Formal Analysis of Non-Malleability for Commitments in EasyCrypt

Denis Firsov[1,2][0000−0003−1267−7898], Sven Laur[3][0000−0002−9891−3347], and Ekaterina Zhuchko[2,3]

[1] Guardtime, Tallinn, Estonia
denis.firsov@guardtime.com
[2] Tallinn University of Technology, Tallinn, Estonia
ekzhuc@ttu.ee
[3] Tartu University, Tartu, Estonia
swen@math.ut.ee

Abstract. In this work, we perform a formal analysis of definitions of non-malleability for commitment schemes in the EasyCrypt theorem prover. There are two distinct formulations of non-malleability found in the literature: the comparison-based definition and the simulation-based definition. In this paper, we do a formal analysis of both. We start by formally proving that the comparison-based definition which was originally introduced by Laur et al. is unsatisfiable. Also, we propose a novel formulation of simulation-based non-malleability and show that it is satisfiable in the Random Oracle Model. Moreover, we validate our definition by proving that it implies hiding and binding of the commitment scheme. Finally, we relate the novel definition to the existing definitions of non-malleability.

Keywords: cryptography · commitments · non-malleability · formal methods · EasyCrypt.

1 Introduction

A commitment scheme is one of the fundamental primitives in cryptography. Intuitively, we can think of a commitment as a locked box containing a message. Only the sender who produced the commitment knows the secret opening key which can unlock the box and reveal the message. The sender can send this box to a receiver and then at a later stage give him the opening key to unlock it.

The most fundamental security properties of commitments are hiding and binding. We say that a commitment is hiding if an adversary is unable to see the message without the opening key (the box which contains the message should not be transparent). We say that a commitment is binding if, once the sender committed to a message and sent the commitment to the receiver, the sender cannot open the commitment to a different message (the box should not have any secret backdoors or double bottoms). But these properties do not prevent all of the attacks and most notably the "man-in-the-middle" attacks.

1.1 Motivation: Non-Malleability of Commitments

The non-malleability property aims to protect commitments against man-in-the-middle attacks. In such an attack, we have Mallory who is an active adversary between two parties: Alice and Bob. Let's assume that Alice sends a commitment c of a message m to Bob. However, all of their communication goes through the man-in-the-middle adversary Mallory who can modify the commitment or simply not deliver it. The goal of Mallory is to generate a commitment c' to another message m' which is non-trivially related to the original message m.[4]

A classical motivating example where non-malleability would be needed is that of a blind auction. Consider an auction where participants bid for an item by publishing commitments to their bids. At the end, bidders open their commitments and the highest bid wins. If the commitment scheme is malleable, an adversary could participate in the auction by posting for each of the other bids a commitment to a bid that is only one dollar higher. In this case, the adversary would have an unfair advantage. Moreover, the adversary has no need to learn the exact amounts that other bidders have placed. The goal of non-malleability definitions is to prevent these types of attacks.

It should also be noted that the described scenario suggests that non-malleability must be a stronger security property than both hiding and binding. Indeed, if the commitments are not hiding, the bidding adversary can see the contents and carry out the same type of attack. If the commitments are not binding, the bidding adversary can commit to one bid and open to another.

There have been several attempts to formally define non-malleability of commitments. Most notably, Crescenzo et al. presented a simulation-based definition [3]. The main idea of their definition is to compare the success probability of an adversary and its simulator. The adversary sees a commitment c of a message m and must produce a commitment c' of a message m' which must be non-trivially related to m. At the same time, the simulator must also produce a message similarly related to m, but without seeing any of the derivatives of m (e.g., commitment on m). If the difference between success probabilities is negligible then the commitment scheme is considered simulation-based non-malleable.

Later, Laur et al. introduced a new formulation of non-malleability which is now known as *comparison-based definition* [8,10,9,6]. The goal of this definition is to phrase non-malleability without referring to a simulator. This was motivated by the fact that definitions formulated in terms of simulators are more complicated to falsify by presenting a specially programmed adversary.

1.2 Discussion

The field of cryptography has rapidly grown in its complexity and subsequently faced a crisis in producing correct proofs. The security guarantees for cryptographic protocols usually come in the form of pen-and-paper proofs. Formalizing

[4] An example of a non-trivial relation could be that the message m' is the same as m except all occurrences of *"PAY TO: Alice"* are replaced with *"PAY TO: Mallory"*.

the intuition behind cryptographic security proofs is not a straightforward task as there could be hidden assumptions and informal reasoning that can easily be forgotten by the author and overlooked by the reader. We believe that formal methods could be used in cryptography to address these problems. For example, EasyCrypt [2] is a theorem prover which was specifically developed for the purpose of verifying cryptographic protocols. The original intention of this paper was to analyze the comparison based non-malleability of commitments introduced by Laur et al. [8]. However, after we started our formal analysis and specified the definition precisely, we were able to conjecture and then prove that the original definition of comparison-based non-malleability is unsatisfiable. Next, we decided to verify satisfiability of simulation-based definitions [4,3,1]. In fact, we were able to express a novel definition of simulation-based non-malleability and prove that it is satisfiable and is also stronger than the notions known from the literature.

