Cryptography

Samuel Jaques and Christophe Petit

Trinity Term 2022

Contents

1 Introduction

The course will cover introductory and advanced topics in cryptology. This year, the advanced part will focus on zero-knowledge proofs (the most popular option based on our poll), including their applications to e-voting.

Learning outcomes: The students will be able to manipulate security definitions and proofs by reductions as used in cryptography. They will understand the main cryptographic tools in use today, the security guarantees they can provide and their limitations. They will also learn some advanced and research topics in cryptography.

Synopsis This course will be organized as a reading course. We will assume no prior exposition to cryptology, hence start with basic concepts including essential cryptographic primitives (digital signatures, public and private key encryption, hash functions), proofs by reduction and major cryptographic algorithms in use today. We will then introduce zero-knowledge proofs and their applications to e-voting.

Organization The course is divided in four parts. For each part:

- We provide a set of reference reading and questions to guide you through this reading.
- We also provide a group project made of a few additional questions. This typically includes both theoretical questions and some implementation work.
- We ask you to prepare a 30min-1h oral presentation with your answers.
- We will meet every other week of the term for 2 hours. We will start these meetings with your presentation, often interrupting it with questions, then carry on with any further question you may have on the reading material.

Advanced warning Christophe Petit and Sam Jaques will be colecturing this module, with Christophe expected to cover the first and last part while Sam covers the middle parts.

Reading list

- Katz-Lindell, Introduction to Modern Cryptography [4].
- Galbraith, Mathematics of Public Key Cryptography [2].

Forum: There is a forum for the course on the Moodle page, intended for discussions and questions about the readings, exercises, group projects, and any other course-related topics. Since we will meet infrequently, please use the forum to ask questions about the material as you read through it.

2 Symmetric key cryptography

The main reference for this part is Katz-Lindell's book "Introduction to Modern Cryptography" [4].

2.1 Individual guided reading

Introduction Reference: Katz-Lindell [4], Chapter 1.

1. What is an encryption scheme? Get acquainted with the following terminology: plaintext, ciphertext, key, key generation algorithm, encryption algorithm, decryption algorithm.
2. What is Kerckhoff's principle? Do you think this principle makes sense? Could there be sensible exceptions?
3. What is Statistical cryptanalysis ? Implement (in your favorite language) one of the "historical ciphers" discussed in Section 1.3, and a statistical analysis attack against it.
4. Which properties should a "secure" encryption scheme satisfy? Try to come up with your own definition of what this means, then compare this definition with p20-21.
5. Imagine you have designed a new encryption scheme. How could you proceed to prove or argue about its security?
6. Compare the meaning of "Security proof" in cryptography with the usual meaning of "proof" in mathematics. Can a cryptographic scheme with a "security proof" still be insecure? Are "security proofs" valuable in cryptography? Should a "security proof" rather be called "security argument"?

Perfectly secure encryption Reference: Katz-Lindell [4], Chapter 2.

1. Study the "one-time pad" cryptosystem. Design and implement a similar system for symbols in an arbitrary group G.
2. Try to break the one-time pad cryptosystem using statistical analysis. Is it possible?
3. What is the meaning of "perfect secrecy"?
4. Prove that the one-time pad cryptosystem is "perfectly secret".
5. Could you use the one-time pad to make online purchases? Discuss.
6. Give an attack against one-time pad when part of the key is used more than once.

Private key encryption and pseudorandomness Reference: Katz-Lindell [4], Chapter 3.

