NTNU | Norwegian University of Science and Technology

COMMITMENTS AND ZERO-KNOWLEDGE

TTM4205 – Lecture 14

Tjerand Silde

24.10.2023

Contents

Contents

Reminder: Special Topic Project

The deadline for submitting group and topic is **Nov 1st**.

Reference Group Meeting

We are planning a new meeting next week. Please provide feedback to me or to the reference group members. We will provide time for you to discuss in a later lecture.

Reference Material

These slides are based on:

- The referred papers in the slides
- BS: parts of chapter 19 and 20
- DW: parts of chapter 7 and 15.3

Informal Definitions

Commitments

A *commitment* is a way to bind yourself to information that later can be opened. It is important that it is not possible to change the committed value afterwards, and that the commitment does not leak the committed value itself.

Example: a guarded safe where you know the code to open.

Informal Definitions

Zero-Knowledge Proofs

A *zero-knowledge proof* is a message or a communication protocol for a *prover* to convince a *verifier* that some statement is true without revealing why or how it is true.

A cheating prover should not be able to convince the verifier about false statements, and the verifier should not learn anything else than the fact that the statement is true.

Use-cases

Commitments and zero-knowledge proofs are widely used in among others the following settings:

- To create digital signatures
- Anonymous contact-tracing (implemented in Smittestopp 2.0)
- Electronic voting systems
- Privacy-preserving transactions
- Multi-party computation protocols

Matthew Green in fundamentals November 27, 2014 4,100 Words

Zero Knowledge Proofs: An illustrated primer

One of the best things about modern cryptography is the beautiful terminology. You could start any number of punk bands (or Tumblrs) named after cryptography terms like ‘hard-core predicate’, ‘trapdoor function’, ‘ or ‘impossible differential cryptanalysis’. And of course, I haven’t even mentioned the one term that surpasses all of these. That term is ‘*zero knowledge*‘.

Figure: https://blog.cryptographyengineering.com/2014/11/27/zero-knowledge-proofs-illustrated-primer

Commitment Schemes and Zero-Knowledge Protocols (2011)

Ivan Damgård and Jesper Buus Nielsen
Aarhus University, BRICS

Abstract

This article is an introduction to two fundamental primitives in cryptographic protocol theory: commitment schemes and zero-knowledge protocols, and a survey of some new and old results on their existence and the connection between them.

Figure: https://homepages.cwi.nl/~schaffne/courses/crypto/2014/papers/ComZK08.pdf

Contents

Algorithms

A commitment scheme consists of the following algorithms:

- **KGen** Outputs public parameters pp.
- **Com** Takes as input pp and a message m. It outputs a commitment cmt and an opening op.
- **Open** Takes as input pp, cmt and op and outputs 1 or 0.

Here, op usually consists of m and some randomness w.

Binding

A commitment is *binding* if it is hard to find two valid openings $\mathsf{op} = (m, w)$ and $\mathsf{op}' = (m', w')$ such that $\mathsf{Open}(\mathsf{cmt}, \mathsf{op})$ and $\mathsf{Open}(\mathsf{cmt}, \mathsf{op}')$ outputs 1 and $m \neq m'$.

This is similar to collision resistance for hash functions.

Hiding

A commitment is *hiding* if it is hard to decide if cmt is a commitment to a given message m or if cmt is sampled uniformly at random from the commitment space.

This is similar to CPA security for encryption schemes.

Hash-Based Commitments

Q: Are the following secure commitment schemes for hash function H, message m, and randomness w?

Hash-Based Commitments

Q: Are the following secure commitment schemes for hash function H, message m, and randomness w?

- Let Com output $\mathsf{cmt} = H(m)$ and $\mathsf{op} = m$.

Hash-Based Commitments

Q: Are the following secure commitment schemes for hash function H, message m, and randomness w?

- Let Com output $\mathsf{cmt} = H(m)$ and $\mathsf{op} = m$.
- Let Com output $\mathsf{cmt} = H(m, w)$ and $\mathsf{op} = (m, w)$.

Hash-Based Commitments

Are the following secure commitment schemes for hash function H, message m, and randomness w?

Hash-Based Commitments

Are the following secure commitment schemes for hash function H, message m, and randomness w?

- Let Com output $\mathsf{cmt} = H(m)$ and $\mathsf{op} = m$.
  Hiding only if m is pseudo-random.
  Binding if H is collision-resistant.

Hash-Based Commitments

Are the following secure commitment schemes for hash function H, message m, and randomness w?

- Let Com output $\mathsf{cmt} = H(m)$ and $\mathsf{op} = m$.
  Hiding only if m is pseudo-random.
  Binding if H is collision-resistant.

- Let Com output $\mathsf{cmt} = H(m, w)$ and $\mathsf{op} = (m, w)$.
  Hiding, if w is pseudo-random.
  Binding if H is collision-resistant.

