Coursework

COMS30023: Cryptology

TB1 2020-2021

Objectives and Means

We have studied security definitions and proofs for symmetric cryptography, and the mathematics underlying asymmetric cryptography. In this coursework, we will be studying a hybrid encryption scheme, which combines Diffie-Hellman with a symmetric authenticated encryption to provide public key authenticated encryption. We will study its security all the way from theory to instantiation and to implementation.

In Q1 (p. 4), we consider the syntax, security and correctness of public-key cryptography, including the object of our studies—the Cryptobox scheme. Q2 (p. 8) focuses on the security of Cryptobox as a *public-key authenticated encryption* scheme, discusses our assumptions, and guides you through a proof of security for Cryptobox. In Q3 (p. 13), you will implement a small Cryptobox command-line utility. Finally, Q4 (p. 15) asks you to consider the effect of using weak primitives on the security of the Cryptobox scheme, and asks you to implement attacks against those weak primitives.

The breakdown of marks is as follows:

Question	Points
Public-Key Security and Cryptobox	10
Security of Cryptobox	20
Implementing Cryptobox	20
Reflection and Cryptanalysis	50
Total:	100

Instructions

Submit, by the deadline (at *13:00 on Friday, December 11*), the following two files:

- a PDF document named "`submission-xxxxxxx.pdf`" (where `xxxxxxx` is your student number)containing your answers to theoretical and dissertative questions, as well as any documentation you wish to submit for examination;
- an archive named "`code-xxxxxxx.yyy`" (where `xxxxxxx` is your student number, and `yyy` is some file extension) containing your code solution to Q3.

Your archive must be self-contained (in that any external libraries can be obtained by following simple instructions included in the archive itself) and fully describe all dependencies. We strongly recommend you write and test your instructions to work on the University's lab machines (accessible via SSH): we will use "compiles and runs on the lab machines" as a baseline for quality, so that anything that fails to meet this requirement will not be marked.

You may also submit additional material should you wish to, but we may not consider it in marking.

Support

Support will be provided throughout weeks 8 to 10 in the following ways:

- synchronously (through Teams) on Mondays between 14:00 and 15:00;
- asynchronously through Teams and the BB forum.

We will use the Team and BB forums for COMS30023.

Notations

Throughout this document, we assume the following finite sets:

- a set $\mathcal{K}_s$ of *secret keys*;
- a set $\mathcal{K}_p$ of *public keys*; we often then call the set $\mathcal{K}_p \times \mathcal{K}_s$ the set of *keypairs*;
- a set $\mathcal{K}$ of *symmetric keys*;
- a set $\mathcal{P}$ of *plaintexts*;
- a set $\mathcal{C}$ of *ciphertexts*; and
- a set $\mathcal{N}$ of *nonces*.

We will often consider a distinguished symbol $\perp$, used to denote failures, and used to extend sets. Given a set $\mathcal{X}$, we denote with $\mathcal{X}_\perp$ the set $\mathcal{X} \uplus \{\perp\}$.

If $\mathcal{X}$ is a finite set, we denote with $x \leftarrow\!\!\$\, \mathcal{X}$ the act of sampling a value uniformly at random in $\mathcal{X}$ and storing the sampled value in variable x.

We will assume that the set of plaintexts is equipped with a length function $|\cdot| \in \mathcal{P} \to \mathbb{N}$, and that the set of ciphertexts can be partitioned as $\mathcal{C} = \biguplus_{i \in \mathbb{N}} \mathcal{C}(i)$, where $\mathcal{C}(i)$ contains all ciphertexts that encrypt plaintexts of length i.

Q1 — Public-Key Security and Cryptobox [10 marks]

In this first question, we think about security definitions for public-key constructions, and about our object of study.

1.1 Syntax and Security for Public-Key Encryption and Signing

We first discuss the syntax and security of simple asymmetric cryptographic schemes.

Syntax of asymmetric encryption: An *asymmetric encryption scheme* is a triple of algorithms $\mathcal{E} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$, where

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}()$ is the key generation algorithm, which probabilistically generates keypairs, outputting a public key pk and a secret key sk;

$c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)$ is the encryption algorithm, which probabilistically encrypts a plaintext p under the recipient's public key pk, producing a ciphertext c;

$m \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c)$ is the decryption algorithm, which deterministically decrypts a ciphertext c using the recipient's secret key sk, producing a plaintext p.

