

# SS2 - exercises

Everyone

August 10, 2022

## Abstract

## Contents

<b>1</b>	<b>What is the identification scheme?</b>	<b>2</b>
1.1	Correctness . . . . .	2
1.1.1	Human readable . . . . .	2
1.2	Security . . . . .	2
1.2.1	Passive adversary . . . . .	3
1.2.2	Active adversary . . . . .	4
<b>2</b>	<b>What is a Discrete Logarithm Problem</b>	<b>4</b>
2.1	In human words . . . . .	4
2.2	Mathematically . . . . .	5
<b>3</b>	<b>What is Signature Scheme?</b>	<b>5</b>
3.1	Correctness . . . . .	5
3.1.1	Human readable . . . . .	5
3.2	Security . . . . .	6
3.2.1	Human readable . . . . .	6
<b>4</b>	<b>What is security model?</b>	<b>6</b>
<b>5</b>	<b>What is the proof of security?</b>	<b>6</b>
<b>6</b>	<b>What is hash function?</b>	<b>6</b>
6.1	Hash function requirements . . . . .	6
6.2	Secure hash requirements . . . . .	6
6.3	ROM - Random Oracle Model . . . . .	7
<b>7</b>	<b>Diffie-Hellmann</b>	<b>7</b>
7.1	What is CDH problem? . . . . .	7
7.2	What is DDH problem? . . . . .	7
7.3	What is GDH problem? . . . . .	8
<b>8</b>	<b>Knowledge proofs</b>	<b>8</b>
8.1	What is IZKP? . . . . .	8
8.2	What is NIZKP? . . . . .	9
<b>9</b>	<b>Goh-Jarecki</b>	<b>9</b>
<b>10</b>	<b>Pairing</b>	<b>9</b>
<b>11</b>	<b>Modified Schnorr</b>	<b>9</b>

# 1 What is the identification scheme?

It is an interactive zero knowledge proof of knowledge of secret key corresponding to the public key (knowledge of a discrete logarithm of a public key). IS is a tuple of  $(Init, KeyGen, P, V, \pi)$ , where

- *Init* - produces the space of computation, it takes  $X$  (security parameter) as input and outputs  $PAR$
- *KeyGen* - key generation function, takes  $PAR$  and outputs a pair of (secret key, public key)
- *P* - prover, interactive Turing Machine that takes secret key and interacts with another ITM  $V$  in the protocol  $\pi$
- *V* - verifier, ITM that takes public key and interacts with prover in the protocol  $\pi$
- $\pi$  - protocol, in which  $P$  interacts with  $V$ , the result is that  $V$  accepts  $P$  (it has secret key) or rejects  $P$  (it doesn't have secret key)

Functionalities:

- identification - system identifies its users
- correctness
- security

## 1.1 Correctness

$$Pr[PAR \leftarrow \lambda; (sk, pk) \leftarrow Gen(PAR); \pi(P(sk), V(pk)) \rightarrow 1] = 1$$

### 1.1.1 Human readable

Probability of positive verification when prover uses secret key and verifier uses corresponding public key will always be 1.

## 1.2 Security

Security is defined via the experiment for a probabilistic polynomial time algorithm  $\mathcal{A}$

**IS security formally**

**Security experiment for passive  $\mathcal{A}$**

The experiment  $\text{Exp}_{\text{IS}}^{\text{PA}, \lambda, \ell}$ :

**Init stage**  $\text{params} \leftarrow \text{ParGen}(1^\lambda)$ ,  $(sk, pk) \leftarrow \text{KeyGen}()$ ,  
 $\mathcal{A}$  given the public key  $pk$ .

**Query stage**  $\mathcal{A}$  runs a polynomial number  $\ell$  of  
 $\pi(\mathcal{P}(sk, pk), \mathcal{V}(pk))$  collecting view  $v^{\mathcal{P}, \mathcal{V}, \ell}$ .

**Impersonation stage**  $\mathcal{A}$  observes the protocol  
 $\pi(\mathcal{A}(pk, v^{\mathcal{P}, \mathcal{V}, \ell}), \mathcal{V}(pk))$ .

In passive mode  $v^{\mathcal{P}, \mathcal{V}, \ell}$  is the transcript  $T = X, c, s$ .