1. Compare “information theoretical security” and “computational security”.
2. Define “efficient” algorithm and “negligible” probability. Compare “asymptotic” and “concrete” usual meanings for these words.
3. Study the indistinguishability definition (Definition 3.9). What does it say? Informally, does it imply that keys remain secret, that the encryption scheme cannot be inverted (without the secret key)? Is it a good security definition of an encryption scheme, or does it fail to capture some attacks that might be realistic?
4. What is a pseudo-random number generator (PRNG)? How can such an object be used in cryptography? What are the security properties that a PRNG must satisfy?
5. The one-time pad is perfectly secure but only if the key used is as large as the message. Describe how a PRNG can help remove that assumption. Are the security guarantees provided by the new scheme equivalent?
6. What is a stream cipher and how can such a thing be constructed?
7. Study the definition of IND-CPA security. How does it compare with the previous definition? Does the attack scenario captured by the definition make any sense in practice? Does the definition capture all the attack scenarios you can think of?
8. Suppose that an encryption scheme is deterministic. Show that it cannot satisfy the IND-CPA security property.
9. Study the definition of pseudorandom function (PRF), and how this object is used in Construction 3.25 to build an IND-CPA secure encryption scheme. Try to prove security of the encryption scheme (by reduction to the security of the pseudo-random function).
10. Study the definition of IND-CCA security. How does it compare with the previous definition? Does the attack scenario captured by the definition make any sense in practice? Does the definition capture all the attack scenarios you can think of?
11. Study the definition of pseudorandom permutation (PRP, Definition 3.38) and compare it with that of a pseudorandom function. Relate PRP and block cipher.
12. Does a PRP immediately lead to a secure encryption scheme?
13. What are the main “ encryption modes of operations”? What are their respective advantages, and which ones are most secure? Use Google to identify concrete applications where these modes are used.
14. Consider the use of *initial values* (IV) in these constructions. Is it important that these values are chosen randomly?
15. Compare block cipher and stream cipher.

Block cipher design Reference: Katz-Lindell [4], Chapter 5.

1. Try to come up with a sensible block cipher design, then read Chapter 5. Are there any similarities between your ideas and the ones described there? Will your scheme likely be vulnerable to some of the attacks described?

2. Suppose you need to use a block cipher. Is it best to design a new one, or use an existing one? What if you are an expert on block cipher design?

3. Given a pseudorandom function, how can we (provably) build a pseudorandom permutation? Study the answer provided by Feistel networks.

Message authentication and collision-resistant hash functions Reference: Katz-Lindell [4], Chapter 4.

1. What do we mean by authentication?

2. Is encrypting a message with an IND-CCA encryption scheme sufficient to guarantee message integrity?

3. What is a *message authentication code* (MAC), and how is it formally defined? What are the security properties expected of a MAC?

4. What are replay attacks? Do MACs offer any protection against replay attacks, and why?

5. Review the list of cryptographic objects defined so far, and try to build a MAC from one of them. Can you prove the security of your construction?

6. Study construction 4.3. Can you prove its security? Understand the proof provided in Katz-Lindell.

7. Consider a variable-length MAC constructed as follows: choose a pseudorandom function f; parse the message into blocks m_i of size equal to domain size of f; compute $t_i = f_k(m_i \oplus i)$; compute $t = t_1 \oplus t_2 \oplus \ldots \oplus t_n$. Provide a concrete attack against this construction.

8. What is CBC-MAC? Try to attack CBC-MAC with an attack as the one developed for the previous question, then try to prove its security. Check the security proof provided in the reference book.

9. Show that adding a random IV in the CBC-MAC construction (instead of a fix one) would NOT lead to a secure MAC, i.e. it would lead to an attack against the scheme.

10. What is a message extension attack? Show that CBC-MAC, as described in Construction 4.7, is vulnerable to such an attack. Discuss possible ways to thwart the attack.

11. Define a hash function, collision resistance, second preimage resistance and preimage resistance. What are the relationships between these security properties?

12. What is a random oracle? Relate this notion to the previous ones.

13. Suppose there are randomly chosen 25 people in a room. What is the probability that two of them have the same birthday? How many people are needed for this probability to be larger than 0.5?