Syntax of asymmetric signatures: An *asymmetric signature scheme* is a triple of algorithms $\mathcal{S} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$, where

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}()$ is the key generation algorithm, which probabilistically generates keypairs, outputting a public key pk and a secret key sk;

$c \leftarrow \mathsf{Sign}_{\mathsf{sk}}(m)$ is the signing algorithm, which signs a message m under the signer's secret key sk, producing a signature σ;

$b \leftarrow \mathsf{Vf}_{\mathsf{pk}}(\sigma, m)$ is the verification algorithm, which verifies that a signature σ is valid for message m under public key pk, outputting a boolean ($\top$ if the signature is valid, $\bot$ otherwise).

1.a) [2 marks] Consider the notion shown in Figure 1, which describes security under Chosen Plaintext Attacks. Note, however, that the adversary is not given access to an oracle giving her the ability to obtain encryptions of chosen plaintexts.

```
$\mathsf{CPA}^{\mathrm{real}}_{\mathcal{E},\mathcal{A}}()$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathcal{E}.\mathsf{KGen}$
$m \leftarrow\$ \mathcal{A}(\mathsf{pk})$
$c^* \leftarrow\$ \mathcal{E}.\mathsf{Enc}_{\mathsf{pk}}(m)$
$b \leftarrow\$ \mathcal{A}(c^*)$
```

```
$\mathsf{CPA}^{\mathrm{ideal}}_{\mathcal{E},\mathcal{A}}()()$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathcal{E}.\mathsf{KGen}$
$m \leftarrow\$ \mathcal{A}(\mathsf{pk})$
$c^* \leftarrow\$ \mathcal{C}(|m|)$
$b \leftarrow\$ \mathcal{A}(c^*)$
```

$$\mathsf{Adv}^{\mathrm{cpa}}_{\mathcal{E}}(\mathcal{A}) = \Pr\left[\mathsf{CPA}^{\mathrm{real}}_{\mathcal{E},\mathcal{A}}() : b\right] - \Pr\left[\mathsf{CPA}^{\mathrm{ideal}}_{\mathcal{E},\mathcal{A}}() : b\right]$$

Figure 1: A CPA notion for public-key encryption schemes

Explain why the notion does indeed capture chosen plaintext attacks, relating your explanation to the principles of modern cryptography.

1.b) [3 marks] Security for asymmetric signatures is meant to capture the fact that it should be hard for an adversary to *forge* a signature on a fresh message m that will be accepted as valid for some public key pk without knowing the corresponding secret key sk.

Comparing the definition in Figure 1 to the corresponding symmetric notion, define a security notion for signatures that captures unforgeability—as outlined above—under chosen message attacks. Compare your definition to the most relevant and appropriate symmetric security notion.

Note, in particular, that in the case of signatures, parties who hold the signer's public key should be unable to produce valid signatures (without expending a lot of resources). This differs greatly from the case of symmetric MACs, which provide a weaker form of authentication.

1.2 Public-Key Authenticated Encryption and Cryptobox

We now move to the main notion and construction we consider in this coursework. We wish to construct a cryptographic algorithm that provides the functionality of authenticated encryption, but does not require the sender and recipient to share a symmetric secret in advance.

Encryption will use both the recipient's public key and the sender's private key. Decryption will use both the recipient's private key, and the sender's public key, and will fail if either of the two does not correspond to that used in encryption.

To simplify the study, our encryption will be nonce-based. We consider security against *nonce-respecting* adversaries.

Syntax for nonce-based hybrid encryption schemes: A *nonce-based hybrid encryption schemes* is a triple of algorithms $\mathcal{E} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$, where

$\mathsf{KGen}()$ is the key generation algorithm, which probabilistically generates keypairs, outputting a public key pk and a secret key sk;

$\mathsf{Enc}^{n}_{\mathsf{sk}_s,\mathsf{pk}_r}(m)$ is the encryption algorithm, which deterministically encrypts a plaintext m with nonce n and under the sender's secret key sk_s and the recipient's public key pk_r;

$\mathsf{Dec}^{n}_{\mathsf{pk}_s,\mathsf{sk}_r}(c)$ is the decryption algorithm, which deterministically decrypts a ciphertext c with nonce n and under the sender's public key pk_s and the recipient's secret key sk_r. Decryption may fail with $\perp$.