Advantage of adversary  $\text{Adv}(\mathcal{A}, \text{EXP}_\lambda^P) = Pr[\pi(\mathcal{A}(PAR, v = \{T_i\}_{i=1}^l, pk), \mathcal{V}(pk)) = 1]$  We say that the IS is secure if  $\text{Adv}(\mathcal{A}, \text{EXP}_\lambda^P)$  is negligible

### 1.2.1 Passive adversary

Experiment for passive adversary:

1.  $\mathcal{A}$  obtains public key
2.  $\mathcal{A}$  observes the protocol  $n$  times and obtains its transcript
3.  $\mathcal{A}$  runs the protocol as the prover without possession of secret key

Identification Schemes ●●
Security of Signatures ●
Security of Encryption ●
Security of Key Exchange ●●

## Passive adversary against IS

### Adversary advantage

The advantage of  $\mathcal{A}$  in the experiment  $\text{Exp}_{\text{IS}}^{\text{PA}, \lambda, \ell}$  as **probability of acceptance in the impersonation stage**:

$$\text{Adv}(\mathcal{A}, \text{Exp}_{\text{IS}}^{\text{PA}, \lambda, \ell}) = \Pr[\pi(\mathcal{A}(\text{pk}, v^{\mathcal{P}}, \mathcal{V}, \ell), \mathcal{V}(\text{pk})) \rightarrow 1].$$

IS is secure if  $\text{Adv}(\mathcal{A}, \text{Exp}_{\text{IS}}^{\text{PA}, \lambda, \ell})$  negligible in  $\lambda$ .

### Security of identification scheme

$\mathcal{A}$  **probability of acceptance** is negligible in  $\lambda$ .

◀ ▶ ↺ ↻ 🔍 ⌂

Identification Schemes ●●
Security of Signatures ●
Security of Encryption ●
Security of Key Exchange ●●

## Passive adversary - Schnorr IS simulation

### Init Stage

- Set the world of computation  $(g, p, q)$ .
- Let  $(g, g^\alpha)$  is the given instance of DLP in  $(g, p, q)$ .
- Set  $\text{pk} = A = g^\alpha$ .

### Simulation for Query Stage

- Choose  $s', c'$  uniformly at random from  $\mathbb{Z}_q^*$ .
- Compute  $X' = g^{s'} / A^{c'}$ .
- Return  $T' = (X', c', s')$ .

### Transcript indistinguishability

The simulated transcript  $T'$ , and the transcript  $T$  from the real protocol execution, have the same probability distribution.

◀ ▶ ↺ ↻ 🔍 ⌂

Identification Schemes   Security of Signatures   Security of Encryption   Security of Key Exchange

## Passive adversary - Schnorr IS Impersonation Stage

### Rewinding $\mathcal{A}$ in Impersonation Stage

- $\mathcal{A}$  send  $X$ .
- Set virtual breakpoint  $B$ .
- Pass  $c$  to  $\mathcal{A}$ .
- $\mathcal{A}$  send  $s$ .
- Rewind (step-back) to virtual breakpoint  $B$ .
- Pass  $c'$  to  $\mathcal{A}$ .
- $\mathcal{A}$  send  $s'$ .

Navigation icons: back, forward, search, etc. 8/94

Identification Schemes   Security of Signatures   Security of Encryption   Security of Key Exchange

## Passive adversary - Secret Key Extraction

Reduction to DLP

### System of equations

If  $\mathcal{A}$  is accepted in both passes: before and after rewinding:

- $s = x + ac$ .
- $s' = x + ac'$ .
- Thus  $a' = (s - s')/(c - c')$ .

### Conclusions

- With the help of  $\mathcal{A}$  we compute the secret key. But this is against the DLP assumption.
- Thus the assumption of  $\mathcal{A}$  existence is false.

Navigation icons: back, forward, search, etc. 10/94

Human readable

### 1.2.2 Active adversary

## 2 What is a Discrete Logarithm Problem

### 2.1 In human words

Discrete Logarithm Problem is a case when we have a group  $G$ . Assuming that  $G$  is a multiplicative group of prime order  $q$  generated by the generator  $g$  and assuming that there exists a random  $x$  from  $Z_q$  (but we don't know this  $x$ , the probability of finding  $x$  while having value  $g^x$  is negligible.

$$s_k = DL_g p_k$$

## 2.2 Mathematically

Identification Schemes   Security of Signatures   Security of Encryption   Security of Key Exchange

**The underlying assumption: DLP**

**Group setup**

- $G$  is a multiplicative group of prime order  $q$  generated by  $g$ .

**Discrete Logarithm Problem (DLP)**

- For any PPT algorithm  $A_{DL}$  it holds:

$$\Pr[A_{DL}(G, g^x) = x \mid G \leftarrow_{\$} \mathcal{G}(\lambda), x \leftarrow_{\$} \mathbb{Z}_q^*] \leq \epsilon_{DL}(\lambda),$$

where  $\epsilon_{DL}(\lambda)$  is negligible.

