Introduction

- Another way to express the second condition is to say that any PPT algorithm A that tries to invert f only succeeds with a probability that is negligible (i.e., bound by a polynomial fraction)
- This means that there is a positive integer $n_0 \in \mathbb{N}$, such that for every PPT algorithm A, every $x \in X$, every polynomial $p(\cdot)$, and all $n_0 \leq n \in \mathbb{N}$ the following relation holds:

$$\Pr[A(f(x), 1^b) \in f^{-1}(f(x))] \leq \frac{1}{p(n)}$$

5. One-Way Functions

5.1 Introduction

- The following (equivalent) notation is also used in the literature:

$$\Pr[(f(z) = y : x \xleftarrow{r} \{0,1\}^b; y \leftarrow f(x); z \leftarrow A(y, 1^b)] \leq \frac{1}{p(n)}$$

- If x is sampled uniformly at random from $\{0,1\}^b$, y is assigned $f(x)$, and z is assigned $A(y, 1^b)$, then the probability that $f(z)$ equals $y = f(x)$ is negligible

5. One-Way Functions

5.1 Introduction

- According to Definition 2.4, a one-way function $f : X \rightarrow Y$ is a **trapdoor (one-way) function**, if there is some extra information with which f can be inverted efficiently

Definition 5.2 (Trapdoor function)

A one-way function $f : X \rightarrow Y$ for which there is a trapdoor information t and a PPT algorithm I that can be used to efficiently compute $x' = I(f(x), t)$ with $f(x') = f(x)$

5. One-Way Functions

5.1 Introduction

- One-way permutations and trapdoor (one-way) permutations are defined similarly
- Instead of talking about one-way functions, trapdoor functions, one-way permutations, and trapdoor permutations, one often refers to such families
- This is because many cryptographic functions required to be one way output bit strings of fixed length, and hence finding a preimage requires a huge but fixed number of tries (e.g., 2^n)
- In complexity theory, the computational complexity to invert such a function is $O(1)$ and hence trivial

5. One-Way Functions

5.1 Introduction

- If one wants to use complexity-theoretic arguments, then one cannot have a constant n
- Instead, one must make n variable, and it must be possible to let n grow arbitrarily large
- Consequently, one has to work with a potentially infinite family of functions, and there must be at least one function for every possible value of n
- Alternative terms for families are "classes," "collections," or "ensembles"
- This mathematical precision is not always enforced

5. One-Way Functions

5.1 Introduction

Definition 5.3 (Family of one-way functions)

A family of functions $F = \{f_i : X_i \rightarrow Y_i\}_{i \in I}$ that fulfills the following two conditions:

- I is an infinite index set
- For every $i \in I$ there is a function $f_i : X_i \rightarrow Y_i$ that is one-way

The notion of a family similarly applies to trapdoor functions, one-way permutations, and trapdoor permutations

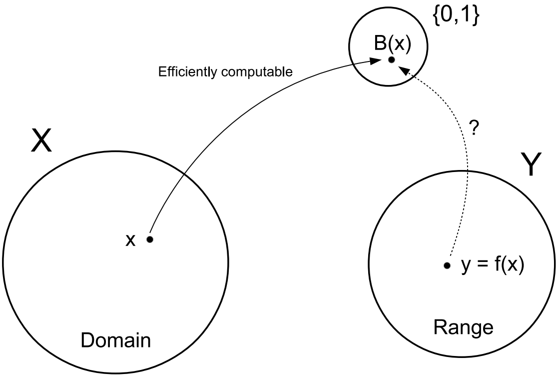
5. One-Way Functions

5.1 Introduction

- The notion of a one-way function suggests that x cannot be computed efficiently from $f(x)$
- This does not exclude the case that some partial information about x can be determined
- Every one-way function f is known to have a **hard-core predicate**, i.e., a predicate $B : X \rightarrow \{0,1\}$ that can be computed efficiently from x but not from $f(x)$
- Hard-core predicates are heavily used, for example, in cryptographically secure PRGs

5. One-Way Functions

5.1 Introduction



5. One-Way Functions

5.1 Introduction

Definition 5.4 (Hard-core predicate)

If $f : X \rightarrow Y$ is a one-way function, then a hard-core predicate of f is a predicate $B : X \rightarrow \{0, 1\}$ that fulfills the following two conditions:

- $B(x)$ can be computed efficiently for all $x \in X$, i.e., there is a PPT algorithm A that can output $B(x)$ for all $x \in X$
- $B(x)$ cannot be computed efficiently from $y = f(x) \in Y$ for $x \in_R X$, i.e., there is no known PPT algorithm A that can output $B(x)$ from $y = f(x)$ for $x \in_R X$

5. One-Way Functions

5.2 Candidate One-Way Functions

- Mathematically speaking, there is no function known to be one way (otherwise $\mathbf{NP} \neq \mathbf{P}$ would also be true)
- There are only a few functions conjectured to be one way
- Most of these functions are centered around modular exponentiation (for some properly chosen modulus m)
  - Discrete exponentiation function: $f(x) = g^x \bmod m$
  - RSA function: $f(x) = x^e \bmod m$
  - Modular square function: $f(x) = x^2 \bmod m$

