Subversion-Resistant Commitment Schemes: Definitions and Constructions

Karim Baghery

University of Tartu, Estonia
karim.baghery@ut.ee

Abstract. A commitment scheme allows a committer to create a commitment to a secret value, and later may open and reveal the secret value in a verifiable manner. In the common reference string model, commitment schemes require a setup phase which is supposed to be done by a third trusted party or distributed authority. During last few years, various news are reported about subversion of *trusted* setup phase in mass-surveillance activities; strictly speaking about commitment schemes, recently it was discovered that the SwissPost-Scytl mix-net uses a trapdoor commitment scheme, that allows undetectably altering the votes once you know the trapdoor [Hae19,LPT19]. Motivated by such news and recent studies on subversion-resistance of various cryptographic primitives, this research studies security of commitment schemes in the presence of a maliciously chosen public commitment key. To attain a clear understanding of achievable security, we present a variation of current definitions called subversion hiding, subversion equivocality and subversion binding. Then we provide both negative and positive results on constructing subversion-resistant commitment schemes, by showing that some combinations of notions are not compatible, while presenting subversion-resistant constructions that can achieve other combinations.

Keywords: Commitment schemes, subversion security, common reference string model

1 Introduction

The notion of commitment [Blu81] is one of the fundamental and widely used concepts in cryptography. Basically, a commitment scheme allows a committer to create a commitment c to a secret value m. But, it may later open the commitment c and reveal the value m (and a randomness) to prove that c was the commitment to the value m [GQ88,Ped92]. The procedure of generating c is called *committing* phase, and revealing message m and some secret information used in committing phase (precisely, randomnesses) called *opening* phase. In the common reference string model, commitment schemes require a *setup* phase that is done by a trusted third party [CIO98], and it is shown that when we have a trusted setup phase, one-way functions are sufficient to construct non-interactive commitment schemes [Nao91,CIO98,HILL99]. During last decades, we have seen

various elegant non-interactive commitment schemes that are deployed as a sub-protocol in wide range of cryptographic applications, to refer some, such as contract signing [EGL85], multi-party computation [GMW87], zero-knowledge proofs [GMW91,Dam90], commit-and-proof systems [GS08,Lip16,DGP+19], e-voting [Gro05,Wik09], shuffle arguments [GL07,Wik09,FLZ16], blockchains and their by-products (e.g. cryptocurrencies [BCG+14,FMMO18] and smart contracts [KMS+16]), and many other sensitive practical applications.

On the security of setup phase. Along with developing various cryptographic primitives in sensitive applications, recently there have been various attacks or flaw reports on setup phase of cryptographic systems that rely on public parameters supposed to be generated honestly. In some cases, attacks are caused from maliciously (or incorrectly) generated public parameters or modifying cryptographic protocol specifications to embed backdoors, with intent to violate security of the main system [BBG+13,PLS13,Gre14,Gab19,LPT19,Hae19]. Particularly about commitment schemes, recently two research [Hae19,LPT19] independently discovered that the implementation of shuffle argument in the SwissPost-Scytl mix-net uses a trapdoor commitment scheme, which allows to break security of the main system without being detected. Clearly speaking, in their case the used commitment scheme has a trapdoor that having access to that, one can alter the votes but also can produce an acceptable shuffle proof. So, given such trapdoor, a malicious party can does an undetectable vote manipulation by an authority who sets up the mixing network[1].

To deal with such concerns, a particular subfield of cryptography was proposed which is known as kleptography [YY96,YY97,Sim83,Sim85,RTYZ16]. Intuitively, in this subfield, an adversary is allowed to replace a cryptographic algorithm with an altered version with intend of subverting its security. To construct practical systems secure against such adversary, we need to develop cryptography that can guarantee security against parameter subversion. Meaning that new cryptographic systems should provide their pre-defined security with trusted parameters, but even the parameters subverted, the system still can guarantee a level of security. As a known technique to mitigate the trust on public parameters, some researches have studied setup corruption or malicious parameter generation with MPC approaches [GO07,GGJS11,KKZZ14]. Initiated by Bellare et al. [BPR14], recently this direction has gotten considerable attention with focus on different cryptographic primitives including symmetric encryption schemes [BPR14], signature schemes [AMV15], non-interactive zero-knowledge proofs [BFS16], and public-key encryption schemes [ABK18]. Each of above researches consider achievable security in a particular cryptographic primitive under subverted public parameters. Non-interactive commitment schemes are another prominent family of cryptographic primitives that their public commitment keys are assumed to be generated honestly by a trusted third party [Ped92,DF02,GOS06,Gro09,Gro10,Lip12]. As such commitment schemes are deployed in various areas of cryptography, so their security is not only im-

[1] More details in `https://people.eng.unimelb.edu.au/vjteague/SwissVote` and `https://e-voting.bfh.ch/publications/2019/`

portant on itself, but also security of other practical systems relies on it (e.g. guaranteeing security of shuffling in mix-net of SwissPost-Scytl). Thus their security under subverting public commitment key can have a crucial effect on security of complete system.

As the main contribution, this paper considers resistance of non-interactive commitment schemes in the case of subverting public commitment key, and provides a treatment with presenting definitions, negative results, and some constructions as positive results.