Navigation icons: back, forward, search, etc. 7/84

## 3 What is Signature Scheme?

It is a non interactive zero knowledge proof of knowledge of discrete logarithm from the public key. Signature Scheme is a tuple of  $(Init, KeyGen, S, V)$ , where

- $Init$  - produces the space of computation, it takes  $X$  (security parameter) as input and outputs  $PAR$
- $KeyGen$  - key generation function, takes  $PAR$  and outputs a pair of (secret key, public key)
- $S$  - signer, non interactive Turing Machine that signs some message using his secret key and produces the signature
- $V$  - verifier, NITM that takes the message and signature over this message and, by using the private key, verifies whether it was really signed by the signer

### 3.1 Correctness

The signature scheme is correct when probability of

- $PAR \leftarrow Init(\lambda)$
- $(sk, pk) \leftarrow KeyGen(PAR)$
- $\sigma \leftarrow Sign(sk, m)$
- $1 \leftarrow Verify(pk, \sigma, m)$

is equal to 1.

#### 3.1.1 Human readable

The signature scheme is correct when having the secret key and signing the message with this secret key, verification of the signature with the public key corresponding to this secret key, will always hold

## 3.2 Security

The security of signature scheme is defined in such a way that the probability of

- $PAR \leftarrow Init(\lambda)$
- $(sk, pk) \leftarrow KeyGen(PAR)$
- $\mathcal{F}^{O_{sign}}(pk, PAR) \rightarrow (m^*, \sigma^*)$
- $Verify(pk, \sigma^*, m^*) \rightarrow 1$
- $m^* \neq m_0, m_1, \dots, m_l = M^*$

is negligible. ( $\neq \epsilon(\lambda, l)$ )

### 3.2.1 Human readable

We assume that the adversary can initially query a finite number of messages and their signatures. The scheme is secure if the probability that adversary creates a new, different message and its signature  $(m^*, \sigma^*)$  without possession of a secret key and the message is correctly certified is negligible.

## 4 What is security model?

The security model is the tuple of algorithms

## 5 What is the proof of security?

Proofs that there is no better algorithm than brute-force to solve the problem, so the probability of the adversary advantage is negligible.

## 6 What is hash function?

Hash function is any function that takes data of any size as an input and outputs the data of fixed size. The output value is computed randomly and then the pair (in, out) is saved in so called hash table. For the same input, hash function returns the same output all the time. Good hash function is very fast and minimizes the collisions. Every hash value is computed with the same probability. For two similar input, hash function should return two roughly different outputs.

### 6.1 Hash function requirements

- mathematical function - results will always be equal if inputs are equal
- fixed output, inputs and outputs strictly defined (compress function)

### 6.2 Secure hash requirements

- negligible (but non-zero) probability of collision
- small change in input results in large changes of output
- one way function - should not be reversible
- finding input which results in a given hash should not be more efficient than brute-force

## 6.3 ROM - Random Oracle Model

Describes a perfect hash function

- table with mappings (Input, Output) is initially empty
- for each hash query  $H(I)$ :
  - if  $I$  exists in table return corresponding hash value
  - if  $I$  does not exist:
    - generate  $O$  = random value from the output group
    - store the  $(I, O)$  pair in ROM table
    - return  $O$

## 7 Diffie-Hellmann

### 7.1 What is CDH problem?

CDH - Computational Diffie-Hellman

$$A = g^\alpha$$

$$B = g^\beta$$

$$C = g^{\alpha\beta}$$

$$CDH(A, B) = C$$

The slide is titled "Computational Diffie-Hellman problem (CDH)". It features a navigation bar at the top with four items: "Identification Schemes", "Security of Signatures", "Security of Encryption", and "Security of Key Exchange". The main content is divided into two sections. The first section, "Group setup", states that  $G$  is a multiplicative group of prime order  $q$  generated by  $g$ . The second section, "CDH", defines the problem for any PPT algorithm  $\mathcal{A}_{CDH}$ : 
$$\Pr[\mathcal{A}_{CDH}(G, g^x, g^y) = g^{xy} \mid G \leftarrow \mathcal{G}(\lambda), x \leftarrow \mathbb{Z}_q^*, y \leftarrow \mathbb{Z}_q^*] \leq \epsilon_{CDH}(\lambda),$$
 where  $\epsilon_{CDH}(\lambda)$  is negligible. At the bottom right, there are navigation icons and the page number "21/94".

### 7.2 What is DDH problem?

DDH - Decisional Diffie-Hellman

$$A = g^\alpha$$

$$B = g^\beta$$

$$C = g^{\alpha\beta}$$

$$DDH(A, B, C) = 1 \text{ if } C = g^{\alpha\beta} \text{ else } 0$$

Identification Schemes   Security of Signatures   Security of Encryption   Security of Key Exchange

## Decisional Diffie-Hellman problem (DDH)

**DDH**

- Let  $G \leftarrow \mathcal{G}(\lambda)$ ,  $x \leftarrow \mathbb{Z}_q^*$ ,  $y \leftarrow \mathbb{Z}_q^*$ ,  $z \leftarrow \mathbb{Z}_q^*$ ,
- $D_0 = (G, g^x, g^y, g^{xy})$ ,  $D_1 = (G, g^x, g^y, g^z)$ .
- For any polynomial time algorithm  $\mathcal{A}_{\text{DDH}}$ :

$$|\Pr[\mathcal{A}_{\text{DDH}}(D_0) = 0] - \Pr[\mathcal{A}_{\text{DDH}}(D_1) = 0]| \leq \epsilon_{\text{DDH}}(\lambda),$$

where  $\epsilon_{\text{DDH}}(\lambda)$  is negligible.