5. One-Way Functions

5.2 Candidate One-Way Functions – Discrete Exponentiation Function

- In $\mathbb{R}$, the exponentiation function maps arbitrary elements $x \in \mathbb{R}$ to $y = \exp(x) = e^x \in \mathbb{R}$, whereas the logarithm function does the opposite i.e., it maps x to $\ln(x)$
- This is true for base e, but it is also true for any other base $a \in \mathbb{R}$
- Formally, the two functions can be expressed as follows:

$$\begin{aligned} \mathrm{Exp}: \ \mathbb{R} &\longrightarrow \mathbb{R} \\ x &\longmapsto a^x \end{aligned} \qquad\qquad \begin{aligned} \mathrm{Log}: \ \mathbb{R} &\longrightarrow \mathbb{R} \\ x &\longmapsto \log_a x \end{aligned}$$

5. One-Way Functions

5.2 Candidate One-Way Functions – Discrete Exponentiation Function

- In $\mathbb{R}$, both the exponentiation function and the logarithm function are continuous and can be computed efficiently, using any form of approximation
- But in a discrete algebraic structure, it is usually not possible to use the notion of continuity and approximate a solution
- In fact, there are cyclic groups in which the exponentiation function (i.e., discrete exponentiation function) can be computed efficiently, whereas the inverse function (i.e., discrete logarithm function) cannot be computed efficiently

5. One-Way Functions

5.2 Candidate One-Way Functions – Discrete Exponentiation Function

- If G is such a (multiplicatively written) group with generator g, then one can express the discrete exponentiation and logarithm functions as follows:

$$\begin{aligned} \mathrm{Exp}: \ \mathbb{N} &\longrightarrow G \\ x &\longmapsto g^x \end{aligned} \qquad\qquad \begin{aligned} \mathrm{Log}: \ G &\longrightarrow \mathbb{N} \\ x &\longmapsto \log_g x \end{aligned}$$

- Depending on the nature of G, no efficient algorithm may be known to compute Log
- Examples are $\langle \mathbb{Z}_p^*, \cdot \rangle$, denoted $\mathbb{Z}_p^*$, or — more realistically — a subgroup of $\mathbb{Z}_p^*$ with $q = (p-1)/2$ elements

5. One-Way Functions

5.2 Candidate One-Way Functions – Discrete Exponentiation Function

- Construction of such a family of one-way functions
  - Index set $I := \{(p,g) \mid p \in \mathbb{P};\ g \text{ generates } G = \mathbb{Z}_p^*\}$
  - Ideally, $I := \{(p,g) \mid p \in \mathbb{P}^*;\ g \text{ generates } G \subset \mathbb{Z}_p^* \text{ with } |G| = q = (p-1)/2\}$ where $\mathbb{P}^*$ refers to the set of all safe primes
  - Family of discrete exponentiation functions

$$\textbf{Exp} := \{\mathrm{Exp} : \mathbb{N} \longrightarrow G,\ x \longmapsto g^x\}_{(p,g)\in I}$$

  - Family of discrete logarithm functions

$$\textbf{Log} := \{\mathrm{Log} : G \longrightarrow \mathbb{N},\ x \longmapsto \log_g x\}_{(p,g)\in I}$$

5. One-Way Functions

5.2 Candidate One-Way Functions – Discrete Exponentiation Function

- If one wants to use **Exp** as a family of one-way functions, then one has to be sure that discrete logarithms cannot be computed efficiently in G
- This is where the **discrete logarithm assumption (DLA)** comes into play
- It suggests that a PPT algorithm A to compute a discrete logarithm can only succeed with a probability that is negligible
- This is (one of the reasons) why p should be a safe prime

5. One-Way Functions

5.2 Candidate One-Way Functions – Discrete Exponentiation Function

- There are several problems phrased around the DLA and the one-way property of the discrete exponential function
  - Discrete logarithm problem (DLP)
  - (Computational) Diffie-Hellman problem (DHP)
  - Decisional Diffie-Hellman problem (DDHP)
- In the definitions, the problems are specified in abstract notation using a cyclic group G and a generator g
- The numerical examples are given in $\mathbb{Z}_7^* = \{1,2,3,4,5,6\}$ with generator $g = 5$ (note that $g = 5$ generates all elements of $\mathbb{Z}_7^*$; i.e., $5^0 = 1, 5^1 = 5, 5^2 = 4, 5^3 = 6, 5^4 = 2$, and $5^5 = 3$)

5. One-Way Functions

5.2 Candidate One-Way Functions – Discrete Exponentiation Function

Definition 5.5 (DLP)

If G is a cyclic group with generator g, then the DLP is to determine $x \in \mathbb{N}$ for g^x

- In $\mathbb{Z}_7^*$ with $g = 5$, the DLP for $g^x = 4$ yields $x = 2$, because $5^2 \bmod 7 = 4$
- The group is so small that all possible values of x can simply be tried out (this doesn't work in large groups)
- The discrete (and cyclic) nature of G makes it impossible to solve the DLP by approximation