As already mentioned, commitment schemes in the common reference string model require a *public commitment key*, a.k.a. public parameters, that is supposed to be generated honestly by a third party and publicly shared among committers and verifiers. Basically, committers and verifiers have to trust the setup phase. To mitigate the trust on setup phase, an alternative is to use multi-party computation (MPC) protocols [GO07,GGJS11,KKZZ14]. But in general, the question of what happens if public commitment key is generated maliciously, has got little attention. In many practical applications, by default end-users admit that a trusted third party will generate the public commitment key and will publicly share it with all users. Beside the fact that in many cases finding a publicly-accepted trusted party is difficult, a natural question can be what would happen if the public commitment key will be generated maliciously? From a committer perspective, can we still achieve the expected security guarantees if the trusted party colludes with the verifier? Similarly, from a verifier's point of view, what would happen if a malicious key generator colludes with the committer, such that the committer will have access to secret information of setup phase. Actually these are the main questions that we address in this research.

First of all, in order to get clear understanding of achievable security in a commitment scheme with subverted setup, we present new variations of current definitions. We call them subversion-resistant definitions, as they are defined to guarantee security of committers and verifiers even if the setup phase of commitment scheme is subverted. Presenting subversion-resistant definitions for commitment schemes is the first contribution of this paper. Before discussing security requirements in subversion-resistant commitment schemes, we first recall the standard requirements where both committer and verifier trust that the public commitment key is generated honestly.

On the security of commitment schemes. An equivocal non-interactive commitment scheme Π_{com} can be considered as a tuple of algorithms $\Pi_{com} = (\mathsf{KGen}, \mathsf{Com}_{\mathsf{ck}}, \mathsf{OpenVer}_{\mathsf{ck}}, \mathsf{Sim.KGen}, \mathsf{Com}^*_{\mathsf{tk}}, \mathsf{Equiv}_{\mathsf{tk}})$. In this model, we consider three known requirements for commitment schemes which are described below.
Hiding: It is hard for any adversary $\mathcal{A}$, given an honestly generated public commitment key ck (by KGen), to generate two messages m_0, m_1 from message space $\mathcal{M}_{\mathsf{ck}}$ such that $\mathcal{A}$ can distinguish between their corresponding commitments c_0 and c_1 where $(c_0, \mathsf{op}_0) \leftarrow \mathsf{Com}_{\mathsf{ck}}(m_0)$ and $(c_1, \mathsf{op}_1) \leftarrow \mathsf{Com}_{\mathsf{ck}}(m_1)$.
Binding: It is hard for any adversary $\mathcal{A}$, given an honestly generated public commitment key ck, to come up with a collision $(c, m_0, \mathsf{op}_0, m_1, \mathsf{op}_1)$, such that op_0 and op_1 are valid opening values of two different pre-images $m_0 \neq m_1$ for c.

Equivocality: Given a secret trapdoor tk associated with the public commitment key ck, it is possible to create a fake commitment that can be opened successfully. More formally, a commitment scheme is (perfectly) equivocal (a.k.a. trapdoor commitment) if for all stateful adversary $\mathcal{A}$, and security parameter λ, we have, $\Pr[\mathsf{ck} \leftarrow \mathsf{KGen}(1^\lambda), m \leftarrow \mathcal{A}(\mathsf{ck}), r \leftarrow_\$ \mathcal{R}_{\mathsf{ck}}, (c, \mathsf{op}) \leftarrow \mathsf{Com}_{\mathsf{ck}}(m; r) : \mathcal{A}(c, \mathsf{op}) = 1] = \Pr[(\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{Sim.KGen}(1^\lambda), m \leftarrow \mathcal{A}(\mathsf{ck}), (c, \eta) \leftarrow \mathsf{Com}^*_{\mathsf{tk}}(\mathsf{ck}), \mathsf{op} \leftarrow \mathsf{Equiv}_{\mathsf{tk}}(c, m, \eta) : \mathcal{A}(c, \mathsf{op}) = 1]$, where $\mathcal{A}$ outputs $m \in \mathcal{M}_{\mathsf{ck}}$, and $\mathcal{R}_{\mathsf{ck}}$ denotes the randomness space.

We note that equivocality implies that the commitment scheme is hiding, as a commitment is indistinguishable from an equivocal commitment which can be opened to any message. In some cases the commitment scheme is only expected to satisfy *hiding* and *binding*, but in many cases, specially in security proofs of larger cryptographic systems which use commitment schemes as a subroutine, it is shown that hiding property is not enough and the scheme should guarantee equivocality as well. Pedersen commitment scheme is one of the most known schemes that satisfies hiding, equivocality and binding [Ped92].

Next, we discuss security goals in subversion-resistant commitment scheme.

On the security of commitment schemes under parameter subversion. The key change in new definitions is that the adversary generates the public commitment key ck. We note that if adversary generates ck, so it can retain some trapdoors tk as a "backdoor" related to the commitment key ck. In section 3.2, we formalize the following requirements for subversion-resistant commitment schemes:

Sub-hiding: (subversion hiding) Even if the adversary $\mathcal{A}$ generates public commitment key ck (consequently it knows the corresponding trapdoor tk), if the ck is *well-formed* [1], it is hard for $\mathcal{A}$ to generate two messages $m_0, m_1 \in \mathcal{M}_{\mathsf{ck}}$ such that $\mathcal{A}$ can distinguish between their corresponding commitments c_0 and c_1 where $(c_0, \mathsf{op}_0) \leftarrow \mathsf{Com}_{\mathsf{ck}}(m_0)$ and $(c_1, \mathsf{op}_1) \leftarrow \mathsf{Com}_{\mathsf{ck}}(m_1)$.