14. How hard is it to compute collisions for the most secure of all hash functions?

15. What is the purpose of the Merkle-Damgaard transform? Study Construction 4.11 and discuss its security.

16. Should hash functions be *keyed* or not? Is the key supposed to be secret in this context?

17. Study the NMAC and HMAC constructions and their security proofs.

18. A MAC can also be used to improve an IND-CCA encryption scheme into a CCA-secure encryption scheme. Study Construction 4.17; try to build a CCA attack against it; then try to argue its security; and finally check the proof provided in the book.

19. Suppose you have a secure encryption scheme and a secure MAC. How should you combine them to obtain both the necessary confidentiality and authentication guarantees? Discuss the Enc-and-MAC, Enc-then-MAC, MAC-then-Enc approaches. Under which conditions is each of these approaches secure? Have they been or are they still used in practice?

20. Authenticated encryption

2.2 Group project

The goals of this group project are to practice security reductions and to understand some aspects of pseudorandom generation in cryptography.

1. Formally define pseudo-random number generator, one-way function and hard-core bit.

2. Show how to build a secure PRNG from any one-way function.

3. Linear Feedback Shift Registers (LFSR) were once considered good pseudo-random number generators. Describe what is an LFSR and implement one using your favourite computer language.

4. Explain how the Berlekamp-Massey algorithm can "break" such a construction, and use it to break your own LFSR implementation.

5. Consider the use of *initial values* (IV) in encryption modes of operations. Is it important that these values are chosen randomly? Implement a concrete attack against one mode of operation (with a block cipher of your choice) when your LFSR is used to generate IV values.

Hints: there is a well-known PRNG construction from any one-way function, which you can find in the Katz-Lindell book [4]. There are numerous sources for the Berlekamp-Massey algorithm; a cryptography source is Antoine Joux's book on *Algorithmic Cryptanalysis* [3]. If you cannot implement the algorithm, we suggest that you use an existing implementation available online, with proper referencing.

3 Public key cryptography

3.1 Individual guided reading

Number Theory Background. Chapter 9 from Katz-Lindell, skipping sections 9.2.2, 9.2.5, 9.3.4, and 9.4.

1. (Problem 9.10) For the group $\mathbb{Z}_{24}$:
   (a) List the elements of the group.
   (b) Is this group cyclic?
   (c) Is 18 a generator? Is 5 a generator?

2. (Problem 9.11) For the group $\mathbb{Z}_{21}^*$:
   (a) List the elements in this group. How many are there?
   (b) What is $\phi(21)$?
   (c) Find a generator of this group, and an element that is not a generator.
   (d) More generally: Let p be a prime, and suppose $g \in \mathbb{Z}_p^*$. How do we decide if g generates $\mathbb{Z}_p^*$? How can we *find* a generator g?

3. What is the group structure of $\mathbb{Z}_p^*$? Given this, for which integers m can you compute an mth root modulo p?

4. What does it mean for a group problem to be "easy" or "hard"? In $\mathbb{Z}_p^*$, which operations are easy or hard?
   (a) Finding $z \equiv xy \mod p$, given x and y
   (b) Finding $z \equiv x^2 \mod p$, given x
   (c) Finding z such that $zx \equiv 1 \mod p$, given x
   (d) Finding $z \equiv x^y \mod p$ given x and y with $y < p$
   (e) Finding z such that $x^z \equiv y \mod p$, given x and y
   (f) Finding z such that $z^x \equiv y \mod p$, given x and y

5. PRIMES is the problem of deciding whether an input integer N is prime or not. Easy: show that PRIMES is in co-NP. Hard: show that PRIMES is in NP.

6. Consider the following probabilistic "algorithm" to solve PRIMES:
   (a) On any input n, return "Composite"

   By the prime number theorem, the fraction of integers of size $\approx n$ which are prime is only $O(\frac{1}{\log n})$. This means the algorithm succeeds with probability $1 - O(\frac{1}{\log n})$, which asymptotically approaches 1. Thus, this is a valid probabilistic algorithm for this problem. What is the error in the logic here? How would you define "probabilistic algorithm" to exclude this error?