5. One-Way Functions

5.2 Candidate One-Way Functions – Discrete Exponentiation Function

Definition 5.6 (DHP)

If G is a cyclic group, g a generator of G, and x and y two positive integers smaller than the order of G, i.e., $0 < x, y < |G|$, then the DHP is to determine g^{xy} for g^x and g^y

- In $\mathbb{Z}_7^*$ with $g = 5$, $x = 3$ and $y = 6$ yield $g^x = 5^3 \bmod 7 = 6$ and $g^y = 5^6 \bmod 7 = 1$
- The DHP is to determine $g^{xy} = 5^{18} \bmod 7 = 1$ from $g^x = 6$ and $g^y = 1$
- The DHP is at the core of the Diffie-Hellman key exchange

5. One-Way Functions

5.2 Candidate One-Way Functions – Discrete Exponentiation Function

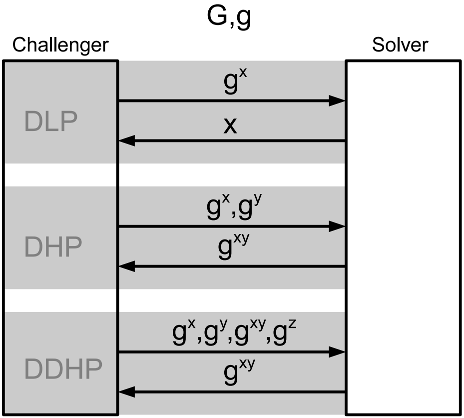
Definition 5.7 (DDHP)

If G is a cyclic group, g a generator of G, and x, y, and z three positive integers smaller than the order of G, i.e., $0 < x, y, z < |G|$, then the DDHP is to decide whether g^{xy} or g^z solves the DHP for g^x and g^y

- In $\mathbb{Z}_7^*$ with $g = 5$, $x = 3$, $y = 6$, and $z = 2$ yield $g^x = 5^3 \bmod 7 = 6$, $g^y = 5^6 \bmod 7 = 1$, and $g^z = 5^2 \bmod 7 = 4$
- The DDHP is to determine whether $g^{xy} = 1$ or $g^z = 4$ solves the DHP (see above)

5. One-Way Functions

5.2 Candidate One-Way Functions – Discrete Exponentiation Function



5. One-Way Functions

5.2 Candidate One-Way Functions – Discrete Exponentiation Function

- An interesting question is how the DLA-based problems, i.e., DLP, DHP, and DDHP, relate to each other
- This question is answered by giving complexity-theoretic reductions: DDHP $\leq_P$ DHP $\leq_P$ DLP
- In many groups, the DLP and the DHP are computationally equivalent
- There are groups in which the DDHP can be solved in polynomial time, whereas the fastest known algorithms to solve the DHP still require subexponential time (e.g., gap Diffie-Hellman groups)

5. One-Way Functions

5.2 Candidate One-Way Functions – RSA function

- The RSA function refers to $f(x) = x^e \bmod m$, where m is a composite integer — usually written as n
- More specifically, n is the product of two distinct primes p and q, i.e., $n = pq$, and e is relatively prime to $\phi(n)$ — where $\phi(n)$ refers to Euler's totient function
- The RSA function can be defined as follows:

$$\begin{aligned} \mathrm{RSA}_{n,e} : \mathbb{Z}_n &\longrightarrow \mathbb{Z}_n \\ x &\longmapsto x^e \end{aligned}$$

- It operates on $\mathbb{Z}_n$ and computes the e-th power of $x \in \mathbb{Z}_n$

5. One-Way Functions

5.2 Candidate One-Way Functions – RSA function

- $\mathrm{RSA}_{n,e}$ yields a permutation on the elements of $\mathbb{Z}_n$, i.e., $\mathrm{RSA}_{n,e} \in \mathrm{Perms}[\mathbb{Z}_n]$
- To compute the inverse function (i.e., e-th roots), one must know the multiplicative inverse element d of e modulo $\phi(n)$
- Using d, the inverse function of $\mathrm{RSA}_{n,e}$ is defined as follows:

$$\begin{aligned} \mathrm{RSA}_{n,d} : \mathbb{Z}_n &\longrightarrow \mathbb{Z}_n \\ x &\longmapsto x^d \end{aligned}$$

- $\mathrm{RSA}_{n,e}$ and $\mathrm{RSA}_{n,d}$ can be computed efficiently

5. One-Way Functions

5.2 Candidate One-Way Functions – RSA function

- To compute $\mathrm{RSA}_{n,d}$, one must know either d, one prime factor of n, i.e., p or q, or $\phi(n)$
- Any of these values yields a trapdoor
- No polynomial-time algorithm is known to compute any of these values from n and e
- The quantum computer is a game changer (using Shor's algorithm)
- But nobody has been able to build a sufficiently large quantum computer yet (in terms of qubits)

5. One-Way Functions

5.2 Candidate One-Way Functions – RSA function

- Construction of a family of one-way functions
  - Index set $I := \{(n, e) \mid n = pq; p, q \in \mathbb{P}; p \neq q; 1 < e < \phi(n); (e, \phi(n)) = 1\}$
  - Family of RSA functions

$$\textbf{RSA} := \{\mathrm{RSA}_{n,e} : \mathbb{Z}_n \longrightarrow \mathbb{Z}_n, \ x \longmapsto x^e\}_{(n,e)\in I}$$