Sub-binding: (subversion binding) Even if the adversary $\mathcal{A}$ generates public commitment key ck (consequently it knows the corresponding trapdoor tk), if the ck is *well-formed*, it is hard for $\mathcal{A}$ to come up with a collision $(c, m_0, \mathsf{op}_0, m_1, \mathsf{op}_1)$, such that op_0 and op_1 are valid opening values of two different pre-images $m_0 \neq m_1$ for commitment c.

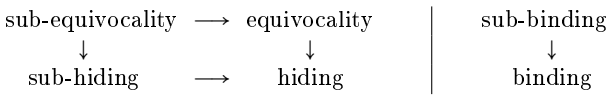
Sub-equivocality: (subversion equivocality) Even if the adversary $\mathcal{A}$ generates public commitment key ck (consequently it knows the corresponding trapdoor tk), the commitment scheme *still* should be equivocal; meaning that the setup phase *still* should be simulatable. Technically speaking, for all stateful adversary $\mathcal{A}$ we should have, $\Pr[(\mathsf{ck}, m) \leftarrow \mathcal{A}(1^\lambda), r \leftarrow_\$ \mathcal{R}_{\mathsf{ck}}, (c, \mathsf{op}) \leftarrow \mathsf{Com}_{\mathsf{ck}}(m; r) : \mathcal{A}(c, \mathsf{op}) = 1]$ is equal to $\Pr[((\mathsf{ck}, \mathsf{tk}), m) \leftarrow \mathsf{Sim}.\mathcal{A}(1^\lambda), (c, \eta) \leftarrow \mathsf{Com}^*_{\mathsf{tk}}(\mathsf{ck}), \mathsf{op} \leftarrow \mathsf{Equiv}_{\mathsf{tk}}(c, m, \eta) : \mathcal{A}(c, \mathsf{op}) = 1]$, where $\mathcal{A}$ outputs $m \in \mathcal{M}_{\mathsf{ck}}$, and $\mathcal{R}_{\mathsf{ck}}$ denotes the randomness space.

We note that subversion resistant version of definitions, imply standard version of them. For instance, sub-equivocality implies equivocality, and as already

[1] Intuitively, the generated ck should have well-defined structure. Later we will have more discussion about verifying the well-formedness of a public commitment key.

Table 1: **Top:** Table of our negative and positive results. In each row we refer to simultaneously achieving all selected notions. **Bottom:** Relation between current and subversion-resistant notions.

	Standard			Subversion Resistant			
	hiding	equivocal	binding	sub-hiding	sub-equivocal	sub-binding	result in
negative		✓				✓	Thm. 1
positive 1	✓	✓	✓	✓	✓		Thm. 2
positive 2	✓		✓	✓		✓	Thm. 3
positive 3	✓	✓	✓	✓			Thm. 4



mentioned equivocality implies hiding, as a commitment is indistinguishable from an equivocal commitment that can be opened to any message. A representation of relations between standard and new notions is shown in bottom of Tab. 1. As it can be seen, sub-binding implies binding, and sub-equivocality implies sub-hiding, equivocality, and consequently hiding.

Achievability of new notions. By considering new notions, a natural question is how much subversion security is achievable in commitment schemes? As often we need security guarantee simultaneously for both involved parties in a commitment scheme; for instance guaranteeing (sub-)equivocality or (sub-)hiding for committer and sub-binding for verifier. One may notice that, any of notions sub-hiding, sub-equivocality and sub-binding individually are achievable. For instance, sub-binding can be achieved if the committer sends the secret value m, but this is not hiding. Similarly, in a commitment scheme if the committer sends empty string and the verifier always accepts, the scheme will achieve sub-equivocality but the scheme would not achieve binding. Such trivial constructions are less practical and they are not considered in this work. Our focus is on achieving practically-interested combinations of the current and new notions. Clearly speaking, currently there exist commitment schemes achieving *binding*, *hiding* and *equivocality*. So, the main target is to lift the current schemes or construct new ones that extra from current security guarantees, they will achieve one or few of new notions: sub-binding, sub-hiding and sub-equivocality. For instance, an important question is "can we achieve *all* X and sub-X notions, for $X \in \{\text{hiding, binding, equivocality}\}$, at the same time?"; or both committer and verifier have to trust setup phase absolutely, and we cannot achieve any of sub-X notions along with current ones!

In this paper, we answer the above questions by categorizing which combinations are achievable and which ones not. It is shown that the answer for the mentioned important question is negative! But the situation is not so bad and we still can achieve some of new notions while keeping the current ones. Such cases imply that in some schemes a committer or verifier can mitigate or ignore the trust on setup phase. We summarize our key results in Tab. 1. Each row of the table, considers constructing commitment schemes that simultaneously

can achieve the indicated notions (by checkmark, ✓). The colours light red and green show the negative and positive results, respectively. Last column lists the theorems that establish the results.