The Cryptobox **construction:** The Cryptobox construction (see Figure 2) uses a Diffie-Hellman key exchange to generate a shared secret between sender and recipient, and uses it in an symmetric nonce-based authenticated encryption. The construction is thus parameterised by:

- a finite cyclic group $(\mathbb{G}, 1, \cdot)$ of order q and one of its generators g;
- a hash function $\mathsf{H} \in \mathbb{G} \to \mathcal{K}$ that maps group elements to symmetric keys;
- a nonce-based authenticated encryption scheme $\mathcal{E}$.

As is usual when working with multiplicative notation for groups, we will use exponents in $\mathbb{Z}$ to denote repeated group multiplication, so that, for $h \in \mathbb{G}$, and $n \in \mathbb{N}$ we have

$$\begin{array}{rcl} h^n & = & \underbrace{h \cdot \ldots \cdot h}_{n \text{ times}} \\ h^{-n} & = & \left(h^{-1}\right)^n \qquad \text{(with } h^{-1} \text{ the group inverse of } h\text{)} \end{array}$$

Recall that this notation allows standard manipulation of exponents (namely, $h^m \cdot h^n = h^{m+n}$ and $(h^m)^n = h^{mn}$), and that exponents can be taken modulo q. We often omit this in algorithmic descriptions, where it does not matter, but note the potential impact on implementation performance of reducing the exponent.

```
$\mathsf{Cryptobox}_{(\mathbb{G},g,q),\mathsf{H},\mathcal{E}}$

$\mathsf{KGen}()$
    $\mathsf{sk} \leftarrow\$ \mathbb{Z}_q$
    $\mathsf{pk} \leftarrow g^{\mathsf{sk}}$
    return $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{Enc}^n_{\mathsf{sk}_s,\mathsf{pk}_r}(m)$
    $k \leftarrow \mathsf{H}(\mathsf{pk}_r^{\mathsf{sk}_s})$
    $c \leftarrow \mathcal{E}.\mathsf{Enc}^n_k(m)$
    return $c$

$\mathsf{Dec}^n_{\mathsf{pk}_s,\mathsf{sk}_r}(c)$
    $k \leftarrow \mathsf{H}(\mathsf{pk}_s^{\mathsf{sk}_r})$
    $m \leftarrow \mathcal{E}.\mathsf{Dec}^n_k(c)$
    return $m$
```

Figure 2: The Cryptobox scheme

1.c) [3 marks] Assume that the AE scheme $\mathcal{E}$ is correct.

Show that the scheme $\mathsf{Cryptobox}_{(\mathbb{G},g,q),\mathsf{H},\mathcal{E}}$ is *correct*. Formally, show that for all keypairs $\mathsf{pk}_s, \mathsf{sk}_s$ and $\mathsf{pk}_r, \mathsf{sk}_r$ in the range of KGen, for all nonces n and for all messages m, we have

$$\mathsf{Dec}^n_{\mathsf{pk}_s,\mathsf{sk}_r}(\mathsf{Enc}^n_{\mathsf{sk}_s,\mathsf{pk}_r}(m))$$

1.d) [2 marks] What happens if one decrypts with *the same keys* used to encrypt? More formally, what should be the result of the following computation?

$$\mathsf{Dec}^n_{\mathsf{pk}_r,\mathsf{sk}_s}(\mathsf{Enc}^n_{\mathsf{sk}_s,\mathsf{pk}_r}(m))$$

Does Cryptobox behave more like a signature or more like a MAC for authentication? Briefly justify your answer.

Q2 — Security of Cryptobox [20 marks]

We now define the security notion we aim to achieve with Cryptobox, discuss the assumptions we will rely on, and prove security of Cryptobox under those assumptions.

2.1 Assumption: Decisional Diffie-Hellman

In practice, very few cryptographic schemes have security that relies directly on the hardness of the Discrete Logarithm Problem. A common assumption used in security proofs is that the Decisional Diffie-Hellman problem—which consists in distinguishing triples of the form (g^a, g^b, g^{ab}) with uniform a and b from uniformly random triples, and is formally defined below—is hard.

```
$\mathsf{DDH}^{\mathrm{real}}_{(\mathbb{G},g,q),\mathcal{D}}()$
$a \leftarrow\$ \mathbb{Z}_q$
$b \leftarrow\$ \mathbb{Z}_q$
$z \leftarrow ab$
$r \leftarrow\$ \mathcal{D}(g^a, g^b, g^z)$
```

```
$\mathsf{DDH}^{\mathrm{ideal}}_{(\mathbb{G},g,q),\mathcal{D}}()$
$a \leftarrow\$ \mathbb{Z}_q$
$b \leftarrow\$ \mathbb{Z}_q$
$z \leftarrow\$ \mathbb{Z}_q$
$r \leftarrow\$ \mathcal{D}(g^a, g^b, g^z)$
```