  - The family comprises both $\mathrm{RSA}_{n,e}$ and $\mathrm{RSA}_{n,d}$
- Because every RSA function $\mathrm{RSA}_{n,e}$ has trapdoors and yields a permutation over $\mathbb{Z}_n$, **RSA** is a family of trapdoor permutations

5. One-Way Functions

5.2 Candidate One-Way Functions – RSA function

- It is assumed that $\mathrm{RSA}_{n,e}$ is hard to invert (for a sufficiently large n and without knowing a trapdoor)
- More specifically, the **RSA assumption** suggests that any PPT algorithm can invert $\mathrm{RSA}_{n,e}$ only with a success probability that is negligible
- There is even a stronger version of the RSA assumption known as **strong RSA assumption**
- It suggests that the success probability for a PPT algorithm remains negligible even if it can select the value of e

5. One-Way Functions

5.2 Candidate One-Way Functions – RSA function

- An obvious way to invert $\mathrm{RSA}_{n,e}$ is to determine a trapdoor, e.g., by solving the **integer factoring problem (IFP)**

Definition 5.8 (IFP)

For $n \in \mathbb{N}$, the IFP is to determine the distinct values $p_1, \ldots, p_k \in \mathbb{P}$ and $e_1, \ldots, e_k \in \mathbb{N}$ such that $n = p_1^{e_1} \cdots p_k^{e_k}$

- The **integer factoring assumption (IFA)** suggests that the IFP cannot be solved efficiently, meaning that any PPT algorithm can solve the IFP only with a success probability that is negligible

5. One-Way Functions

5.2 Candidate One-Way Functions – RSA function

- Under the RSA and IFA assumptions, the **RSA problem (RSAP)** is computationally intractable

Definition 5.9 (RSAP)

If (n, e) is a public key with $n = pq$ and $c \equiv m^e \pmod{n}$ a ciphertext, then the RSAP is to determine m, i.e., computing the e^{th} root of c modulo n (without trapdoor)

- It is obvious that RSAP $\leq_P$ IFP
- The converse, i.e., IFP $\leq_P$ RSAP, is not known to be true
- RSAP and IFP are not computationally equivalent

5. One-Way Functions

5.2 Candidate One-Way Functions – RSA function

- According to the strong RSA assumption, the value of e may be considered as an additional parameter
- The respective problem is called the **flexible RSAP**: For given n and c, find e and m such that $c \equiv m^e \pmod{n}$
- Clearly, flexible RSAP $\leq_P$ RSAP
- This can easily be shown by fixing an arbitrary value for e and solving the respective RSAP

5. One-Way Functions

5.2 Candidate One-Way Functions – Modular square function

- Starting with the "normal" RSA function in $\mathbb{Z}_n$, one may replace e with the value 2 (that is invalid for the "normal" RSA function)
- This yields the **modular square function**:

$$\begin{array}{rcccl} \mathrm{Square}_n & : & \mathbb{Z}_n & \longrightarrow & QR_n \\ & & x & \longmapsto & x^2 \end{array}$$

- 2 is not relatively prime to $\phi(n)$, and hence Square_n is not bijective and does not yield a permutation over $\mathbb{Z}_n$
- The range of the modular square function is QR_n

5. One-Way Functions

5.2 Candidate One-Way Functions – Modular square function

- QR_n is a proper subgroup of $\mathbb{Z}_n$, i.e., $QR_n \subset \mathbb{Z}_n$
- There are values $x_1, x_2, \ldots$ in $\mathbb{Z}_n$ that are mapped to the same value x^2 in QR_n, and hence Square_n is not injective
- This suggests that the inverse **modular square root function**

$$\begin{array}{rcrcl} \mathrm{Sqrt}_n & : & QR_n & \longrightarrow & \mathbb{Z}_n \\ & & x & \longmapsto & x^{1/2} \end{array}$$

  is not properly defined
- To properly define it, one has to make sure that Square_n is injectice (or bijective, respectively)

5. One-Way Functions

5.2 Candidate One-Way Functions – Modular square function

- This can be achieved by restricting the domain and codomain to QR_n (where n is usually a Blum integer)
- In this case, Square_n is bijective and yields a permutation over QR_n, and hence Sqrt_n always has a solution.
- More specifically, every $x \in QR_n$ has four square roots modulo n, of which one is again an element of QR_n
- This unique square root of x is called the **principal square root** of x modulo n

5. One-Way Functions

5.2 Candidate One-Way Functions – Modular square function

- Construction of a family of one-way permutations
  - $I := \{n \mid n = pq; p, q \in \mathbb{P}; p \neq q; |p| = |q|; p, q \equiv 3 \pmod 4\}$
  - Family of modular square functions

$$\textbf{Square} := \{\mathrm{Square}_n \;:\; QR_n \longrightarrow QR_n, \;\; x \longmapsto x^2\}_{n \in I}$$