Negative result. We first ask whether we can achieve sub-binding (binding with a subverted commitment key) along with the current notions, namely *sub-binding*, binding, hiding, and equivocality. The negative result (first row in results of Tab. 1) indicates that we cannot achieve even standard equivocality and sub-binding at the same time. This implies there is no commitment scheme in the common reference string model which can achieve both equivocality and sub-binding at the same time. We should emphasize that here we consider the standard notion of equivocality, where the public commitment key ck is generated honestly by a trusted third party. Clearly this implies that we cannot achieve sub-equivocality and sub-binding at the same time. Indeed this result also answers negatively to the important question asked before. In section 4, in proof of Thm. 1 establishing this result, we will observe that definitions of equivocality and sub-binding are not compatible and one can use algorithms $\mathsf{Com}^*_{\mathsf{tk}}$ and $\mathsf{Equiv}_{\mathsf{tk}}$ in the definition of equivocality to break the sub-binding.

Positive results. In the rest we describe the listed positive results in Tab. 1.

Positive 1: By considering the negative result one concludes that we cannot construct sub-binding and equivocal (and sub-equivocal) non-interactive commitment schemes. So, the next best scenario would be the case that one can achieve all notions but sub-binding. In Thm. 2 we show that actually this case is possible, and this gives the first positive result in Tab. 1. The proof of Thm. 2, relies on the Bilinear Diffie-Hellman Knowledge of Exponents (BDH-KE) assumption (defined in Def. 3) in a group equipped with bilinear map [2].

Positive 2: After the first positive result, one may ask if there exists any practical commitment scheme that can achieve sub-binding. We already know from the negative result which sub-binding cannot be achieved by equivocality (and sub-equivocality). So by considering this, the best scenario can be constructing commitment schemes which can achieve all notions except equivocality and sub-equivocality. Second positive result in Tab. 1 shows actually we can construct such commitment schemes. This result is shown in Thm. 3, which is established under some standard assumptions. We will observe that the result follows from the existence of binding and hiding commitment scheme with *trivial* setup phase.

Positive 3: The third positive result in Tab. 1 is a commonly used case in practice. The commitment scheme already satisfies hiding, equivocality and binding under a honestly generated commitment public key. When we consider the case that public commitment keys are generated maliciously, it does not break completely, and indeed it still achieves hiding. In practice, this can provide a layer of protection for a committer without trusting a third party. In subsection 5.3 we will show with minimal checking, the known commitment schemes proposed by

[2] Intuitively, BDH-KE assumption states given bilinier groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with generators g_1 and g_2, if an adversary can come up with g_1^x and g_2^x, it must know x [Dam92]

Pedersen [Ped92] can achieve this case. This result might look redundant, as it is a restricted form of the set considered in first positive result. But we will observe that unlike the first positive result which is achieved under a knowledge assumption (non-falsifiable), this result is established under some standard assumptions. Indeed already there exist some commitment schemes achieving hiding, equivocality and binding, and without adding extra assumption with minimal checking we can achieve sub-hiding which is stronger notion than standard hiding.

On the achievability of all combinations and other notions. The main under focus question in this paper is for $X \in \{\text{hiding, binding, equivocality}\}$, which combinations of X and sub-X are achievable at the same time. In Tab. 1, we only talked about four cases from 2^6 cases which one may think of. Our considered cases, which are more practically interested ones, cover many of those cases but still one can use a similar approach and go through over all combinations and evaluate achievability of each one. For instance, by considering relations between variations in Tab. 1, one may notice several trivial cases, and more importantly the negative result covers a big set of cases which are impossible to achieve.

From a different perspective, in our analysis, we focused only on three notions hiding, equivocality and binding. There are several other notions about commitment schemes that could be considered, for instance knowledge binding [BL07,Gro09], where states if an adversary can break the binding property, there exists an extraction procedure to extract the message from adversary; or non-malleable commitment schemes [DIO98,FF00]. With a similar approach one can consider a similar analysis for a wide range of notions.

Discussions on results. As our focus is on decreasing the trust on the setup phase of non-interactive commitment schemes, so one may notice that the final achievement can be a two-party commitment scheme, such that non of the parties will require to trust the setup phase. They simply can agree on some system parameters. Already it is shown that to construct two-party commitment schemes without setup phase, *injective* functions are enough [Blu81,Yao82,GL89]. On the other hand, it is shown that there is no black-box construction of non-interactive commitment schemes from one-way functions [MP12]. In the second positive result, we discuss about two-party commitment schemes that do not require a particular setup phase and their public commitment keys are *trivial*, but they only can achieve (sub-)hiding and (sub-)biding (Discussed in Section 5.2). In many cryptographic systems, it is shown the deployed commitment schemes require equivocality, specially in minimizing the round complexity of zero-knowledge proofs [BFM88], or even constructing efficient non-interactive zero-knowledge proofs [GS08,Gro10,Lip12]. We discuss in section 5.1 where one can achieve sub-equivocality and binding in commitment schemes used in such zero-knowledge proofs [Lip12]. We note that sub-equivocality is a stronger notion than standard equivocality as in the first one, the setup phase should be simulatable even if it is done maliciously. So a direct observation is that sub-equivocality can decrease the trust on such proof systems. Finally, there are cases where we require to use non-interactive commitment schemes but under standard (falsifiable) assumptions, specially about universally composable commitment schemes [Can01]. Our

result in section 5.3 shows still in such cases the committer can achieve a level of security (precisely, sub-hiding) without trusting to the setup phase, which is stronger than previous cases.