$$\mathsf{Adv}^{\mathrm{ddh}}_{\mathbb{G},g,q}(\mathcal{D}) = \Pr\left[\mathsf{DDH}^{\mathrm{real}}_{(\mathbb{G},g,q),\mathcal{D}}() : r\right] - \Pr\left[\mathsf{DDH}^{\mathrm{ideal}}_{(\mathbb{G},g,q),\mathcal{D}}() : r\right]$$

Figure 3: The Decisional Diffie-Hellman problem

More formally, we say that the DDH problem is *hard in a cyclic group* $\mathbb{G}$ if, all efficient distinguishers $\mathcal{D}$ have a small advantage $\mathsf{Adv}^{\mathrm{ddh}}_{\mathbb{G},g,q}(\mathcal{D})$ in distinguishing real DDH tuples from random ones.[1]

[1] Our notation makes explicit the generator and order, however, our terminology does not. The order q is the order of the group and could be left implicit. The choice of generator is less obvious. Fortunately, the hardness of DDH does not depend on the choice of generator: if it is easy to solve DDH in $\mathbb{G}$ with generator g, then it is easy to solve DDH in $\mathbb{G}$ with any other generator $g_0 \neq g$. Indeed, there exists a constant $c \in \mathbb{Z}_q \setminus \{0\}$ (namely $c = \log_{g_0} g$) such that, for any $x \in \mathbb{Z}$, we have $g^x = g_0^{cx}$. Multiplying the exponent by a non-zero constant does not affect the distribution of the group elements we present to the adversary, and the same adversary would be exactly as successful distinguishing with generator g as with generator g_0.

g^a, g^b, g^z can therefore be expressed as $g_0^{a'}$, $g_0^{b'}$ and $g_0^{z'}$ with a' and b' uniform in $\mathbb{Z}_q$, and z' computed from a' and b' as z is from a and b.

2.a) [3 marks] Prove that an adversary that solves the Discrete Logarithm problem in $\mathbb{G}$ can be used to solve the Decisional Diffie-Hellman problem with similar probability and complexity.

2.2 Assumption: Entropy Smoothing hash functions

A useful property for hash functions to have in protocol applications is that they are good at extracting randomness. This is captured by the notion of entropy smoothing, which states that it should be hard for an adversary to distinguish the hash of a uniformly random input from a random output. We define this notion as a pair of games and an advantage, as per our usual style.[2] We specialise the games to hash functions mapping group element to symmetric keys.

```
$\mathsf{ES}^{\mathrm{real}}_{\mathsf{H},\mathcal{D}}()$
    $x \leftarrow\$ \mathbb{G}$
    $h \leftarrow \mathsf{H}(x)$
    $b \leftarrow\$ \mathcal{D}(\mathsf{H}(x))$
```

```
$\mathsf{ES}^{\mathrm{ideal}}_{\mathsf{H},\mathcal{D}}()$
    $x \leftarrow\$ \mathbb{G}$
    $h \leftarrow\$ \mathcal{K}$
    $b \leftarrow\$ \mathcal{D}(h)$
```

$$\mathsf{Adv}^{\mathrm{es}}_{\mathsf{H}}(\mathcal{D}) = \Pr\left[\mathsf{ES}^{\mathrm{real}}_{\mathsf{H},\mathcal{D}}() : b\right] - \Pr\left[\mathsf{ES}^{\mathrm{ideal}}_{\mathsf{H},\mathcal{D}}() : b\right]$$

Figure 4: Entropy-smoothing

A hash function H is entropy-smoothing if all efficient distinguishers $\mathcal{D}$ have a small ES advantage $\mathsf{Adv}^{\mathrm{es}}_{\mathsf{H}}(\mathcal{D})$.

2.3 Goal: Nonce-Based Public-Key Authenticated Encryption

The Cryptobox construction achieves security similar to nonce-based authenticated encryption, but does so without requiring an initial shared secret. We still require that the sender and recipient know and trust each other's public keys—but those are public and easier to share in a verifiable way.