The paper is structured as follows: in Sec. 2, we formalize comparison-based non-malleability of Laur et al. [8] and prove that it is unsatisfiable. In Sec. 3, we introduce a novel simulation-based non-malleability definition and analyze its properties. More specifically, in Sec. 3.1, we prove that the novel simulation-based non-malleability implies hiding and binding of the commitment scheme. In Sec. 3.2, we prove that the novel definition is satisfiable in the Random Oracle Model. In Sec. 3.3, we relate the novel formulation of simulation-based non-malleability to the previous ones known from the literature. In Sec. 4, we conclude and present possible research directions for future work.

Our results are formalized in the EasyCrypt theorem prover and the proof-scripts can be found in the supplementary material [5].

2 Comparison-Based Non-Malleability

In the standard library of EasyCrypt, commitments are defined as programs (modules in the EasyCrypt parlance) with a predetermined interface.

Definition 1 (Commitment Scheme). *A commitment scheme C is a module that has efficiently computable procedures with the following functionality:*

- *$gen()$: generates public keys (also known as public parameters) of a commitment scheme.*
- *$commit(pk, m)$: takes a public key pk and a message m as its parameters and returns the pair of a commitment value c and an opening key d (also known as a certificate).*
- *$verify(pk, m, c, d)$: verifies the commitment c on the message m with respect to the opening key d.*

In our formalization, we assume that we are working with commitment schemes without an internal state. More specifically, the procedure $C.gen$ must be implementable as an effective distribution of public keys. The procedure $C.commit$ must be implementable as a pure function with explicit argument for a random string. The procedure $C.verify$ must be implementable as a pure deterministic

function. However, in Sec. 3.2 we make an exception to this rule when we define the commitment scheme C_n^k based on the Lazy Random Oracle which efficiently models a truly random function and as the result relies on internal state.

As the next step, we formalize the comparison-based non-malleability definition by Laur et al. [8].

Definition 2 (Laur et al.). *A commitment scheme C is **(comparison-based) non-malleable** iff for any adversary A, the advantage $AdvC(C, A)$ is negligible, where*

$$AdvC(C, A) := |\mathsf{Pr}\,[r \leftarrow GN_0(C, A).main() : r = 1] - \mathsf{Pr}\,[r \leftarrow GN_1(C, A).main() : r = 1]\,|.$$

```
1: module GN_0(C, A)
2:     fun main() = {
3:         pk ← C.gen()
4:         M ← A.init(pk)
5:         m ←$ M
6:         (c, d) ← C.commit(pk, m)
7:         (c', R) ← A.commit(c)
8:         (d', m') ← A.decommit(d)
9:         v ← C.verify(pk, m', c', d')
10:        return v ∧ R(m, m') ∧ c ≠ c'
11:    }
12: end
```

```
1: module GN_1(C, A)
2:     fun main() = {
3:         pk ← C.gen()
4:         M ← A.init(pk)
5:         m ←$ M; n ←$ M
6:         (c, d) ← C.commit(pk, m)
7:         (c', R) ← A.commit(c)
8:         (d', m') ← A.decommit(d)
9:         v ← C.verify(pk, m', c', d')
10:        return v ∧ R(n, m') ∧ c ≠ c'
11:    }
12: end
```

(For simplicity of presentation, in Def. 2 the adversary computes a single commitment c' while in the original definition of Laur et al. the adversary was allowed to return n commitments and $n+1$-place relation R. In our EasyCrypt formalization, we work with the original definition, but in the paper we show the simplified version since this detail is irrelevant for the main unsatifiability result.)

Both games are parameterized by a commitment scheme C and an adversary A. In the game GN_0, A is given the public key pk and is asked to compute a message distribution $\mathcal{M}$. A message m is then sampled from $\mathcal{M}$ and a commitment-opening pair (c, d) is computed with respect to m. Next, A is given the commitment c and asked to produce a commitment c' and a relation R. After that, A is given the opening d and asked to produce an opening-message pair (c', d'). The adversary wins the game if the pair (c', d') is valid with respect to m', the relation R is satisfied by a pair (m, m') and A's commitment c' is different from c. The only difference in the game GN_1 is that a second message n is sampled from the message distribution (independently from m). The commitment-opening pair is still computed with respect to the message m, but the winning condition of GN_1 considers whether $R(n, m')$ holds (line 10).

The adversary's overall advantage is defined in terms of its ability to distinguish between games GN_0 and GN_1. In other words, A has to win one game and lose the other in order to increase the advantage. This means that to be

successful, the adversary has to find the exact relation R which will hold given the pair (m, m') and will not hold given the pair (n, m'), or vice versa.

2.1 Unsatisfiability of the Comparison-Based Definition

In this section, we show that Def. 2 is not satisfiable by any realistic commitment scheme.[5] More specifically, we construct a single adversary which can break the comparison-based non-malleability of any (realistic) commitment scheme with unacceptably high probability.