  - Family of inverse functions

$$\textbf{Sqrt} := \{\mathrm{Sqrt}_n \;:\; QR_n \longrightarrow QR_n, \; x \longmapsto x^{1/2}\}_{n \in I}$$

5. One-Way Functions

5.2 Candidate One-Way Functions – Modular square function

- In the case of the "normal" RSA function, the problems of computing e-th roots in $\mathbb{Z}_n$ and factoring n are not known to be computationally equivalent
- In contrast, modular squares can always be computed efficiently, whereas modular square roots (if they exist) can be computed efficiently iff the prime factorization of n is known
- This suggests that the problems of computing square roots in QR_n and factoring n are computationally equivalent

5. One-Way Functions

5.3 Integer Factorization Algorithms

- The IFP has attracted many mathematicians in the past
- There are several integer factorization algorithms to choose from
- Some of these algorithms are **special-purpose**, whereas others are **general-purpose**
- In practice, algorithms of both categories are routinely combined and used one after another

5. One-Way Functions

5.3 Integer Factorization Algorithms – Special-Purpose Algorithms

- Trial division
- $P-1$ algorithm (John M. Pollard, 1970s)
- $P+1$ algorithm (Hugh C. Williams, 1980s)
- Elliptic curve method (Hendrik W. Lenstra, late 1980s)
- Pollard Rho (John M. Pollard, 1975)

5. One-Way Functions

5.3 Integer Factorization Algorithms – General-Purpose Algorithms

- General-purpose integer factorization algorithms work equally well for all n
- Most of these algorithms exploit an idea of Fermat
- It starts from the fact that every odd integer $n \geq 3$ can be written as the difference of two squares, i.e., $n = x^2 - y^2$, for $x, y \in \mathbb{N}$ (where y may also be zero)
- According to the third binomial formula, $x^2 - y^2$ is equal to $(x+y)(x-y)$, and this suggests that $p = (x+y)$ and $q = (x-y)$ are factors of n (if n is prime, then the factors are trivial, i.e., n and 1)

5. One-Way Functions

5.3 Integer Factorization Algorithms – General-Purpose Algorithms

- For example, to factorize $n = 91$ one has to find two integers for which the difference of the squares is equal to this value
- In this example, $x = 10^2 = 100$ and $y = 3^2 = 9$ satisfy this property, and hence $p = 10 + 3 = 13$ and $q = 10 - 3 = 7$ yield the two (prime) factors of 91 (i.e., $13 \cdot 7 = 91$)
- Fermat also proposed a method to find a valid (x, y)-pair
- But the method is efficient only if x and y are similarly sized and not too far away from $\sqrt{n}$
- Otherwise, the method is not efficient and largely impractical

5. One-Way Functions

5.3 Integer Factorization Algorithms – General-Purpose Algorithms

- There are several algorithms that can be used to find such (x, y)-pairs (instead of Fermat's method)
  - Continued fraction
  - Sieving methods
    - Quadratic sieve (QS)
    - Number field sieve (NFS)
    - Special number field sieve (SNFS)
    - General number field sieve (GNFS)
- The NFS algorithm (and its variants) consists of two steps, of which one can be parallelized and optimized with special hardware (e.g., TWINKLE, SHARK, YASD, ...)

5. One-Way Functions

5.3 Integer Factorization Algorithms

- A USD 100 factorization challenge (RSA-129) was posted in the August 1977 issue of the *Scientific American*
- In 1994, it was solved with a distributed version of the QS

$$
\begin{aligned}
\text{RSA-129} &= 114381625757888867669235779976146612010218296721242 \\
&\quad 362562561842935706935245733897830597123563958705 \\
&\quad 058989075147599290026879543541 \\
&= 3490529510847650949147849619903898133417764638493 \\
&\quad 387843990820577 \\
&\quad * \\
&\quad 32769132993266709549961988190834461413177642967992 \\
&\quad 942539798288533
\end{aligned}
$$

5. One-Way Functions

5.3 Integer Factorization Algorithms

- RSA Factoring Challenge (officially running until 2007)
  - RSA-576 (2003, USD 10,000)
  - RSA-640 (2005, USD 20,000)
  - RSA-704 (2012)
  - RSA-768 (2009)
  - RSA-240 (795-bit number, December 2019)
  - RSA-250 (829-bit number, February 2020)
  - ...
- The bottom line is that the current state of the art in factorizing large integers is still below 1,024 bits
- Longer keys ($\geq$ 2,048 bits) are recommended

5. One-Way Functions

5.4 Algorithms for Computing Discrete Logarithms

- Several public key cryptosystems are based on the computational intractability of the DLP in a cyclic group
- If somebody were able to solve the DLP and efficiently compute discrete logarithms, then he or she would be able to break these systems
- It is therefore important to know the most efficient algorithms that can be used to compute discrete logarithms
- Again, there are generic and nongeneric (special-purpose) algorithms

5. One-Way Functions

5.4 Algorithms for Computing Discrete Logarithms

- There are a few generic algorithms that can be used to solve the DLP in a cyclic group G
- $O(\sqrt{|G|})$ is a lower bound for the time complexity of such an algorithm
- Improvements are only possible if the prime factorization of $|G|$ is known
- In this case (and if the prime factors of $|G|$ are sufficiently small), the Pohlig-Hellman algorithm can be used to efficiently solve the DLP