[2] The notion thus defined is not *exactly* the entropy smoothing you will find in the literature. We took liberties in defining hash functions and their security in the lectures and notes, which are now forcing us to take liberties here.

Figure 5 formally defines this notion: the adversary is tasked with distinguishing "real-world" oracles—that correctly encrypt chosen plaintexts (without allowing nonce repetitions) and decrypt chose ciphertexts (as long as they were not produced by the encryption oracle); from "ideal-world" oracles where encryptions are produced by sampling uniformly at random in (a restriction of) the set of ciphertexts, and decryption always fails.

```
$\mathsf{PKAE}^{\mathbb{E},\mathbb{D}}_{\mathcal{E},\mathcal{A}}()$
    $(\mathsf{pk}_s, \mathsf{sk}_s) \leftarrow\$ \mathcal{E}.\mathsf{KGen}$
    $(\mathsf{pk}_r, \mathsf{sk}_r) \leftarrow\$ \mathcal{E}.\mathsf{KGen}$
    $b \leftarrow\$ \mathcal{A}^{\mathbb{E}(\cdot,\cdot),\mathbb{D}(\cdot,\cdot)}(\mathsf{pk}_s, \mathsf{pk}_r)$
```

Real-world oracles

```
$\mathbb{E}^{\text{real}}(n, m)$
    No repeat nonces
    $c \leftarrow \mathcal{E}.\mathsf{Enc}^n_{\mathsf{sk}_s,\mathsf{pk}_r}(m)$
    return $c$
```

```
$\mathbb{D}^{\text{real}}(n, c)$
    No query $c \leftarrow \mathbb{E}(n, c)$
    $m_\perp \leftarrow \mathcal{E}.\mathsf{Dec}^n_{\mathsf{pk}_s,\mathsf{sk}_r}(c)$
    return $m_\perp$
```

Ideal-world oracles

```
$\mathbb{E}^{\text{ideal}}(n, m)$
    No repeat nonces
    $c \leftarrow\$ \mathcal{C}(|m|)$
    return $c$
```

```
$\mathbb{D}^{\text{ideal}}(n, c)$
    No query $c \leftarrow \mathbb{E}(n, c)$
    return $\perp$
```

$$\mathsf{Adv}^{\text{pkae}}_{\mathcal{E}}(\mathcal{A}) = \Pr\left[\mathsf{PKAE}^{\mathbb{E}^{\text{real}},\mathbb{D}^{\text{real}}}_{\mathcal{E},\mathcal{A}}() : b\right] - \Pr\left[\mathsf{PKAE}^{\mathbb{E}^{\text{ideal}},\mathbb{D}^{\text{ideal}}}_{\mathcal{E},\mathcal{A}}() : b\right]$$

Figure 5: A CPA notion for public-key encryption schemes

We say that a nonce-based hybrid encryption scheme $\mathcal{E}$ is a secure non-based public-key authentication encryption scheme if the advantage $\mathsf{Adv}^{\text{pkae}}_{\mathcal{E}}(\mathcal{D})$ of any efficient adversary $\mathcal{D}$ in distinguishing the real world from the ideal world is small.

2.b) [2 marks] Think back to the public-key CPA notion from Figure 1, and Q1.a. Explain why the PKAE security notion shown in Figure 5 *does* make use of an encryption oracle despite presenting a public-key interface and giving public keys to the adversary.

2.4 **Proving** Cryptobox **secure**

In this security proof, we show that any adversary that breaks the PKAE security of Cryptobox has:

i. solved the DDH problem in the underlying group;
ii. distinguished a random digest from the hash of a random input; or
iii. broken the security of the underlying AE scheme.

We do so using a sequence of indistinguishability results, considering two intermediate “games”, one of which you are asked to define.

```
$\mathsf{Exp}_{\mathcal{A}}^{\mathbb{E},\mathbb{D}}()$
$(\mathsf{pk}_s, \mathsf{sk}_s) \leftarrow\!\$\ \mathsf{Cryptobox.KGen}$
$(\mathsf{pk}_r, \mathsf{sk}_r) \leftarrow\!\$\ \mathsf{Cryptobox.KGen}$
$z \leftarrow\!\$\ \mathbb{Z}_q$
$b \leftarrow\!\$\ \mathcal{A}^{\mathbb{E}(\cdot,\cdot),\mathbb{D}(\cdot,\cdot)}(\mathsf{pk}_s, \mathsf{pk}_r)$
```

```
$\mathbb{E}^{\mathrm{ddh}}(n, m)$
No repeat nonces
$k \leftarrow \mathsf{H}(g^z)$
$c \leftarrow \mathcal{E}.\mathsf{Enc}_k^n(m)$
return $c$
```

```
$\mathbb{D}^{\mathrm{ddh}}(n, c)$
No query $c \leftarrow \mathbb{E}(n, c)$
$k \leftarrow \mathsf{H}(g^z)$
$m \leftarrow \mathcal{E}.\mathsf{Dec}_k^n(c)$
return $m$
```