7. Consider the following experiment $\mathsf{SquareRoot}_{\mathcal{A},\mathsf{GenModulus}}(n)$:

(a) Run $\mathsf{GenModulus}(1^n)$ to obtain (N, p, q).
(b) Select a random $x \in \mathbb{Z}_N^*$
(c) $\mathcal{A}$ is given N and x and outputs y.
(d) The output of the experiment is 1 if $y^2 \equiv x \mod N$, and 0 otherwise.

Show that, given an efficient algorithm $\mathcal{A}$ that succeeds at SquareRoot, there is an efficient algorithm that succeeds at Factor (defined in Section 9.2.3). (This reduction becomes relevant for factoring cryptanalysis later).

8. (Problem 9.19) Formally define the CDH assumption. Prove that hardness of the CDH problem relative to G implies hardness of the discrete-logarithm problem relative to G, and that hardness of the DDH problem relative to G implies hardness of the CDH problem relative to G.

Public Key Encryption Reference: Katz-Lindell, Chapter 11, and Chapter 1 (sections 12.1 to 12.4.2, and 12.5.1, 12.5.2, 12.5.4, and 12.5.6)

1. (Problem 11.3) Describe a person-in-the-middle attack on the Diffie–Hellman protocol where the adversary shares a key k_A with Alice and a (different) key k_B with Bob, and Alice and Bob cannot detect that anything is wrong.

2. (Problem 11.4) Consider the following key-exchange protocol:

   (a) Alice chooses uniform $k, r \in \{0,1\}^n$, and sends $s := k \oplus r$ to Bob.
   (b) Bob chooses uniform $t \in \{0,1\}^n$, and sends $u := s \oplus t$ to Alice.
   (c) Alice computes $w := u \oplus r$ and sends w to Bob.
   (d) Alice outputs k and Bob outputs $w \oplus t$.

   Show that Alice and Bob output the same key. Analyze the security of this protocol against a passive eavesdropper.

3. Suppose that the modulus p generated for Diffie-Hellman or El-Gamal is actually composite. What problems will this cause?

4. Pick your favourite cyclic group that isn't $\mathbb{Z}_n^*$ or points on an elliptic curve. Probably this group will not work well for Diffie-Helman or El-Gamal; what property is it missing?

5. Some implementations of RSA use a fixed exponent e (often 3 or 65537). Since new primes p and q are generated each time it is used, there is some risk that e divides $p-1$ or $q-1$. Suppose that this happens, and someone encrypts a (padded) message as $c \equiv m^e \mod N$. Can c be decrypted by someone knowing the secret key (p, q)?

6. Related, suppose someone pads their message m to m', and m' is no longer co-prime to N. What happens when m' is encrypted? Should an implementation check to ensure that this does not occur?

7. (Problem 12.12):One of the attacks on plain RSA discussed in Section 12.5.1 involves a sender who encrypts two related messages using the same public key. Formulate an appropriate definition of security ruling out such attacks, and show that any CPA-secure public-key encryption scheme satisfies your definition.

8. (Problem 12.14): Consider the following modified version of padded RSA encryption: Assume messages to be encrypted have length exactly $\|N\|/2$. To encrypt, first compute $\hat{m} := \texttt{0x00}\|r\|\texttt{0x00}\|m$ where r is a uniform string of length $\|N\|/2 - 16$. Then compute the ciphertext $c := \hat{m}^e \mod N$. When decrypting a ciphertext c, the receiver computes $\hat{m} := c^d \mod N$ and returns an error if $\hat{m}$ does not consist of 0x00 followed by $\|N\|/2 - 16$ arbitrary bits followed by 0x00. Show that this scheme is not CCA-secure. Why is it easier to construct a chosen-ciphertext attack on this scheme than on PKCS #1 v1.5?