ElGamal Commitment

Let $\mathbb{G}$ be a group of prime order p and let g and h be independent generators for $\mathbb{G}$. Let m be a message in $\mathbb{G}$ and w be uniform randomness in $\mathbb{Z}_p$.

An ElGamal commitment is computed as $\mathsf{cmt} = (g^w, m \cdot h^w)$.

Q: Is this commitment scheme hiding and binding?

Security

The ElGamal commitment scheme is:

- (computationally) hiding if w is pseudo-random and the DLOG problem is hard in $\mathbb{G}$.
- (unconditionally) binding since only one w exist for g^w.

Backdoor

We must be a bit careful about how we choose parameters.

Backdoor

We must be a bit careful about how we choose parameters.

- How can we break the scheme if we know $t = \log_g h$?

Backdoor

We must be a bit careful about how we choose parameters.

- How can we break the scheme if we know $t = \log_g h$?
- We break hiding by computing $m = (m \cdot h^w) \cdot (g^w)^{-t}$.

Mitigations

We must make sure that no one knows $t = \log_g h$, for example by computing both generators as outputs from a random oracle (hash function) on publicly agreed input, e.g., a given number of decimals of π or e or lottery numbers etc.

Contents

Algorithms

Let x be a NP-statement and let w be a witness such that a given relation (e.g. discrete logarithm) $\mathcal{R}(w, x)$ is satisfied.

A zero-knowledge proof consists of the following algorithms:

- KGen Outputs public parameters pp.
- Prove Takes as input pp, x and w. It outputs a proof π.
- Verify Takes as input x and π and outputs 1 or 0.

The Prove algorithm might be an interactive protocol.

Soundness

A zero-knowledge proof is *sound* if it is hard for a cheating prover to produce an accepting proof π for a statement x without there existing or the prover knowing a witness w.

This is similar to binding for commitment schemes.

Zero-Knowledge

A zero-knowledge proof is *zero-knowledge* if it is hard for a cheating verifier to learn anything about w when given x and π, except for learning that the relation $\mathcal{R}(w, x)$ is satisfied.

This is similar to hiding for commitment schemes.

Proof of DL

Given a group $\mathbb{G}$ of prime order p with generator g where the relation $\mathcal{R}(w, x)$ is satisfied if $x = g^w$. The DL ZK-proof:

1. Prover samples $r \leftarrow\$ \mathbb{Z}_p$ and sends $R = g^r$ to the verifier.
2. Verifier samples $c \leftarrow\$ \mathbb{Z}_p$ and sends c to the prover.
3. Prover compute $z = r - c \cdot w$ and sends z to the verifier.
4. If $R = g^z \cdot x^c$ then the verifier outputs 1, otherwise 0.

This is the interactive version of the Schnorr signature scheme without the message m and hash function H.

Security

We argue *soundness* as following:

A prover that does not know w have to guess c in advance to be able to answer the challenge correctly (unless it can compute DL, but then it could find w in the first place).

Assuming that the prover can guess c, then it can sample a random z and compute R as $R = g^z \cdot x^c$ and send it to the verifier in the first round. The probability of cheating is $1/p$.

(A proper proof would create an *extractor* using *rewinding*.)

Security

We argue *zero-knowledge* as following:

A verifier receive R and z from the prover. r is sampled uniformly at random, so R is a uniformly random element in $\mathbb{G}$. By a similar argument, z is a uniform element in $\mathbb{Z}_p$.

We create a *simulator* that does the following:

1. sample uniform c from $\mathbb{Z}_p$
2. sample uniform z from $\mathbb{Z}_p$
3. compute $R = g^z \cdot x^c$ in $\mathbb{G}$
4. output the transcript (R, c, z)

This transcript is identically distributed as a real execution.

Fiat-Shamir Transform

To make an interactive protocol non-interactive, we use the *Fiat-Shamir transform*, where the challenge c is the output of a hash function H applied to the context of the proof, e.g., the statement, public parameters and messages.

For example, $c = H(\mathsf{pp}, R)$ in the proof system above. Then we do not need interaction. Or $c = H(\mathsf{pp}, R, m)$ for signing m.

Fake Proofs

It is extremely important to hash everything when using Fiat-Shamir! Otherwise the prover can produce fake proofs.

Q: How can we fake the DL proof if $c = H(\text{pp})$?

Fake Proofs

It is extremely important to hash everything when using Fiat-Shamir! Otherwise the prover can produce fake proofs.

Q: How can we fake the DL proof above if $c = H(\mathrm{pp})$?

A: We know c before we need to choose R (simulator).

How not to prove your election outcome

Thomas Haines[*], Sarah Jamie Lewis[†], Olivier Pereira[‡], and Vanessa Teague[§]
[*]Norwegian University of Science and Technology
[†]Open Privacy Research Society, Canada
[‡]UCLouvain – ICTEAM – B-1348 Louvain-la-Neuve, Belgium
[§]The University of Melbourne – School of Computing and Information Systems, Melbourne, Australia

Figure: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9152765

Questions?