Figure 6: Intermediate experiment

2.c) [3 marks] For this question, consider a fixed group $(\mathbb{G}, g, q)$, a fixed hash function H and a fixed symmetric AE scheme $\mathcal{E}$. We denote with Cryptobox the scheme instantiated with these primitives.

Consider the intermediate experiment Exp and oracles shown in Figure 6. Prove the following lemma by constructing a reduction $\mathcal{B}$ that uses a PKAE adversary $\mathcal{A}$ as a black box to break the DDH assumption.

Lemma 1. *For any PKAE adversary $\mathcal{A}$, there exists a DDH adversary $\mathcal{B}$ that has similar time and query complexity and is such that*

$$\Pr\Big[\mathsf{PKAE}^{\mathbb{E}^{real},\mathbb{D}^{real}}_{\mathsf{Cryptobox},\mathcal{A}}() : b\Big] - \Pr\Big[\mathsf{Exp}^{\mathbb{E}^{ddh},\mathbb{D}^{ddh}}_{\mathcal{A}}() : b\Big] \leq \mathsf{Adv}^{\mathrm{ddh}}_{\mathbb{G},g,q}(\mathcal{B})$$

Hint: Note that the symmetric key used in all queries to $\mathbb{E}^{\mathrm{real}}$ and $\mathbb{D}^{\mathrm{real}}$ is in fact fixed.

2.d) [10 marks] Define a new pair of oracles $\mathbb{E}^{\mathrm{es}}$ and $\mathbb{D}^{\mathrm{es}}$ and finish proving the following concrete security theorem for Cryptobox.

Theorem 1 (Security of Cryptobox). *For any PKAE adversary $\mathcal{A}$, there exist a DDH adversary $\mathcal{B}_{ddh}$, an Entropy Smoothing adversary $\mathcal{B}_{es}$ and an Authenticated Encryption adversary $\mathcal{B}_{ae}$ that have similar time and query complexity (when relevant) to that of $\mathcal{A}$, and are such that*

$$\mathsf{Adv}^{\mathrm{pkae}}_{\mathsf{Cryptobox}_{(\mathbb{G},g,q),\mathsf{H},\mathcal{E}}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathrm{ddh}}_{\mathbb{G},g,q}(\mathcal{B}_{ddh}) + \mathsf{Adv}^{\mathrm{es}}_{\mathsf{H}}(\mathcal{B}_{es}) + \mathsf{Adv}^{\mathrm{ae}}_{\mathcal{E}}(\mathcal{B}_{ae})$$

Hint: You will need to define a new pair of oracles that serves as an intermediate step in the slow idealization of the oracle. (The previous question idealizes the generation of the shared DH secret; the intermediate step you must consider now further idealizes the generation of the symmetric key.) You should be able to reuse the intermediate experiment Exp. Note also that, for any $a, b, c \in \mathbb{R}$, it holds that $a - b + b - c = a - c$.

2.e) [2 marks] Explain—as formally as possible—why the above is sufficient to prove security of Cryptobox when it uses a group in which DDH is hard, an entropy smoothing hash function and a secure authenticated encryption scheme.

Q3 — Implementing Cryptobox [20 marks]

Implement a command-line utility that provides 3 commands:

keygen: the command should take at most one parameter (the security parameter, or key length) and output a newline-separated keypair on standard output, public key first;

encrypt: the command should take at most four parameters (the sender's secret, the recipient's public key, the nonce and the plaintext) and output the ciphertext to standard output;

decrypt: the command should take at most four parameters (the sender's public key, the recipient's secret, the nonce and the ciphertext) and output the raw plaintext to standard output.