2 Preliminaries

Let $\lambda \in \mathbb{N}$ be the information-theoretic security parameter, and 1^λ denotes its unary representation; say $\lambda = 128$. If x is a (binary) string then $|x|$ is its length. If S is a finite set then $|S|$ denotes its size and $s \leftarrow_\$ S$ denotes picking an element uniformly random from S and assigning it to s. The empty string is shown with $\{\}$, e.g. $\mathsf{ck} = \{\}$. All adversaries will be stateful. For an algorithm $\mathcal{A}$, let $\mathrm{im}(\mathcal{A})$ be the image of $\mathcal{A}$, i.e., the set of valid outputs of $\mathcal{A}$, let $\mathsf{RND}(\mathcal{A})$ denote the random tape of $\mathcal{A}$, and let $r \leftarrow_\$ \mathsf{RND}(\mathcal{A})$ denote sampling of a randomizer r of sufficient length for $\mathcal{A}$'s needs. By $y \leftarrow \mathcal{A}(x; r)$ we denote the fact that $\mathcal{A}$, given an input x and a randomizer r, outputs y. By $y \leftarrow_\$ \mathcal{A}(x; \cdots)$, we denote letting $y \leftarrow \mathcal{A}(x; r)$ for random r. For algorithms $\mathcal{A}$ and $\mathsf{Ext}_\mathcal{A}$, we write $(y \,\|\, y') \leftarrow (\mathcal{A} \,\|\, \mathsf{Ext}_\mathcal{A})(x; r)$ as a shorthand for "$y \leftarrow \mathcal{A}(x; r)$, $y' \leftarrow \mathsf{Ext}_\mathcal{A}(x; r)$". Note that $\mathsf{Ext}_\mathcal{A}$ and $\mathcal{A}$ use internally the same randomness r. We denote by $\mathsf{negl}(\lambda)$ an arbitrary negligible function, and by $\mathsf{poly}(\lambda)$ an arbitrary polynomial function. For a tuple of integers $\Gamma = (\gamma_1, \ldots, \gamma_n)$ with $\gamma_i \le \gamma_{i+1}$, let $(a_i)_{i \in \Gamma} = (a_{\gamma_1}, \ldots, a_{\gamma_n})$. We sometimes denote $(a_i)_{i \in [n]}$ as $\boldsymbol{a}$. We say that $\Gamma = (\gamma_1, \ldots, \gamma_2) \in \mathbb{Z}$ is an (n, λ)-*nice* tuple, if $0 \le \gamma_1 \le \cdots \le \gamma_i \le \gamma_n = \mathsf{poly}(\lambda)$. For distributions A and B, $A \approx_c B$ means that they are computationally indistinguishable. In games, we let $\Pr[G]$ denote the probability that game G returns true.

In the case of pairing-based groups, we will use additive notation together with the bracket notation [EHK+13], i.e., in group $\mathbb{G}_\mu$, $[a]_\mu = a[1]_\mu$, where $[1]_\mu$ is a fixed generator of $\mathbb{G}_\mu$. A *bilinear group generator* $\mathsf{BGgen}(1^\lambda)$ returns $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2)$, where p (a large prime) is the order of cyclic abelian groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$. Finally, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficient non-degenerate bilinear pairing, s.t. $\hat{e}([a]_1, [b]_2) = [ab]_T$. Denote $[a]_1 \bullet [b]_2 = \hat{e}([a]_1, [b]_2)$.

3 Security of Commitment Schemes under Parameters Subversion

As briefly mentioned in introduction, to get a clear understanding of achievable security in commitment schemes we first precisely determine new goals. Before going through them, we recall definition and standard requirements of non-interactive commitment schemes in current setting, when the setup phase is trusted. Recall that in a non-interactive commitment scheme, a committer computes the commitment to a secret value and sends it to the receiver. Later, the opening phase is also non-interactive, which might be simply revealing committed value and used randomness or giving a proof-of-knowledge that the committed value is known for committer.

In basic form, a non-interactive commitment scheme consists of a tuple of polynomial time algorithms $(\mathsf{KGen}, \mathsf{Com}_\mathsf{ck}, \mathsf{OpenVer}_\mathsf{ck})$. The algorithm KGen is a

probabilistic algorithm that takes as input the setup information gk and generates a commitment key ck and a trapdoor key tk. The setup information can for instance describe a finite group over which we are working, or simply the security parameter written in unary representation. The commitment key ck specifies a message space $\mathcal{M}_{\mathsf{ck}}$, a randomizer space $\mathcal{R}_{\mathsf{ck}}$ and a commitment space $\mathcal{C}_{\mathsf{ck}}$. It is usually assumed that it is easy to verify membership of the message space, randomizer space and the commitment space and it is possible to sample randomizers uniformly at random from $\mathcal{R}_{\mathsf{ck}}$. The algorithm $\mathsf{Com}_{\mathsf{ck}}$ takes as input the commitment key ck, a message $m \in \mathcal{M}_{\mathsf{ck}}$, a randomizer $r \in \mathcal{R}_{\mathsf{ck}}$ from the randomizer space and outputs a commitment $c \in \mathcal{C}_{\mathsf{ck}}$ and an opening information op. Finally, the algorithm $\mathsf{OpenVer}_{\mathsf{ck}}$ takes as input the commitment key ck, a commitment $c \in \mathcal{C}_{\mathsf{ck}}$, a message $m \in \mathcal{M}_{\mathsf{ck}}$ and opening information op and returns 1 if c is the commitment of m with opening information op; otherwise returns 0.