Theorem 1. *There exists an adversary A such that for any commitment scheme C the comparison-based non-malleability advantage of A is as follows:*

$$AdvC(C, A) = \frac{1}{4} - \frac{1}{4} \cdot \mathsf{Pr}\left[\begin{array}{c} pk \leftarrow C.gen();\ (c, d) \leftarrow C.commit(pk, 0); \\ (c', d') \leftarrow C.commit(pk, 0) : c = c' \end{array}\right].$$

(Here, it is enough to assume that commitments generated by C are sufficiently random to make $AdvC(C, A)$ close to $\frac{1}{4}$.)

Proof. The adversary A is defined as follows: in the initialization phase, the adversary returns a uniform distribution of booleans. During the commit phase, c' is fixed to be a commitment on $m' = 0$. Moreover, the relation $R(m, m')$ is also fixed and will only hold true if $m = 0$ and $m' = 0$. During the "decommit" phase, A checks if c was indeed a commitment on message $m = 0$ and if so, returns $(0, d')$ as the message-opening pair. If the verification fails, the adversary intentionally loses the game (denoted by $\perp$). In order to calculate the adversary's advantage,

```
 1: module A(C)
 2:     var pk, c, d'
 3:     fun init(pk) := { return {0,1} }        ▷ {0,1} is a uniform distribution of bits
 4:     fun commit(c) = {
 5:         (c', d') ← C.commit(pk, 0)
 6:         R ← λm0 m1.m0 = 0 ∧ m1 = 0
 7:         return (R, c')
 8:     }
 9:     fun decommit(d) = {
10:         if C.verify(pk, 0, c, d) then
11:             return (d', 0)
12:         end if
13:         return ⊥                  ▷ denotes a pair which always fails the verification
14:     }
15: end
```

[5] We assume that in realistic schemes commitment values contain a sufficient amount of randomness.

we inline A into the games GN_0 and GN_1 and argue as follows:

$$
\begin{aligned}
&\mathsf{Pr}\left[\begin{array}{c} pk \leftarrow C.gen();\ mdistr \leftarrow A.init(pk);\ m \xleftarrow{\$} mdistr; \\ (c,d) \leftarrow C.commit(pk,m);\ (c',d') \leftarrow C.commit(pk,0); \\ (d',0) \leftarrow A.decommit(d) : \\ C.verify(pk,0,c',d'), m = 0, c \neq c' \end{array}\right] \\
-&\mathsf{Pr}\left[\begin{array}{c} pk \leftarrow C.gen();\ mdistr \leftarrow A.init(pk);\ m \xleftarrow{\$} mdistr;\ n \xleftarrow{\$} mdistr; \\ (c,d) \leftarrow C.commit(pk,m);\ (c',d') \leftarrow C.commit(pk,0); \\ (d',0) \leftarrow A.decommit(d) : \\ C.verify(pk,0,c',d'), m = 0, n = 0, c \neq c' \end{array}\right] \\
\overset{(1)}{=}\ &(\mathsf{Pr}\,[\,GN_0(A).main() : m = 0\,] \\
&-\ \mathsf{Pr}\,[\,GN_0(A).main() : m = 0, c = c'\,]) \\
-\ &(\mathsf{Pr}\,[\,GN_1(A).main() : m = 0, n = 0\,] \\
&-\ \mathsf{Pr}\,[\,GN_1(A).main() : m = 0, n = 0, c = c'\,]) \\
\overset{(2)}{=}\ &\frac{1}{2} - \mathsf{Pr}\,[\,GN_0(A).main() : m = 0, c = c'\,] \\
-\ &\frac{1}{4} + \frac{1}{2} \cdot \mathsf{Pr}\,[\,GN_0(A).main() : m = 0, c = c'\,] \\
\overset{(3)}{=}\ &\frac{1}{4} - \frac{1}{2} \cdot \mathsf{Pr}\left[\begin{array}{c} pk \leftarrow C.gen();\ mdistr \leftarrow A.init(pk);\ m \xleftarrow{\$} \{0,1\}; \\ (c,d) \leftarrow C.commit(pk,m);\ (c',d') \leftarrow C.commit(pk,0) : \\ m = 0, c = c' \end{array}\right] \\
\overset{(4)}{=}\ &\frac{1}{4} - \frac{1}{4} \cdot \mathsf{Pr}\left[\begin{array}{c} pk \leftarrow C.gen();\ (c,d) \leftarrow C.commit(pk,0); \\ (c',d') \leftarrow C.commit(pk,0) : c = c' \end{array}\right].
\end{aligned}
$$

In step (1), we observe that for any sound scheme the commitment verification (i.e., $C.verify(pk, 0, c', d') = 1$) is guaranteed to succeed. Also, we rewrite the winning probability in terms of an event complement to the $c \neq c'$ condition. In step (2), we can restate all the probabilities in relation to GN_0 by observing that n is independent from m and making explicit the probability of sampling $n = 0$ as a coefficient. In step (3), we compute the probabilities and inline the game GN_0. In step (4), we observe that the remaining probabiltiy expression is non-zero only when $m = 0$, so we can simplify the game further.