Other requirements. Operation failures *must* make use of well-documented return codes, and display helpful error and diagnostic messages on standard error when errors are caused by invalid inputs. The implementation *must* be secure even against an adversary that has access to return codes and error messages.

Nonces and ciphertexts *must* be parsed and printed as hexadecimal-encoded bytestrings by default.

There is no prescribed format for public and private keys: your program can safely assume it will only receive keys produced by its own keygen command without further modification.

Your command-line utility *may* accept additional options (including reading inputs from standard input), which *must* be documented.

Instantiation of cryptographic primitives. Your implementation *must not* use Curve25519 as its Diffie-Hellman group, and *must not* use ChaCha+Poly (ChaCha20 + Poly1305) as AE component.

Apart from the above requirement, you are free to instantiate the cryptographic primitives (Diffie-Hellman group, hash function and Authenticated Encryption scheme) as you see fit. You must also choose an appropriate nonce length.

You *must* argue your choice of DH and AE primitives with respect to the assumptions made in the security proof and your knowledge of cryptanalysis and background research.You *must* argue your choice of nonce length, based on the constraints our security model places on nonces. You do not need to argue your choice of hash function beyond current best practice. Additional consideration may be given to implementations that argue also for the effect of their choices on performance.

Other considerations You are free to use libraries, and any language you choose, but we *must* be able to compile and run your code on the University lab machines, following any simple instructions you give us—ideally simply typing ‘`make`’.

As per the University’s policies, you retain all copyright on the code you write for this coursework. However, we request that you please refrain from making it publicly accessible or share it with anyone until the results have been published.

Marking criteria Your code will be assessed based on functionality and compliance with the given functional-security requirements, as well as on the quality of the code and its documentation. The documentation should justify how the security requirements are met.

In particular, your choice of primitives will be assessed based on its compliance with the given functional-security requirements (and those that arise from the security proof), as well as the relevance and quality of the discussions surrounding that choice (including consideration of alternatives).

Vivas during Week 12 or during the exam period may be used to further examine particular aspects of the implementation.

Q4 — Reflection and Cryptanalysis [50 marks]

We now reflect over some of the choices made in the proof, focusing in particular on the assumptions, and consider some generic attacks on some of the primitives used in Cryptobox.

```
$\mathsf{wODH}^{\mathrm{real}}_{(\mathbb{G},g,q),\mathsf{H},\mathcal{A}}()$
$u \leftarrow\$ \mathbb{Z}_q$
$v \leftarrow\$ \mathbb{Z}_q$
$z \leftarrow uv$
$b \leftarrow\$ \mathcal{A}(g^u, g^v, \mathsf{H}(g^z))$
```

```
$\mathsf{wODH}^{\mathrm{ideal}}_{(\mathbb{G},g,q),\mathsf{H},\mathcal{A}}()$
$u \leftarrow\$ \mathbb{Z}_q$
$v \leftarrow\$ \mathbb{Z}_q$
$z \leftarrow\$ \mathbb{Z}_q$
$b \leftarrow\$ \mathcal{A}(g^u, g^v, \mathsf{H}(g^z))$
```

$$\mathsf{Adv}^{\mathrm{wodh}}_{(\mathbb{G},g,q),\mathsf{H}}(\mathcal{A}) = \Pr\left[\mathsf{wODH}^{\mathrm{real}}_{(\mathbb{G},g,q),\mathsf{H},\mathcal{A}}() : b\right] - \Pr\left[\mathsf{wODH}^{\mathrm{ideal}}_{(\mathbb{G},g,q),\mathsf{H},\mathcal{A}}() : b\right]$$

Figure 7: A weak version of the Oracle Diffie-Hellman assumption.

4.a) [7 marks] Consider the (weak) Oracle Diffie-Hellman (wODH) assumption, defined by the experiments shown in Figure 7.[3] Compare it to the combination of the DDH and Entropy Smoothing assumptions we used in Q2, considering in particular their relative hardness, their relative usefulness in proving Cryptobox secure, and whether you believe they apply to your particular choice of instantiation. (*Max.* 500 words.)

4.b) In this part we consider the effect of known cryptanalysis techniques on the choices for $(\mathbb{G}, g, q)$.

i. [14 marks] Let $p = 1492993$ and define

$$\mathbb{F}_{p^2} = \{\alpha x + \beta : \alpha, \beta \in \mathbb{Z}_p, x^2 + 192168x + 7 \equiv 0 \pmod{p}\}.$$

Define $g = x$; then g generates $\mathbb{F}_{p^2}$ as a multiplicative group (you do not have to prove this).