22/94

### 7.3 What is GDH problem?

GDH - Gap Diffie-Hellman

Identification Schemes   Security of Signatures   Security of Encryption   Security of Key Exchange

## Gap Computational Diffie-Hellman problem (GDH)

**DDH oracle**

Let  $\mathcal{O}_{\text{DDH}}$  denotes the PPT algorithm (so called *Decisional Diffie-Hellman Oracle*) which, for  $G \leftarrow \mathcal{G}(\lambda)$ ,  $x \in \mathbb{Z}_q^*$ ,  $y \in \mathbb{Z}_q^*$ ,  $z \in \mathbb{Z}_q^*$ , computes  $\mathcal{O}_{\text{DDH}}(G, g^x, g^y, g^z) = 1$  if and only if  $z = xy \pmod q$ .

**GDH**

For any probabilistic polynomial time algorithm  $\mathcal{A}_{\text{GDH}}^{\mathcal{O}_{\text{DDH}}}$  having access to Decisional Diffie-Hellman Oracle  $\mathcal{O}_{\text{DDH}}$  it holds:

$$\Pr[\mathcal{A}_{\text{GDH}}^{\mathcal{O}_{\text{DDH}}}(G, g^x, g^y) = g^{xy} \mid G \leftarrow \mathcal{G}(\lambda), x \leftarrow \mathbb{Z}_q^*, y \leftarrow \mathbb{Z}_q^*] \leq \epsilon_{\text{GDH}}(\lambda),$$

where  $\epsilon_{\text{GDH}}(\lambda)$  is negligible.

23/94

## 8 Knowledge proofs

Knowledge proof is a method of proving by one party to another party that one party have knowledge about the certain information without revealing this information and any additional.

### 8.1 What is IZKP?

Interactive zero knowledge proof requires the interaction between two parties, which means they have to communicate at the same time.



## 8.2 What is NIZKP?

Non interactive zero knowledge proof does not require any interaction between both parties. It works in such a way that one party performs its computations at first, then, at any time, the second party can verify the incoming data.

## 9 Goh-Jarecki

TBD

## 10 Pairing

**Pairings**

**Properties**

Let  $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$  be groups with generators  $g_1, g_2, g_T$  respectively. Let  $q = |\langle g_1 \rangle| = |\langle g_2 \rangle| = |\langle g_T \rangle|$ . Let  $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$  be a pairing function with the following properties:

- 1 **Bilinearity**:  $\forall (a, b \in \mathbb{Z}_q, g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2)$ :  
 $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$ .
- 2 **Computability**: Computing  $\hat{e}$  is efficient.
- 3 **Non-degeneracy**:  $\exists (g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2) : \hat{e}(g_1, g_2) \neq 1$ .

If  $\mathbb{G}_1 = \mathbb{G}_2$ , pairing is symmetric. In such a case, we drop lower indices for both group and generator in notation.

## 11 Modified Schnorr

The idea behind the modification is to address the threat that an adversary with the knowledge of  $c$ ,  $s = x + ac$  and a leaked  $x$  can compute a static secret  $a$ . Therefore, instead of sending  $s$  in plain, a prover calculates and sends  $S = g^s$ , hiding  $s$  in the exponent. A new generator  $g$  is computed by hashing  $X$  and  $c$  using function  $H$ , that is,  $g = H(X|c)$ . Even if the ephemeral value  $x$  is leaked, the adversary has to solve the instance of Discrete Logarithm problem to obtain value  $a$  from  $S$ . On the verifier's side, bilinear pairing  $e$  is used to check the linear equation  $s = x + ac$  in the exponent. The equation  $e(S, g) = e(H(X|c), XA^c)$  holds because  $e(H(X|c)^{x+ac}, g) = e(H(X|c), XA^c)$ .

Why the equation holds:

$$e(H(X|c)^{x+ac}, g) = e(H(X|c), XA^c) \quad (1)$$

$$e(H(X|c)^{x+ac}, g) = e(H(X|c), g^x g^{ac}) \quad (2)$$

$$e(H(X|c)^{x+ac}, g) = e(H(X|c), g^{x+ac}) \quad (3)$$

$$e(H(X|c), g)^{x+ac} = e(H(X|c), g)^{x+ac} \quad (4)$$