In this paper, we consider (subversion-resistant) equivocal commitment (a.k.a. trapdoor commitment) schemes that additionally achieve equivocality which is a stronger notion in comparison with hiding. In such cases, a commitment scheme consists of a tuple of algorithms $(\mathsf{KGen}, \mathsf{Com}_{\mathsf{ck}}, \mathsf{OpenVer}_{\mathsf{ck}}, \mathsf{Sim.KGen}, \mathsf{Com}^*_{\mathsf{tk}}, \mathsf{Equiv}_{\mathsf{tk}})$. In an equivocal commitment scheme, given the secret trapdoor tk associated with commitment key ck, it is possible to open a commitment to any message. This property is usually considered by additional PPT algorithms $\mathsf{Com}^*_{\mathsf{tk}}$ and $\mathsf{Equiv}_{\mathsf{tk}}$. $\mathsf{Com}^*_{\mathsf{tk}}$ that take the trapdoor tk (generated by $\mathsf{Sim.KGen}$) as input and outputs an equivocal commitment c and an equivocation key ek. Then, $\mathsf{Equiv}_{\mathsf{tk}}$ on inputs ek, c and a message $m \in \mathcal{M}_{\mathsf{ck}}$ creates an opening $\mathsf{op} \in \mathcal{R}_{\mathsf{ck}}$ of the commitment, so that $c = \mathsf{Com}_{\mathsf{ck}}(m; \mathsf{op})$.

Beside all the mentioned algorithms for (non-subversion-resistant) non-interactive commitment schemes, in this paper, we introduce a new algorithm CKVer that verifies well-formedness of public commitment key ck and is necessary to construct subversion-resistant commitment schemes. In the rest, we formally define a (subversion-resistant) commitment scheme and the target goals.

Definition 1 ((Subversion-resistant) Equivocal Commitment Scheme). *An (subversion-resistant) equivocal commitment scheme consists of a tuple of algorithms* $\Pi_{com} = (\mathsf{KGen}, \mathsf{CKVer}, \mathsf{Com}_{\mathsf{ck}}, \mathsf{OpenVer}_{\mathsf{ck}}, \mathsf{Sim.KGen}, \mathsf{Com}^*_{\mathsf{tk}}, \mathsf{Equiv}_{\mathsf{tk}})$ *as described below,*

Key Generation, $\mathsf{ck} \leftarrow \mathsf{KGen}(\mathsf{gk})$**:** *Generates a commitment public key* ck *and associated trapdoor key* tk*. It returns public commitment key* ck *and keeps secret or removes* tk*. It also specifies a message space* $\mathcal{M}_{\mathsf{ck}}$*, a randomness space* $\mathcal{R}_{\mathsf{ck}}$*, and a commitment space* $\mathcal{C}_{\mathsf{ck}}$*. This algorithm is supposed to be run by a trusted or distributed authority;*

Commitment Key Verification, $0/1 \leftarrow \mathsf{CKVer}(\mathsf{gk}, \mathsf{ck})$**:** CKVer *is a probabilistic algorithm that given some setup information* gk *and commitment key* ck*, returns either 0 (the* ck *is incorrectly formed) or 1 (the* ck *is correctly formed);*

Committing, $(c, \mathsf{op}) \leftarrow \mathsf{Com}_{\mathsf{ck}}(m; r)$**:** *Outputs a commitment* c *and an opening information* op*. This algorithm specifies a function* $\mathsf{Com}_{\mathsf{ck}} : \mathcal{M}_{\mathsf{ck}} \times \mathcal{R}_{\mathsf{ck}} \rightarrow \mathcal{C}_{\mathsf{ck}}$*.*

Given a message $m \in \mathcal{M}_{\mathsf{ck}}$, the committer picks a randomness $r \in \mathcal{R}_{\mathsf{ck}}$ and computes the commitment $(c, \mathsf{op}) = \mathsf{Com}_{\mathsf{ck}}(m; r)$.

Opening Verification, $0/1 \leftarrow \mathsf{OpenVer}_{\mathsf{ck}}(c, m, \mathsf{op})$**:** *Outputs 1 if the value $m \in \mathcal{M}_{\mathsf{ck}}$ is the committed message in the commitment c with opening value op, and returns 0 if (m, op, c) does not correspond to a valid pair opening-commitment.*

Simulation of Key Generation, $(\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{Sim.KGen}(\mathsf{gk})$**:** *Generates a commitment public key ck and associated trapdoor key tk. It also specifies a message space $\mathcal{M}_{\mathsf{ck}}$, a randomness space $\mathcal{R}_{\mathsf{ck}}$, and a commitment space $\mathcal{C}_{\mathsf{ck}}$.*

Trapdoor Committing, $(c, \mathsf{ek}) \leftarrow \mathsf{Com}^*_{\mathsf{tk}}(\mathsf{ck})$**:** *Given commitment key ck and tk as input, outputs an equivocal commitment c and an equivocation key ek.*

Trapdoor Opening, $\mathsf{op} \leftarrow \mathsf{Equiv}_{\mathsf{tk}}(c, m, \mathsf{ek})$**:** *On inputs equivocation key ek, c and a message $m \in \mathcal{M}_{\mathsf{ck}}$ creates an opening $\mathsf{op} \in \mathcal{R}_{\mathsf{ck}}$ of the commitment, so that $c = \mathsf{Com}_{\mathsf{ck}}(m; \mathsf{op})$ and returns op.*

A (subversion-resistant) non-interactive commitment scheme satisfies *completeness* if for $\mathsf{ck} \leftarrow \mathsf{KGen}(\mathsf{gk})$ and any honestly generated commitment of $m \in \mathcal{M}_{\mathsf{ck}}$, it successfully passes the verification by $\mathsf{OpenVer}_{\mathsf{ck}}(\mathsf{Com}_{\mathsf{ck}}(m; \mathsf{op}), m, \mathsf{op})$.