By implementing (an) algorithm(s) of your choice, compute the discrete logarithm $a = \log_g(g^a)$ of

$$g^a = 1119782x + 54140.$$

[3]The actual ODH assumption also gives the adversary access to an oracle that computes $\mathsf{H}(g^{vw})$ for any adversarial input $w \neq u$.

ii. [2 marks] How would your attack scale to cryptographic-size parameters, e.g. for $p^2 \approx 2^{3000}$?

iii. [5 marks] Give a choice of p and $\mathbb{G} = \mathbb{F}_{p^2}^*$, with p at least 64 bits, for which index calculus would *not* be the best known attack on the discrete logarithm problem in $\mathbb{G}$. Justify your choice. How would you search for parameters for which index calculus would be the best known attack?

iv. Let $\mathbb{G} = \mathbb{F}_{2^9}^*$, where

$$\mathbb{F}_{2^9} = \left\{ \sum_{i=0}^{8} a_i x^i : a_i \in \mathbb{Z}_2,\ x^9 + x^4 + 1 \equiv 0 \pmod 2 \right\}$$

Let $g = x$; then g generates $\mathbb{F}_{2^9}^*$ as a multiplicative group (you do not have to prove this).

α) [10 marks] Using index calculus with a factor base of

$$\{x+1, x^4+x+1, x^2+x+1\},$$

compute $a \pmod{2^9-1}$ such that $g^a = x^4 + x$.

β) [2 marks] What would a good alternative choice of factor base be for this example?

v. [2 marks] How would you improve the generic index calculus attack in the case of binary fields (that is fields of size 2^n for some positive integer n)?

4.c) We now briefly discuss preimage attacks on hash functions.

i. [6 marks] Find a bitstring whose SHA256 hash *ends* with the bitstring whose hexadecimal encoding is your student number. Give a hexadecimal encoding of the bitstring and explain your approach. We expect code to be written, but it does not have to be submitted. We may ask you to explain your code to us in discussions.

ii. [2 marks] Consider Cryptobox. What can you say about its security if built with a hash function that is vulnerable to preimage attacks?

Closing: boilerplate information

Deadline. The deadline for submission of all optional unit assignments is 13:00 on Friday 11th of December. Students should submit all required materials to the "Assessment, submission and feedback" section of Blackboard - it is essential that this is done on the Blackboard page related to the "With Coursework" variant of the unit.

Time commitment. The expectation is that students will spend 3 full working weeks on their two assignments. The effort spent on the assignment for each unit should be approximately equal, being roughly equivalent to 1.5 working weeks each.

Academic Offences. Academic offences (including submission of work that is not your own, falsification of data/evidence or the use of materials without appropriate referencing) are all taken very seriously by the University. Suspected offences will be dealt with in accordance with the University's policies and procedures. If an academic offence is suspected in your work, you will be asked to attend an interview with senior members of the school, where you will be given the opportunity to defend your work. The plagiarism panel are able to apply a range of penalties, depending the severity of the offence. These include: requirement to resubmit work, capping of grades and the award of no mark for an element of assessment.

Extenuating circumstances. If the completion of your assignment has been significantly disrupted by serious health conditions, personal problems, periods of quarantine, or other similar issues, you may be able to apply for consideration of extenuating circumstances (in accordance with the normal university policy and processes). Students should apply for consideration of extenuating circumstances as soon as possible when the problem occurs, using the online form.[4]

You should note however that extensions are not possible for optional unit assignments. If your application for extenuating circumstances is successful, it is most likely that you will be required to retake the assessment of the unit at the next available opportunity.

Implications of UK "travel window" The UK Government will be instigating a "travel window" to allow students return home from University once UK national restrictions have been lifted. This will take place between the 3rd and 9th of December and as such, will occur during the optional unit assignment period. Students whose work is significantly impacted by extended periods of travel and quarantine during this period should apply for extenuating circumstances. As discussed previously, extensions are not possible for optional unit assignments. If your application for extenuating circumstances is successful, it is most likely that you will be required to retake the assessment of the unit at the next available opportunity.

[4] `https://apps.powerapps.com/play/3172b943-0956-4b88-bf3d-3f37871d1170?tenantId=b2e47f30-cd7d-4a4e-a5da-b18cf1a4151b`