Observe that for any message m, the following probability can be safely assumed to be negligible for any realistic commitment scheme:

$$
\mathsf{Pr}\left[\begin{array}{c} pk \leftarrow C.gen();\ (c,d) \leftarrow C.commit(pk,m); \\ (c',d') \leftarrow C.commit(pk,m) : c = c' \end{array}\right].
$$

The fully formal derivation can be found in the file `CNM_unsat.ec` of the accompanying development.

The reason why the adversary A is able to have a non-negligible advantage is because it could "intentionally lose" in the decommit phase. Once it receives the opening d, it can easily verify the content of the given commitment c and if verification fails, intentionally lose the game. Finally, we find it interesting that

this analysis shows that the comparison-based definition cannot be instantiated with any realistic commitment scheme, but could be proved for some paradoxical schemes: for example, the perfectly hiding and completely non-binding "constant"-commitment scheme from Sec. 3.3 satisfies Def. 2.

3 Simulation-Based Non-Malleability

In this section, we introduce a novel definition of simulation-based non-malleability which is inspired by the previously discussed comparison-based definition and existing simulation-based definitions. The strong sides of the novel formulation is that it is provably stronger than existing definitions, it implies hiding and biding of a commitment scheme, and it is satisfiable in the Random Oracle Model.

Definition 3 (Sim-NM). *A commitment scheme C is **(simulation-based) non-malleable** iff for any adversary A there exists a simulator S so that for any advice string h the advantage $AdvS(C, A, S, h)$ is negligible, where*

$$AdvS(C, A, S, h) := \mathsf{Pr}\left[r \leftarrow SG_0(C, A).main(h) : r = 1\right] - \mathsf{Pr}\left[r \leftarrow SG_1(C, A, S).main(h) : r = 1\right].$$

```
1: module SG_0(C, A)
2:     fun main(h : advice) = {
3:         pk ← C.gen()
4:         (M, R) ← A.init(pk, h)
5:         m ←$ M
6:         (c, d) ← C.commit(pk, m)
7:         c' ← A.commit(pk, c)
8:         (d', m') ← A.decommit(d)
9:         v ← C.verify(pk, m', c', d')
10:        return R(m, m')
11:            ∧ (c, d) ≠ (c', d') ∧ v
12:    }
13: end
```

```
1: module SG_1(C, A, S)
2:     fun main(h : advice) = {
3:         pk ← C.gen()
4:         (M, R) ← A.init(pk, h)
5:         m ←$ M
6:
7:
8:         m' ← S.run(h, pk, M, R)
9:
10:        return R(m, m')
11:
12:    }
13: end
```

The game SG_0 is parameterized by a commitment scheme C, an adversary A, and an advice string h (e.g. it can encode the history of previous runs). In game SG_0, the adversary computes a distribution $\mathcal{M}$ and relation R based on the public key and the advice. Next, we sample a message m from $\mathcal{M}$ and compute the commitment-opening pair (c, d) on the message m using the commitment scheme C. Next, the adversary must produce a commitment c' given c as the parameter. Then, we reveal the opening d to the adversary and it must produce a message m' and an opening d'. The adversary wins the game if the relation R is satisfied by the pair (m, m'), the tuple (c', d') is a valid commitment-opening pair with respect to the message m', and the adversary's commitment-opening pair (c', d') is different from (c, d).

The game SG_1 is parameterized by a commitment scheme C, an adversary A, a simulator S, and an advice string h. The game SG_1 starts similarly to SG_0, namely, the adversary A generates a distribution $\mathcal{M}$ and a relation R based on the public key and the advice. Next, we sample a message m. Finally, the simulator receives the advice, the public key, the message distribution, and the relation as its arguments and must produce a message m'. The simulator wins the game if the relation R is satisfied by the pair (m, m').

Note that in SG_1 the message m is independent from m'. This aspect makes this definition simpler to justify. Indeed, simulation-based non-malleability claims that the adversary A is not getting any non-negligible advantage from observing a commitment and opening of m as compared to the simulator which is able to satisfy the same relation R without ever seeing any derivatives of m.

3.1 Hiding and Binding

In this section, we observe that the simulation-based non-malleability introduced in Def. 3 is strong enough to imply hiding and binding of the commitment scheme. We think that this fact is a good validation of the novel definition of non-malleability. Indeed, in Sec. 1.1 we argued that the goal of non-malleability definitions is to prevent man-in-the-middle attacks which strongly suggests that non-malleability must be a stronger security property than the basic requirements of hiding and binding. To the best of our knowledge, Def. 3 is the first formulation of non-malleability for commitments which implies hiding and binding.

Definition 4 (Binding). *We define the binding advantage of an adversary A with respect to the commitment scheme C as the probability that A computes two valid openings for the same commitment and two distinct messages:*

$$AdvB(C, A) \quad := \quad \mathsf{Pr}\left[\begin{array}{c} pk \leftarrow C.gen(); (c, m_1, d_1, m_2, d_2) \leftarrow A.bind(pk); \\ v_1 \leftarrow C.verify(pk, c, m_1, d_1); \\ v_2 \leftarrow C.verify(pk, c, m_2, d_2) : \\ v_1 = 1, v_2 = 1, m_1 \neq m_2 \end{array}\right].$$

A commitment scheme is binding iff for any efficient adversary the binding advantage is negligible.