3.1 Notions for Honest Parameters: Hiding, Equivocality, Binding

A non-interactive commitment scheme $(\mathsf{KGen}, \mathsf{Com}_{\mathsf{ck}}, \mathsf{OpenVer}_{\mathsf{ck}}, \mathsf{Sim.KGen}, \mathsf{Com}^*_{\mathsf{tk}}, \mathsf{Equiv}_{\mathsf{tk}})$ has to satisfy various security definitions including hiding, binding and equivocality that are defined as below.

Hiding. For tuple $(\mathsf{KGen}, \mathsf{Com}_{\mathsf{ck}}, \mathsf{OpenVer}_{\mathsf{ck}})$, it is hard for any adversary $\mathcal{A}$, to generate two messages $m_0, m_1 \in \mathcal{M}_{\mathsf{ck}}$ such that $\mathcal{A}$ can distinguish between their corresponding commitments c_0 and c_1 where $(c_0, \mathsf{op}_0) \leftarrow \mathsf{Com}_{\mathsf{ck}}(m_0; r_0)$ and $(c_1, \mathsf{op}_1) \leftarrow \mathsf{Com}_{\mathsf{ck}}(m_1; r_1)$. Meaning that, for all security parameter λ,

$$\Pr\left[\begin{array}{l}\mathsf{ck} \leftarrow \mathsf{KGen}(\mathsf{gk}), (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{ck}), b \leftarrow_{\$} \{0,1\}, \\ r_b \leftarrow_{\$} \mathcal{R}_{\mathsf{ck}}, (c_b, \mathsf{op}_b) \leftarrow \mathsf{Com}_{\mathsf{ck}}(m_b; r_b), b' \leftarrow \mathcal{A}(\mathsf{ck}, c_b) : b' = b\end{array}\right] \approx_\lambda 0 \ .$$

Binding. For tuple $(\mathsf{KGen}, \mathsf{Com}_{\mathsf{ck}}, \mathsf{OpenVer}_{\mathsf{ck}})$, it is hard for any adversary $\mathcal{A}$, to come up with a collision $(c, m_0, \mathsf{op}_0, m_1, \mathsf{op}_1)$, such that op_0 and op_1 are valid opening values for two different pre-images $m_0 \neq m_1$ for c. Meaning that for any adversary $\mathcal{A}$, for all security parameter λ, the following probability is negligible,

$$\Pr\left[\begin{array}{l}\mathsf{ck} \leftarrow \mathsf{KGen}(\mathsf{gk}), (c, (m_0, \mathsf{op}_0), (m_1, \mathsf{op}_1)) \leftarrow \mathcal{A}(\mathsf{ck}) : (m_0 \neq m_1) \\ \wedge (\mathsf{OpenVer}_{\mathsf{ck}}(c, m_0, \mathsf{op}_0) = 1) \wedge (\mathsf{OpenVer}_{\mathsf{ck}}(c, m_1, \mathsf{op}_1) = 1)\end{array}\right] \approx_\lambda 0 \ .$$

Equivocability. A tuple $(\mathsf{KGen}, \mathsf{Com}_{\mathsf{ck}}, \mathsf{OpenVer}_{\mathsf{ck}}, \mathsf{Sim.KGen}, \mathsf{Com}^*_{\mathsf{tk}}, \mathsf{Equiv}_{\mathsf{tk}})$ is equivocable if there exist PPT algorithms $\mathsf{Com}^*_{\mathsf{tk}}$ (trapdoor committing) and $\mathsf{Equiv}_{\mathsf{tk}}$ (trapdoor opening) that given trapdoor of public commitment key, can

come up with a fake commitment and a valid opening such that they would be indistinguishable from the real ones. More formally,

$$\Pr\begin{bmatrix}\mathsf{ck} \leftarrow \mathsf{KGen}(\mathsf{gk}), m \leftarrow \mathcal{A}(\mathsf{ck}),\\ r \leftarrow_{\$} \mathcal{R}_{\mathsf{ck}}, (c, \mathsf{op}) \leftarrow \mathsf{Com}_{\mathsf{ck}}(m; r) :\\ \mathcal{A}(c, \mathsf{op}) = 1\end{bmatrix} \approx_{\lambda} \Pr\begin{bmatrix}(\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{Sim.KGen}(\mathsf{gk}),\\ m \leftarrow \mathcal{A}(\mathsf{ck}), (c, \mathsf{ek}) \leftarrow \mathsf{Com}^{*}_{\mathsf{tk}}(\mathsf{ck}),\\ \mathsf{op} \leftarrow \mathsf{Equiv}_{\mathsf{tk}}(c, m, \mathsf{ek}) : \mathcal{A}(c, \mathsf{op}) = 1\end{bmatrix}$$

where $\mathcal{A}$ outputs $m \in \mathcal{M}_{\mathsf{ck}}$. It is worth to mention that equivocality implies hiding, as a commitment is indistinguishable from an equivocal commitment that can be opened to any message.