Digital Signatures Reference: Chapter 13, up to the end of 13.6 (section 13.5.3 has some material on elliptic curves we can skip). Section 13.7 on TLS is a good extra section if you are interested in the real-world applications.

1. Compare a digital signature scheme to a MAC. In what scenarios would you use each?

2. (Problem 13.2) In Section 13.4.1 we showed an attack on the plain RSA signature scheme in which an attacker forges a signature on an arbitrary message using two signing queries. Show how an attacker can forge a signature on an arbitrary message using a single signing query.

3. (Problem 13.3)Assume the RSA problem is hard. Show that the plain RSA signature scheme satisfies the following weak definition of security: an attacker is given the public key $\langle N, e\rangle$ and a uniform message $m \in \mathbb{Z}_N^*$. The adversary succeeds if it can output a valid signature on m without making any signing queries.

4. (Problem 13.4) Consider a "padded RSA" signature scheme where the public key is $\langle N, e\rangle$ as usual, and a signature on a message $m \in \{0,1\}^\ell$ is computed by choosing uniform $r \in \{0,1\}^{2n-\ell-1}$ and outputting $[(r\|m)^d \mod N]$.

   (a) How can verification be done for this scheme?

   (b) Show that this scheme is insecure.

5. Suppose someone re-uses the same value of k for Schnorr signatures. Describe an attack on this. (Sony actually made this mistake with the Playstation 3, with DSA instead of Schnorr).

6. In April 2022 a lot of zero-knowledge proofs based on the Fiat-Shamir transform were found to have a major vulnerability. In fact this exact vulnerability appears in Construction 13.12 in the Katz-Lindell book. In the interactive Schnorr identification scheme, the verifier has $\mathbb{G}$, q, g, and y (q is the order of g, and $y = g^x$ for the prover's secret key x). In practice, these values must also be sent to the verifier from the prover, so the prover must also "commit" to these values via the hash. Instead, in Construction 13.12, $r := H(I, m)$, where $I = g^k$ for random k and message m. Since this does not include y in the hash, show how a malicious prover could choose y to forge a signature. Can you think of scenarios where this attack would not be a problem, and scenarios where it would be a problem?

7. This question describes a bug in digital signatures present in Java from 2020 until April 2022.

   (a) In the last step of the DSA algorithm, the verifier must compute $g^{\alpha s^{-1}}$. Show that if q (a prime) is the order of g, that if we define $x := s^{q-2} \mod q$, then $g^{s^{-1}} = g^x$.

   (b) If we compute s^{-1} using the method of the last question, what happens when we compute the inverse of 0?

   (c) In the non-interactive version of DSA (Construction 13.13), suppose the function F has the property that $F(1) = 0$. Show that if we use the previous method to compute inverses, and we forget to check that $r, s \neq 0$, that the signature $(0, 0)$ will pass the verification for any message and any public key.

   It might seem strange to use a hash function F such that $F(1) = 0$, but in fact the elliptic curve DSA does precisely this!

3.2 Group project

This group project considers various notions of security related to the RSA cryptosystem. Feel free to use any programming language, though Sage is well-suited to these kinds of problems.

1. Consider the following problems (where N is assumed to have the form $N = pq$ for primes p and q):

   - The RSA problem (see Section 9.2.4).
   - Given y, e, and N, find x such that $x^e \equiv y \mod N$.
   - Given e and N, find $d < N$ such that $x^{ed} \equiv 1 \mod N$ for all $x \in (\mathbb{Z}/N\mathbb{Z})^*$.
   - Given N, find the two prime factors.

   Which problems reduce to which other problems?

2. Factoring is generally a hard problem, but certain instances can be easy. For example, a bad way to generate primes is to pick one prime randomly, then pick another prime that is close to the first (e.g., the next largest prime). The following 2048-bit RSA modulus was generated in this way. Factor it.