5. One-Way Functions

5.4 Algorithms for Computing Discrete Logarithms

- Generic algorithms
  - Brute-Force Search
  - Baby-Step Giant-Step Algorithm (Daniel Shanks, 1971)
  - Pollard Rho (John M. Pollard, 1978)
- Nongeneric (special-purpose) algorithms
  - Index calculus method (ICM) for $\mathbb{Z}_p^*$ and some other groups
  - NFS

5. One-Way Functions

5.5 Elliptic Curve Cryptography

- Public key cryptosystems get their security from the assumed intractability of inverting a one-way function
- This is not equally difficult in all algebraic structures
- For example, there are nongeneric (special-purpose) algorithms with subexponential running times (e.g., ICM, NFS, ...) to invert the discrete exponentiation function (and solve the DLP) in $\mathbb{Z}_p^*$
- These algorithms do not work in all groups

5. One-Way Functions

5.5 Elliptic Curve Cryptography

- This is where **elliptic curve cryptography (ECC)** comes into play
- In a group of points on an elliptic curve over a finite field no nongeneric (special-purpose) algorithm to solve the DLP (ECDLP) is known to exist
- This does not mean that such an algorithm does not exist (it is just not known)
- The bottom line is that one can work with shorter keys (and still achieve the same level of security)

5. One-Way Functions

5.5 Elliptic Curve Cryptography

- ECC employs groups of points on an elliptic curve over a finite field $\mathbb{F}_q$, where q is an odd prime (prime field) or some power of a prime (extension field)
- In the second case, the prime 2 is most frequently used (i.e., binary extension field of characteristic 2)
- If $q = 2^m$ for some $m \in \mathbb{N}$, then m is the degree of the (binary extension) field
- Prime fields are mainly used in software implementations, whereas binary extension fields are mainly used in hardware implementations

5. One-Way Functions

5.5 Elliptic Curve Cryptography

- If p is an odd prime, then the Weierstrass equation

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

with $a, b \in \mathbb{Z}_p$ and $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$ yields an elliptic curve over $\mathbb{Z}_p$:

$$\begin{aligned} E(\mathbb{Z}_p) = \{(x, y) \quad | \quad & x, y \in \mathbb{Z}_p \ \wedge \\ & y^2 \equiv x^3 + ax + b \pmod{p} \ \wedge \\ & 4a^3 + 27b^2 \not\equiv 0 \pmod{p}\} \end{aligned}$$

5. One-Way Functions

5.5 Elliptic Curve Cryptography

- $E(\mathbb{Z}_p)$ comprises all $(x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p = \mathbb{Z}_p^2$ that solve to the Weierstrass equation
- One can graphically interpret (x, y) as a point in the (x, y)-plane
- In addition to the points on the curve, one also considers a point at infinity, denoted $\mathcal{O}$
- This point yields the identity element required for the group operation
- If one uses $E(\mathbb{Z}_p)$ to refer to an elliptic curve defined over $\mathbb{Z}_p$, then it implicitly also includes $\mathcal{O}$

5. One-Way Functions

5.5 Elliptic Curve Cryptography

- For $p = 23$ and $a = b = 1$ $[4 \cdot 1^3 + 27 \cdot 1^2 \not\equiv 0 \pmod{23}]$, the elliptic curve $y^2 \equiv x^3 + x + 1$ is defined over $\mathbb{Z}_{23}$
- Besides $\mathcal{O}$, $E(\mathbb{Z}_{23})$ comprises the following 27 elements:

$$\begin{array}{ccccccc}
(0,1) & (0,22) & (1,7) & (1,16) & (3,10) & (3,13) & (4,0) \\
(5,4) & (5,19) & (6,4) & (6,19) & (7,11) & (7,12) & (9,7) \\
(9,16) & (11,3) & (11,20) & (12,4) & (12,19) & (13,7) & (13,16) \\
(17,3) & (17,20) & (18,3) & (18,20) & (19,5) & (19,18) &
\end{array}$$

- This sums up to 28 elements of $E(\mathbb{Z}_{23})$
- Animation to visualize the group elements

5. One-Way Functions

5.5 Elliptic Curve Cryptography

- If n is the number of points on an elliptic curve over a finite field $\mathbb{F}_q$, then n is of the order of q
- A theorem due to Helmut Hasse bounds n as

$$q + 1 - 2\sqrt{q} \leq n \leq q + 1 + 2\sqrt{q}$$

- In the previous example, the Hasse theorem suggests that $E(\mathbb{Z}_{23})$ has between $23 + 1 - 2\sqrt{23} = 14.4\ldots$ and $23 + 1 + 2\sqrt{23} = 35.5\ldots$ elements (28 is in this range)

5. One-Way Functions

5.5 Elliptic Curve Cryptography

- In addition to a set of elements, a group must also have an associative operation
- In ECC, this operation is called addition (mainly for historical reasons), meaning that two points on an elliptic curve are added
- The addition operation can be explained geometrically or algebraically
- The geometric explanation is particularly useful for the addition of two points on an elliptic curve over $\mathbb{R}$