To express the relation between non-malleability and binding, we will need an extra property which we call the "unpredictability" of the commitment-opening pair. Intuitively, a commitment scheme is unpredictable if the commitment-opening pair is sufficiently random. We express this by saying that a commitment scheme is unpredictable if the adversary cannot guess the canonically generated commitment-opening pair for a message chosen by the adversary.

Definition 5 (Unpredictability). *We define the unpredictability advantage of an adversary A with respect to the commitment scheme C as the probability that A guesses the commitment-opening pair for the message of its choice:*

$$AdvU(C, A) := \mathsf{Pr}\left[\begin{array}{c} pk \leftarrow C.gen();\ (m, L) \leftarrow A.guess(pk); \\ (c, d) \leftarrow C.commit(pk, m) : (c, d) \in L \end{array}\right].$$

A commitment scheme is unpredictable iff for any efficient adversary A the unpredictability advantage is negligible.

Next, we state a lemma which relates simulation-based non-malleability and binding of a commitment scheme.

Lemma 1 (Sim-NM $\implies$ Binding). *For any commitment scheme C and any adversary A there exists a non-malleability adversary A' and an unpredictability adversary A'' so that for any simulator S the binding advantage of A is upper bounded as follows:*

$$AdvB(C, A) \leq 2 \cdot AdvS(C, A', S, \epsilon) + 6 \cdot AdvU(C, A'').$$

(Here, ϵ is the empty advice string.)

The EasyCrypt proof can be found in the file `NSNM_Pure_Binding.ec` of the accompanying development. Below, we sketch the main idea of the proof. The adversaries A' and A'' are constructed from A. In the initialization phase ($A'.init$ method), we call $A.bind$ to get the tuple (c, m_1, d_1, m_2, d_2) and return the pair $(\mathcal{M}, R)$ as the result, where $\mathcal{M}$ is the uniform distribution of messages m_1 and m_2, and R is the equality relation. If A wins the binding game (i.e., (c, m_1, d_1, m_2, d_2) satisfies the binding experiment) then in the $A'.commit$ phase we return the commitment c; otherwise, we return a canonically generated commitment for m_1. In the $A'.decommit(d)$ phase, if A wins the binding game and d opens m_1, we return d_2 (similarly for m_2, *mutatis mutandis*). Then we show that if A wins the binding game then A' wins SG_0. However, if A fails the binding game then with probability $\frac{1}{2}$ the adversary A' still wins the game SG_0. At the same time, any simulator S can win the game $SG_1(C, A', S, \epsilon)$ with probability not bigger than $\frac{1}{2}$. The unpredictability term arises from the probability that the game SG_0 and the adversary A' produce equal commitment-opening pairs.

Next, we address the hiding property. We say that a commitment scheme is hiding if any efficient adversary cannot distinguish between commitments generated for messages of its choice.

Definition 6 (Hiding). *We define a module $HidingExperiment$ parameterized by a commitment scheme C, an adversary A, and a boolean b:*

```
1: module HidingExperiment(C, A)
2:     fun main(b : bool) = {
3:         pk ← C.gen()
4:         (m0, m1) ← A.choose(pk)
5:         (c, d) ← C.commit(pk, mb)
6:         r ← A.unhide(c)
7:         return r
8:     }
9: end
```

In the module definition we refer to the message m_b, where b is the parameter of the main procedure. The hiding advantage of the adversary A with respect to the scheme C is defined as follows:

$$\begin{aligned} AdvH(C,A) := |\mathsf{Pr}\,[r \leftarrow HidingExperiment(C,A).main(1) : r = 1] \\ - \mathsf{Pr}\,[r \leftarrow HidingExperiment(C,A).main(0) : r = 1]|. \end{aligned}$$

A commitment scheme is hiding iff for any efficient adversary the hiding advantage is negligible.

Lemma 2 (Sim-NM $\implies$ Hiding). *For any commitment scheme C and any adversary A there exists a non-malleability adversary A' and an unpredictability adversary A'' so that for any simulator S the hiding advantage of A is upper bounded as follows:*

$$AdvH(C,A) \leq 2 \cdot AdvS(C,A',S,\epsilon) + 2 \cdot AdvU(C,A'').$$

The EasyCrypt proof can be found in the file `NSNM_Pure_Hiding.ec` of the accompanying development. The main idea of the proof is as follows. Let us assume that A is a hiding adversary such that (the converse case is symmetric):

$$\begin{aligned} \mathsf{Pr}\,[r \leftarrow HidingExperiment(C,A).main(1) : r = 1] \\ \leq \mathsf{Pr}\,[r \leftarrow HidingExperiment(C,A).main(0) : r = 1]\,. \end{aligned}$$

We construct the adversary A' as follows: in the initialization phase ($A'.init$ method), we call $A.choose$ to get the tuple (m_1, m_2) and return as a result the pair $(\mathcal{M}, R)$, where $\mathcal{M}$ is the uniform distribution of messages m_1 and m_2, and R is the equality relation. In the $A'.commit$ phase, we call $A.unhide(c)$ and get the bit b. If $b = 0$ then we compute and return a commitment for m_1, otherwise for m_2. In the $A'.decommit$ phase, we return the opening computed in the previous phase. Finally, we show that if A guesses correctly then A' wins SG_0. At the same time, any simulator S can win the game $SG_1(C, A', S)$ with probability not bigger than $\frac{1}{2}$. As before, the unpredictability term arises from the probability that the game SG_0 and the adversary A' produce equal commitment-opening pairs.