   205204139815946734809513614540475892533080880817391511275086853366
   200172643852060732544960965303687051678251280236318296045391766599
   722619076991985428134096549695239089151751185608839826200085859637
   928410494733293546366651345515849514411349684602515198183825443771
   952742613130712066667587867205376684104969207

3. Explain the difference between a key recovery attack and other possible attacks.

4. Read Boneh's "Twenty Years of Attacks on the RSA Cryptosystem" [1]. Implement Wiener's low private exponent attack, and use it to recover the private exponent d from the following RSA public key:

   N = 61896782498722570672675460781419087071557527498815117364800
   80710597994412035208280041226270153691844373131020930880904079333 2
   54542104536652629586957457204800543269010253200597593526955583014 4
   80139668832999080410462538560239321088575624277429996266568693113 9
   542131667314970475573849523268774957064129501489 3953

   e = 169713265362374355734526259168457537473720157506821980155782100159
   4394855370769369701138794187817610014356376860997623459163422229288883513
   9913351932067475375247495922459951772990110292382983781668395554229494598
   7112908793724732330116912806568804169170817571057645696647913373108529358
   887925195841409540236059

5. Read Section 10.1.3 on the quadratic sieve algorithm for factoring, and implement it. This paper might also be helpful `https://math.dartmouth.edu/~carlp/PDF/qs08.pdf`. Try to factor this 64-bit RSA modulus with your sieve: 8741435393895281443.

4 Zero-Knowledge Proofs

4.1 Individual guided reading

Basic, intuitive introduction. See also the various fun ideas here.

More formal survey(These notes cover the same information).

Original Fiat-Shamir paper

Goldreich, Micali, and Wigderson

1. Schnorr's proof system proves knowledge of some w such that $g^w = h$ for a public value h. Show that if the order of g is composite, then a dishonest verifier can obtain some information about the secret exponent.

2. Given the previous question, is a Schnorr proof insecure if g is an element of $\mathbb{Z}_N^*$ where N is an RSA modulus?

3. Show that if you can find second pre-images of the hash used in the Fiat-Shamir transform, you can forge a signature.

4. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups with the same prime order q. Let g_1 and g_2 be two generators. Alice has one private key $x \in \mathbb{Z}_q^*$ and two public keys g_1^x and g_2^x (the same exponent for each). She wants to prove to Bob that the exponents are the same, so she and Bob engage in the following protocol:

Alice (Prover)		Bob (Verifier)
$r \leftarrow \mathbb{Z}_q^*$		
$t_1 \leftarrow g_1^r$		
$t_2 \leftarrow g_2^r$	$\xrightarrow{t_1,t_2}$	
	$\xleftarrow{c}$	$c \leftarrow \mathbb{Z}_q$
$s \leftarrow r + cx \mod q$	$\xrightarrow{s}$	

   (a) Construct a method for Bob to verify this.

   (b) Show that this scheme is complete, sound, and honest-verifier zero-knowledge.

   (c) Construct a non-interactive version of this scheme, i.e., a single message that Alice can send to Bob that proves the same equality.

   (d) In certain applications, we want to anonymise a public/private key-pair (g, g^x) to (h, h^x). This can be easily done by picking a random r and setting $h := g^r$, and $h^x = (g^x)^r$. Suppose Alice randomizes her key in this way and want to prove to Bob that the new key matches the old one. They could use the protocol just described, but with $\mathbb{G}_2 = \mathbb{G}_1$.

      i. Will that affect your security proofs?

      ii. Here Alice is allowed random values in the second keypair. Is your non-interactive scheme vulnerable to the attack of problem 6? If so, construct a scheme that is not vulnerable. If your scheme was already secure, construct a scheme that *is* vulnerable.

5. Given g, g^x, g^y, and g^z, construct a zero-knowledge protocol to prove that $z = xy \mod q$ (where q is the order of g).