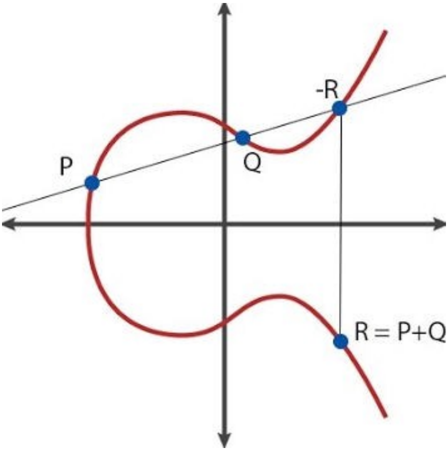
5. One-Way Functions

5.5 Elliptic Curve Cryptography

- If $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ are two distinct points on $E(\mathbb{R})$, then $R = P + Q = (x_3, y_3)$ is constructed as follows:
  - Draw a line through P and Q
  - This line intersects $E(\mathbb{R})$ in a third point
  - R is the reflection of this point on the x-axis.
- If $P = (x_1, y_1)$, then $R = 2P = (x_3, y_3)$ is constructed as follows:
  - Draw the tangent line to $E(\mathbb{R})$ at P
  - This line intersects $E(\mathbb{R})$ in a second point
  - R is the reflection of this point on the x-axis

5. One-Way Functions

5.5 Elliptic Curve Cryptography



5. One-Way Functions

5.5 Elliptic Curve Cryptography

- The fact that $\mathcal{O}$ is the neutral element of the point addition means that $P + \mathcal{O} = \mathcal{O} + P = P$ for all $P \in E(\mathbb{Z}_q)$
- If $P = (x, y) \in E(\mathbb{Z}_q)$, then $-P = (x, -y)$
- This yields another point on the elliptic curve (due to the symmetry of the curve related to the x-axis)
- In $E(\mathbb{Z}_{23})$, $P = (3, 10)$ has the inverse $-P = (3, 13)$ — because $-10 = -10 + 23 = 13$ in $\mathbb{Z}_{23}$
- P and $-P$ sum up to $\mathcal{O}$, i.e., $P + (-P) = \mathcal{O}$

5. One-Way Functions

5.5 Elliptic Curve Cryptography

- If $P = (x_1, y_1) \in E(\mathbb{Z}_q)$ and $Q = (x_2, y_2) \in E(\mathbb{Z}_q)$, then $P + Q = (x_3, y_3)$ can be computed as follows:

$$\begin{aligned} \lambda &= \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq Q \\ \frac{3x_1^2 + a}{2y_1} & \text{if } P = Q \end{cases} \\ x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= \lambda(x_1 - x_3) - y_1 \end{aligned}$$

5. One-Way Functions

5.5 Elliptic Curve Cryptography

- For $P = (3,10)$ and $Q = (9,7)$
  $\lambda = \frac{7-10}{9-3} = \frac{-3}{6} = 20 \cdot 4 = 80 \equiv 11 \pmod{23}$
  $x_3 = 11^2 - 3 - 9 = 121 - 3 - 9 = 109 \equiv 17 \pmod{23}$
  $y_3 = 11(3-17) - 10 = 33 - 187 - 10 = -164 \equiv 20 \pmod{23}$
- Consequently, $(3,10) + (9,7) = (17,20)$

$(0,1)$	$(0,22)$	$(1,7)$	$(1,16)$	$(3,10)$	$(3,13)$	$(4,0)$
$(5,4)$	$(5,19)$	$(6,4)$	$(6,19)$	$(7,11)$	$(7,12)$	$(9,7)$
$(9,16)$	$(11,3)$	$(11,20)$	$(12,4)$	$(12,19)$	$(13,7)$	$(13,16)$
$(17,3)$	$(17,20)$	$(18,3)$	$(18,20)$	$(19,5)$	$(19,18)$	

- EC Calculator

5. One-Way Functions

5.5 Elliptic Curve Cryptography

- If one adds $P = (3, 10)$ to itself, then $P + P = 2P = (x_3, y_3)$ is computed as follows:

$$\begin{aligned} \lambda &= \frac{3(3^2)+1}{20} = \frac{5}{20} = \frac{1}{4} = 4^{-1} \equiv 6 \pmod{23} \\ x_3 &= 6^2 - 6 = 30 \equiv 7 \pmod{23} \\ y_3 &= 6(3-7) - 10 = 18 - 42 - 10 = -34 \equiv 12 \pmod{23} \end{aligned}$$

- Consequently, $2P = (7, 12)$
- This can be iterated to compute multiples of P

5. One-Way Functions

5.5 Elliptic Curve Cryptography

- $3P = (19,5)$, $4P = (17,3)$, $5P = (9,16)$, $6P = (12,4)$, $7P = (11,3)$, $8P = (13,16)$, $9P = (0,1)$, $10P = (6,4)$, $11P = (18,20)$, $12P = (5,4)$, $13P = (1,7)$, $14P = (4,0)$, $15P = (1,16)$, $16P = (5,19)$, $17P = (18,3)$, $18P = (6,19)$, $19P = (0,22)$, $20P = (13,7)$, $21P = (11,20)$, $22P = (12,19)$, $23P = (9,7)$, $24P = (17,20)$, $25P = (19,18)$, $26P = (7,11)$, $27P = (3,13)$, and $28P = \mathcal{O}$
- After having reached $nP = \mathcal{O}$, a full cycle is finished and everything starts from scratch, i.e., $29P = P = (3,10)$, $30P = 2P = (7,12)$, ...