3.2 Construction in the Random Oracle Model

In this section, we define a commitment scheme based on the Lazy Random Oracle (LRO) and prove that it satisfies simulation-based non-malleability introduced in Def. 3. This means that we assume the existence of a truly random function and give oracle-access to this function to all parties involved in the process, including all the components of the commitment scheme and the adversary. Intuitively, a random oracle models an ideal hash function.

Given input queries the LRO must produce "random" outputs. However, it has to be consistent and respond with the same output if it is queried on the same input more than once. To achieve that, LRO keeps a mapping of input queries to random values which are sampled and stored upon every fresh query.

Definition 7 (Lazy Random Oracle). *We implement a lazy random oracle as a module LRO^n. The module is parameterized by a natural n which determines the size of the output bitstrings. The module is stateful – it stores the variable $LRO^n.m$ which is a (finite) mapping from input queries to bitstrings of length n. The procedure $LRO^n.init$ initializes the variable $LRO^n.m$ with an empty mapping. The procedure $LRO^n.o(x)$ checks if the provided argument x is not already in the domain of $LRO^n.m$ in which case it updates the mapping by associating the value x with a randomly sampled bitstring r of length n. Finally, the value associated with x is returned.*

```
 1: module LRO^n
 2:     var m : (in_t, bits)fmap                    ▷ in_t is a data type of input queries
 3:     fun init() = { LRO^n.m ← empty }
 4:     fun o(x : in_t) = {
 5:         r ←$ {0,1}^n                            ▷ uniform distribution of all n-bit strings
 6:         if x ∉_D LRO^n.m then
 7:             LRO^n.m.[x] ← r
 8:         end if
 9:         return LRO^n.m.[x]
10:     }
11: end
```

In the following, we present a standard implementation of an LRO-based commitment scheme C_n^k (here, k and n are security parameters: k determines the size of the opening key, and n is the security parameter of the LRO).

Definition 8 (LRO-Commitment Scheme). *The commitment scheme C_n^k is implemented as follows:*

- *$gen()$: initializes the LRO oracle. The scheme C_n^k does not need a public key, so the gen procedure simply returns the element of a singleton type (denoted by tt).*
- *$commit(pk, m)$: samples an opening key d from the uniform distribution of k-bit strings; computes commitment c as the result of a call to the oracle $LRO^n.o(m, d)$; returns (c, d) as the resulting commitment-opening pair.*
- *$verify(pk, m, c, d)$: verifies the commitment c of the message m with respect to the opening d by checking that $LRO^n.o(m, d)$ call returns c.*

The commitment scheme C_n^k is sound. More specifically, the commitment-opening pairs which are generated on messages verify on these messages. However, this is only true if adversaries do not re-initialize the LRO. In EasyCrypt, we can specify the set of adversaries who cannot directly write to the variables of the LRO^n module or call the procedures $LRO^n.init$ and $C_n^k.gen$.

Theorem 2 (C_n^k is Sim-NM). *C_n^k is a non-malleable commitment scheme with respect to Def. 3. More specifically, for any adversary A who is doing at most q*

```
 1: module C_n^k
 2:     fun gen() = { LRO^n.init(); return tt }
 3:     fun commit(pk, m) = {
 4:         d <$- {0,1}^k
 5:         c <- LRO^n.o(m, d)
 6:         return (c, d)
 7:     }
 8:     fun verify(pk, m, c, d) = {
 9:         return LRO^n.o(m, d) = c
10:     }
11: end
```

queries to the random oracle LRO^n there is a simulator S, so that for any advice string h the adversary's advantage is as follows:

$$AdvS(C_n^k, A, S, h) \leq \frac{2q^2}{2^n} + \frac{q}{2^n} + \frac{q}{2^k}.$$

The EasyCrypt proof of the theorem is in the file `NSNM_ROM_Construction.ec` of the accompanying development. Here, we only outline its main idea. By proving the properties of the LRO we show that for any adversary A the probability of winning the game SG_0 could only be negligibly larger than the probability of the following event:

1. The adversary A wins the game SG_0.
2. At the end of the game, the mapping $LRO^n.m$ contains no duplicates in its range (i.e., two different values x and y, so that $LRO^n.m.[x] = LRO^n.m.[y]$).
3. The commitment value c' returned by the adversary is different from the value c generated by the game SG_0.
4. After the $A.commit$ call (line 7) the variable $LRO^n.m$ contains a value c' in its range.