6. Consider the following protocol to prove knowledge of the factorisation of an RSA modulus:

Alice (Prover)		Bob (Verifier)
pk= (N, e), sk=d	$\xrightarrow{N,e}$	
		$m \leftarrow_{\$} (\mathbb{Z}_N)^*$
	$\xleftarrow{c}$	$c = \text{RSA.Enc}(m, N, e)$
$m' = \text{RSA.Dec}(c, N, d)$	$\xrightarrow{m'}$	
		Accept iff $m = m'$

   Is this a complete, sound, honest-verifier-zero-knowledge Σ-protocol?

7. Venturi's notes state that the Fiat-Shamir transform can turn every Σ-protocol into a signature scheme, but this is not exactly true. Consider the following protocol to prove knowledge of the factorisation of an RSA modulus (where H is a secure hash function, and $H' : \{0,1\}^* \to \{1,2,3,4\}$ is a pseudorandom function)

Alice (Prover)		Bob (Verifier)
pk= (N, e), sk=p, q		
$r \leftarrow_{\$} (\mathbb{Z}_N)^*$		
$t = r^2 \mod N$	$\xrightarrow{N,e,t}$	
		$s \leftarrow_{\$} (\mathbb{Z}_N)^*$
	$\xleftarrow{h,v}$	$h = H(s), v = s^2 \mod N$
Compute s_1, s_2, s_3, s_4 as square roots of v		
Set $s' = s_i$ such that $H(s_i) = h$		
Compute r_1, r_2, r_3, r_4 as square roots of v		
Let $i = H'(s') \in \{1,2,3,4\}$	$\xrightarrow{r_i,s'}$	
		Accept iff $s = s'$ and $r_i^2 \equiv t \mod N$

   (the t and r_i are only there to prove special soundness). Explain why the Fiat-Shamir transform does not create a valid signature scheme out of this protocol.

4.2 Group project

Goldwasser, Micali, and Wigderson show that all languages in NP have zero-knowledge proofs by showing that an NP-complete problem – 3-colouring – admits zero-knowledge proofs. However, in practice, 3-colouring is hard to work with, and arithmetic circuits are more natural. The goal of this group project is to implement a zero-knowledge proof system for arithmetic circuits.

1. A *commitment scheme* is a method to commit to some data without revealing it. It consists of two algorithms: $\mathsf{Commit}(m) \to (c, o)$ produces a public commitment c and a secret value o that can open the commitment; $\mathsf{Open}(c, o) \to m$ reveals the committed value.

   A Pederson commitment scheme has two public points $g, h \in \mathbb{G}$ for a cyclic group $\mathbb{G}$ of order q (where no one should know the discrete log of h with respect to g). To commit to a message $m \in \mathbb{Z}_q$, select a random $r \in \mathbb{Z}_q$ and compute $c := g^m h^r$. The opening is $o := (m, r)$. To open a commitment, one computes $g^m h^r$ and checks if this equals c, and if so, output m.

(a) Implement this commitment scheme.

(b) Show that the scheme is statistically *hiding*: Given a commitment c, the probability that $\mathsf{Commit}(m)$ outputs c is the same for all messages m.

(c) Show that the scheme is computationally *binding*: Under a standard cryptographic assumption, an adversary should not be able to find one commitment c and two openings o and o' such that $\mathsf{Open}(c, o) = m$ and $\mathsf{Open}(c, o') = m'$.

2. Suppose Alice and Bob both have a commitment $c = g^m h^r$ and Alice wants to prove to Bob that she has knows an opening (m, r). Using a Fiat-Shamir transform, implement a non-interactive version of the following protocol:

Alice (Prover)		Bob (Verifier)
$c,\ m,\ r$		c
$k_1, k_2 \leftarrow_\$ \mathbb{Z}_q$		
$t \leftarrow g^{k_1} h^{k_2}$	$\xrightarrow{t}$	
	$\xleftarrow{h}$	$h \leftarrow_\$ \mathbb{Z}_q^*$
$s_1 \leftarrow k_1 + hm$		
$s_2 \leftarrow k_2 + hr$	$\xrightarrow{s_1, s_2}$	
		Accept iff $t = g^{s_1} h^{s_2} c^h$

(a) Show that this is a zero-knowledge proof.