5. One-Way Functions

5.5 Elliptic Curve Cryptography

- In this example, the order of the group n is 28
- According to Lagrange's theorem, the order of any element divides n
- For example, the point $7P = (11, 3)$ has order 4 (that divides 28), because $4 \cdot 7P = 28P = \mathcal{O}$ (and $4P = (17, 3)$ has order 7, because $7 \cdot 4P = 28P = \mathcal{O}$)
- In ECC, all standard curves are chosen so that n is prime (so every element has order n and may serve as a generator)
- This is different from other cyclic groups, where a generator must first be found

5. One-Way Functions

5.5 Elliptic Curve Cryptography

- For every $E(\mathbb{F}_q)$, the group of points on that curve (together with $\mathcal{O}$) and the addition operation form a cyclic group
- ECC uses such a group and takes its security from the assumed intractability of the **elliptic curve discrete logarithm problem (ECDLP)**

Definition 5.10 (ECDLP)

If $E(\mathbb{F}_q)$ is an elliptic curve over $\mathbb{F}_q$, P a point on $E(\mathbb{F}_q)$ of order n, and Q another point on $E(\mathbb{F}_q)$, then it is to determine an $x \in \mathbb{Z}_n$ such that $Q = \underbrace{P + \ldots + P}_{x \text{ times}} = xP$

5. One-Way Functions

5.5 Elliptic Curve Cryptography

- There are no subexponential algorithms known to solve the ECDLP
- Again, this has the advantage (for the cryptographer) that the resulting elliptic curve cryptosystems are equally secure with smaller key sizes
- For example, to reach the security level of 2,048 (3,072) bits in a conventional public key cryptosystem like RSA, it is estimated that 224 (256) bits are sufficient in ECC
- Key length estimations
- This is the order of magnitude people work with today

5. One-Way Functions

5.5 Elliptic Curve Cryptography

- Based on the intractability assumption of the ECDLP, Neal Koblitz and Victor Miller independently proposed elliptic curve cryptosystems in the mid-1980s
- Such cryptosystems are best viewed as elliptic curve versions of DLP-based cryptosystems, in which the cyclic group (e.g., $\mathbb{Z}_p^*$ or a subgroup) is replaced by a group of points on an elliptic curve over a finite field
- Consequently, there are ECC variants of Diffie-Hellman, Elgamal, DSA, ...
- IFP-based cryptosystems have no useful ECC variants

5. One-Way Functions

5.5 Elliptic Curve Cryptography

- A distinguishing feature of ECC is that every user may select a different elliptic curve $E(\mathbb{F}_q)$
- This is true even if the same finite field $\mathbb{F}_q$ is used
- This flexibility has advantages and disadvantages
- For example, it may make interoperability difficult and raise concerns about backdoors (e.g., Dual_EC_DRBG)
- Anyway, implementing an elliptic curve cryptosystem is involved, and one has to be cautious about patent claims

5. One-Way Functions

5.5 Elliptic Curve Cryptography

- May standardization bodies are active in ECC
- Most importantly, the **elliptic curve digital signature algorithm (ECDSA)** is the elliptic curve variant of the DSA proposed in 1992
- It is standardized in NIST FIPS 186, ISO/IEC 14888-3 (and ISO/IEC 15946-1), ANSI X9.62, and IEEE Std 1363-2000
- P-256 from FIPS 186-4 is an elliptic curve that is particularly widely used in the field

5. One-Way Functions

5.5 Elliptic Curve Cryptography

- Mainly due to the Dual_EC_DRBG incident, people are worried about elliptic curves recommended by U.S. agencies
- This also applies to the curves promoted by the Standards for Efficient Cryptography Group (SECG) that are in line with NIST (e.g., secp256k1 as used in Bitcoin)
- Alternative curves
  - Brainpool curves (e.g., RFC 5639)
  - SafeCurves
    - Curve25519 (Ed25519 for signatures)
    - Curve448-Goldilocks (Ed448-Goldilocks for signatures)
    - E-521
    - ...

5. One-Way Functions

5.6 Final Remarks

- Most public key cryptosystems in use today are based on one (or several) one-way function(s)
- This is also true for ECC that operates in groups in which known special-purpose algorithms to compute discrete logarithms do not work
- It is sometimes recommended to use cryptosystems that combine different types of one-way functions
- This strategy becomes useless if all functions simultaneously turn out not to be one-way or a hardware device can be built that allows an adversary to efficiently invert them (e.g., a quantum computer)

Questions and Answers

Thank you for your attention