The described event allows us to construct a simulator S which calls $A.commit(r)$ with some uniformly sampled n-bit string r. Then we can show that with overwhelming probability the adversary A is not going to be able to distinguish between the commit call made by the simulator who provides a parameter r versus the honestly generated commitment c as in the game SG_0. Moreover, the event (1)–(4) ensures that after the call $A.commit(r)$ with the random r, the simulator can extract the unique message-opening pair (m', d') associated to c' in the mapping $LRO^n.m$. This is important since the simulator S does not know which message m was sampled in the beginning of the game SG_1, and hence cannot continue the simulation of SG_0 game by calling $A.decommit(m)$. Overall, this strategy ensures that whenever A wins the game SG_0, the simulator wins the game SG_1 (modulo negligible error).

3.3 Related Notions

In this section, we review two related definitions of simulation-based non-malleability which appear in the literature [3,1].

Definition 9 (Crescenzo et al.). *A commitment scheme C is non-malleable iff for any adversary A there exists a simulator S so that for any relation R and distribution $\mathcal{M}$ the adversary's non-malleability advantage is negligible, where*

$$AdvCS(C, A, S, R, \mathcal{M}) := \Pr\left[r \leftarrow SGC_0(C, A).main(R, \mathcal{M}) : r = 1\right] - \Pr\left[r \leftarrow SGC_1(S).main(R, \mathcal{M}) : r = 1\right].$$

```
1: module SGC_0(C, A)
2:     fun main(R, M) = {
3:         m <$- M
4:         pk ← C.gen()
5:         (c, d) ← C.commit(pk, m)
6:         c' ← A.commit(pk, c)
7:         (d', m') ← A.decommit(d)
8:         v ← C.verify(pk, m', c', d')
9:         return v ∧ R(m, m') ∧ c ≠ c'
10:    }
11: end
```

```
1: module SGC_1(S)
2:     fun main(R, M) = {
3:         m <$- M
4:
5:
6:
7:
8:         m' ← S.run(R, M)
9:         return R(m, m')
10:    }
11: end
```

The main difference between Def. 3 and Def. 9 is that in the first definition we let the adversary to specify the message distribution $\mathcal{M}$ and relation R while in the second definition both are universally quantified parameters. Another important difference is in the winning condition of the adversary's game. More specifically, in SG_0, the adversary wins if it generates a commitment-opening pair (c', d') which is different from the canonically generated pair (c, d). On the other hand, the game SGC_0 requires a stronger condition, namely, $c \neq c'$. Unfortunately, this makes it possible to prove that the "constant"-commitment scheme is non-malleable according to Def. 9, where "constant"-commitment scheme is defined as follows:

```
1: module ConstComm
2:     fun gen() = { return tt }
3:     fun commit(pk, m) = { return (tt, m) }
4:     fun verify(pk, m, c, d) = { return m = d ∧ c = tt }
5: end
```

For any message, the commitment is a constant value (denoted by tt). The opening for a message m is the message m itself. The verification algorithm checks that the commitment is tt and that the message and its opening are the same.

It is easy to see that *ConstComm* is a perfectly hiding and non-binding commitment scheme. Unfortunately, *ConstComm* can be shown to be non-malleable with respect to Def. 9.

Lemma 3. *The "constant"-commitment scheme ConstComm is non-malleable with respect to Def. 9. More specifically, for any adversary A, simulator S, distribution $\mathcal{M}$, and relation R, the advantage is as follows:*

$$AdvCS(ConstComm, A, S, R, \mathcal{M}) \leq 0.$$

The lemma is proved by observing that the adversary's winning condition $c \neq c'$ is never satisfied, hence, no adversary will win the game SGC_0 (see the file `ConstComm.ec`).

In [1], Arita presents a similar non-malleability definition which addresses the problem of the previous definition by adjusting the winning condition.

Definition 10 (Arita). *A commitment scheme C is non-malleable iff for any adversary A there exists a simulator S, so that for any message distribution $\mathcal{M}$ and an antireflexive relation R the following difference is negligible:*

$$\begin{aligned} AdvAS(C, A, S, R, \mathcal{M}) := {} & \mathsf{Pr}\left[r \leftarrow SGA_0(A, C).main(R, \mathcal{M}) : r = 1\right] \\ & - \mathsf{Pr}\left[r \leftarrow SGA_1(S).main(R, \mathcal{M}) : r = 1\right], \text{where } R \text{ is antireflexive.} \end{aligned}$$

(Recall that R is antireflexive iff $\forall a, b : R(a, b) \implies a \neq b$.)

```
1: module SGA_0(C, A)
2:     fun main(R, M) = {
3:         m <$- M
4:         pk <- C.gen()
5:         (c, d) <- C.commit(pk, m)
6:         c' <- A.commit(pk, c)
7:         (d', m') <- A.decommit(d)
8:         v <- C.verify(pk, m', c', d')
9:         return v /\ R(m, m')
10:    }
11: end
```

```
1: module SGA_1(S)
2:     fun main(R, M) = {
3:         m <$- M
4:
5:
6:
7:
8:         m' <- S.run(R, M)
9:         return R(m, m')
10:    }
11: end
```

The main difference of Arita's definition as compared to the previous one is that the winning condition $c \neq c'$ is replaced with the condition of antireflexivity on the relation R. (This effectively means that the condition $c \neq c'$ is replaced with the condition $m \neq m'$.) This has the effect that the "constant"-commitment scheme *ConstComm* can now be proved malleable.