3. Given three commitments to c_1, c_2, and c_3 to messages m_1, m_2, m_3, describe and implement a non-interactive zero-knowledge proof that $m_1 m_2 = m_3 \mod q$. (use Q5).

4. Given commitments to c_1, c_2 to messages m_1, m_2, describe and implement a method to produce a commitment c to $m_1 + m_2 \mod q$. Given openings $o_1 = (m_1, r_1)$ and $o_2 = (m_2, r_2)$, can you produce a valid opening $o = (m, r)$ for $m = m_1 + m_2 \mod q$?

5. An *arithmetic circuit* is a circuit where the wires represent values in $\mathbb{Z}_q$, and there are $q + 2$ gates:

- a multiplication gate, which takes two input wires x and y and outputs $z = xy \mod q$,
- an addition gate, which takes two input wires x and y and outputs $z = x + y \mod q$.
- q different constant gates, which take no inputs and output a constant value.

If $q = 2$ this is traditional binary circuit with AND and XOR gates.

The arithmetic circuit satisfiability problem is: given a circuit $\mathcal{C}$ and an output string y, is there an input string x such that evaluating $\mathcal{C}$ on input x will output y?

This is an NP-complete problem (3-SAT is easily reducible to this form when $q = 2$, and for larger q one can directly simulate boolean circuits).

Design and implement a non-interactive zero-knowledge proof system for the arithmetic circuit satisfiability problem. In terms of the size of the circuit, what is the asymptotic complexity for:

(a) the prover's computational complexity;

(b) the size of the proof;

(c) the verifier's computational complexity.

5 Application of Zero-Knowledge Proofs: e-voting

Zero-knowledge proofs have countless applications, including e-cash, group, ring and blind signatures, and ZKSNARKs.

Here we will focus on e-voting. Papers to read include Helios, Helios2.0 and Belenios.

5.1 Individual guided reading

1. What are the main security properties you would require from a voting scheme?
2. Explain what is meant by "end-to-end verifiable voting system".
3. Recall the ElGamal encryption protocol and its security properties, and describe its *homomorphic properties*.
4. What is exponential ElGamal protocol ? Relate this protocol and its security to the previous one. Describe its homomorphic properties.
5. Explain how the Sato-Kilian proofs of a shuffle works.
6. Describe potential attack goals from a voter, a voting authority, Helios or an external party. Explain if and how Helios protects against these attacks.
7. What is the purpose of the "Coerce me" button in the first version of Helios ?
8. Assuming nobody performs auditing, what are the main security consequences?
9. What are the main differences between Helios and Helios 2.0?
10. What is treshold ElGamal encryption?
11. Explain how the election result is verified in Helios, Helios 2.0 and Belenios?
12. What are the main differences between Helio and Belenios voting systems?

5.2 Group project

The goal of this group project is to generalize the use of zero-knowledge proofs to other election systems.

- Choose a cardinal and a ranking voting system in use in the world. (Or alternatively, choose your preferred voting system not already implemented by some Helios variant.)
- Describe these systems, then sketch an end-to-end verifiable voting system that could be used to implement these elections. In particular, describe the non interactive zero-knowledge proof systems needed in your systems.
- Analyze the efficiency of your solutions; can you improve it?
- Show that your protocols are secure under well established assumptions.

References

[1] Dan Boneh. Twenty years of attacks on the rsa cryptosystem. *NOTICES OF THE AMS*, 46:203–213, 1999.

[2] Steven D. Galbraith. *Mathematics of Public Key Cryptography*. Cambridge University Press, 2012.

[3] Antoine Joux. *Algorithmic Cryptanalysis*. Chapman & Hall/CRC, 1st edition, 2009.

[4] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC Press, 2007.