Lemma 4. *The "constant"-commitment scheme ConstComm is malleable with respect to Def. 10. More specifically, let $\mathcal{M}$ be the uniform distribution over booleans and $R(m, m')$ be the relation satisfied iff $m \neq m'$. Then there exists an adversary A so that for any simulator S the adversary's advantage is as follows:*

$$AdvAS(ConstComm, A, S, R, \mathcal{M}) = \frac{1}{2}.$$

(See the file `ConstComm.ec` for details.)

Also, we have preliminary results indicating that one can conclude the hiding and binding properties from Arita's definition, but the security level drops proportionally to the size of the message space. This happens because the distribution $\mathcal{M}$ and relation R could not be computed during the execution of a game, but need to be specified as parameters. This, in turn, makes it impossible to guarantee that the messages m_1 and m_2 (returned by $A.choose$ and $A.bind$ in the binding and hiding games, respectively) satisfy the relation R sufficiently often.

Finally, it is easy to see that both related definitions of non-malleability are weaker than the one introduced in Def. 3.

Lemma 5. *If a commitment scheme is non-malleable with respect to Def. 3 then it is also non-malleable with respect to Def. 9 and Def. 10.*

The proof only requires simple transformations of the SGC_0 and SGA_0 adversaries which add the initialization procedures necessary for SG_0 which return the respective message distribution $\mathcal{M}$ and the relation R. (See the file `NSNM_Related.ec` for details.)

4 Conclusions

The problem of inadequate definitions in cryptography is not new [7]. The errors in definitions may take many years to be discovered and the impact of these errors can range from a minimal nuisance to an actual threat that can be realised as an attack in the real world.

In the beginning of our investigation, we were surprised to find the definition of comparison-based non-malleability unsatisfiable. The paper [8] radiates confidence of the authors that their definition is not only satisfiable, but that some constructions provide unreasonably high level of security. Moreover, the paper is well-cited. However, according to our best knowledge, we are the first to spot the mistake. We attribute our discovery of unsatisfiability to the fact that our investigation was carried out in the formal setting of the EasyCrypt theorem prover. Although the idea behind the proof is fairly simple, the formal derivation took considerable effort.

We also argued that it is desirable for a non-malleability definition to imply hiding and biding properties of commitments. Unfortunately, we were not able to find any such definition in the literature. We continued by proposing a novel simulation-based definition and showed that it implies hiding and binding. We also provided a simple construction in the Random Oracle Model that satisfies the proposed definition. On top of this, we have demonstrated that our novel definition is stronger than the previous simulation-based definitions of non-malleability.

Finally, this work stresses the need to provide higher assurance to the cryptographic security proofs. We believe that formal methods provide a solution which ensures rigor necessary for the mission critical systems.

In the future, we plan to investigate the applications of non-malleability of commitments to timestamping services.

References

1. S. Arita, *A straight-line extractable non-malleable commitment scheme*, IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **90-A** (2007), 1384–1394.
2. Gilles Barthe, Benjamin Gregoire, Sylvain Heraud, and Santiago Béguelin, *Computer-aided security proofs for the working cryptographer*, vol. 6841, 08 2011, pp. 71–90.
3. Giovanni Di Crescenzo, Jonathan Katz, Rafail Ostrovsky, and Adam Smith, *Efficient and non-interactive non-malleable commitment*, Cryptology ePrint Archive, Report 2001/032, 2001, `https://ia.cr/2001/032`.
4. Danny Dolev, Cynthia Dwork, and Moni Naor, *Nonmalleable cryptography*, SIAM Review **45** (2003), no. 4, 727–784.
5. Denis Firsov and Ekaterina Zhuchko, *Formal analysis of non-malleability for commitments in EasyCrypt*, `https://github.com/dfirsov/commitments-non-malleability-ec/tree/v.1.0.0`, 2022.
6. Sameh Khalfaoui, Jean Leneutre, Arthur Villard, Jingxuan Ma, and Pascal Urien, *Security analysis of out-of-band device pairing protocols: A survey*, Wireless Communications and Mobile Computing **2021** (2021), 1–30.
7. Neal Koblitz and Alfred Menezes, *Critical perspectives on provable security: Fifteen years of "another look" papers*, Advances in Mathematics of Communications **13** (2019), 517–558.
8. Sven Laur and Kaisa Nyberg, *Efficient mutual data authentication using manually authenticated strings*, International Conference on Cryptology and Network Security, Springer, 2006, pp. 90–107.
9. Ming Li and et al., *Secure ad-hoc trust initialization and key management in wireless body area networks*, ACM TRANS. SENSOR NETW (2012).
10. Shahab Mirzadeh, Haitham Cruickshank, and Rahim Tafazolli, *Secure device pairing: A survey*, IEEE Communications Surveys Tutorials **16** (2014), no. 1, 17–40.