3.2 Notions for Subverted Parameters: Sub-hiding, Sub-equivocality, Sub-binding

As we observed in Def. 1, a critical assumption in non-interactive commitment schemes is that the public commitment key ck is honestly generated by a trusted third party or distributed authority. But as our main target is to consider achievable security in commitment schemes when trusted setup phase is compromised, so we cannot assume such assumption and instead we define subversion-resistance analogues sub-hiding, sub-equivocality and sub-binding of the notions hiding, equivocality and binding. In new notions, the key difference is that the setup phase is compromised and the commitment key ck is selected by an adversary, refereed as $\mathcal{A}$ (or a subverter) rather than via the honest key-generation algorithm KGen prescribed by Π_{com}. Also, as briefly mentioned before, there is a new algorithm CKVer that verifies well-formedness of public commitment key ck.

Sub-hiding. Subversion hiding asks that if an adversary, refereed as $\mathcal{A}$ (or a subverter) creates a *well-formed* commitment public key ck in any way it likes, it still will not be able to say which of two messages m_0 and m_1 is committed, even if it picks both messages itself. In other words, it is hard for any adversary $\mathcal{A}$ to generate two messages $m_0, m_1 \in \mathcal{M}_{\mathsf{ck}}$ such that $\mathcal{A}$ can distinguish between their corresponding commitments c_0 and c_1 where $(c_0, \mathsf{op}_0) \leftarrow \mathsf{Com}_{\mathsf{ck}}(\mathsf{ck}, m_0)$ and $(c_1, \mathsf{op}_1) \leftarrow \mathsf{Com}_{\mathsf{ck}}(\mathsf{ck}, m_1)$ even if the well-formed commitment public key ck is generated by the adversary $\mathcal{A}$. Meaning that, for all security parameter λ,

$$\Pr\begin{bmatrix}(\mathsf{ck}, (m_0, m_1)) \leftarrow \mathcal{A}(\mathsf{gk}), b \leftarrow_{\$} \{0,1\}, \mathsf{CKVer}(\mathsf{gk}, \mathsf{ck}) = 1,\\ r_b \leftarrow_{\$} \mathcal{R}_{\mathsf{ck}}, (c_b, \mathsf{op}_b) \leftarrow \mathsf{Com}_{\mathsf{ck}}(m_b; r_b), b' \leftarrow \mathcal{A}(\mathsf{ck}, c_b) : b' = b\end{bmatrix} \approx_{\lambda} 0 \ .$$

The definition reflects that a malicious key generator $\mathcal{A}$ can be considered as a malicious verifier $\mathcal{A}$ who aims to learn about the committed message before officially opening by the committer. In new definitions, by *well-formedness* of ck, we mean CKVer will verify ck successfully.

Sub-binding. Subversion binding implies that if an adversary $\mathcal{A}$ (a malicious committer) creates a well-formed commitment public key ck in any way it likes

(consequently it knows the corresponding trapdoor tk), it still will not be able to double open. In other words, it is hard for any adversary $\mathcal{A}$, to come up with a collision $(c, m_0, \mathsf{op}_0, m_1, \mathsf{op}_1)$, such that op_0 and op_1 are valid opening values for two different pre-images $m_0 \neq m_1$ for c, even if the well-formed commitment public key ck is generated by $\mathcal{A}$ itself. Basically, for all security parameter λ,

$$\Pr\left[\begin{array}{l}(\mathsf{ck}, c, (m_0, \mathsf{op}_0), (m_1, \mathsf{op}_1)) \leftarrow \mathcal{A}(\mathsf{gk}) : \mathsf{CKVer}(\mathsf{gk}, \mathsf{ck}) = 1 \ \wedge (m_0 \neq m_1) \\ \wedge\, (\mathsf{OpenVer}_{\mathsf{ck}}(c, m_0, \mathsf{op}_0) = 1) \wedge (\mathsf{OpenVer}_{\mathsf{ck}}(c, m_1, \mathsf{op}_1) = 1)\end{array}\right] \approx_\lambda 0 \ .$$

Sub-binding reflects a malicious key generator $\mathcal{A}(\mathsf{gk})$ is like a cheating committer $\mathcal{A}$ who aims to double open. So the definition considers general case by allowing $\mathcal{A}$ to generate the commitment key ck.

Sub-equivocability. Sub-equivocality guarantees that even if an adversary $\mathcal{A}$ (a malicious key generator) generates a well-formed commitment public key ck, there exists a simulator which is able to produce the full view of key generation phase, and also by having secret trapdoor tk associated with the public commitment key ck, it is possible to create a fake commitment which can be opened to any message. More formally,

$$\Pr\left[\begin{array}{l}(\mathsf{ck}, m) \leftarrow \mathcal{A}(\mathsf{gk}), \\ \mathsf{CKVer}(\mathsf{gk}, \mathsf{ck}) = 1, r \leftarrow_{\$} \mathcal{R}_{\mathsf{ck}}, \\ (c, \mathsf{op}) \leftarrow \mathsf{Com}_{\mathsf{ck}}(m; r) : \\ \mathcal{A}(c, \mathsf{op}) = 1\end{array}\right] \approx_\lambda \Pr\left[\begin{array}{l}((\mathsf{ck}, \mathsf{tk}), m) \leftarrow \mathsf{Sim}.\mathcal{A}(\mathsf{gk}), \\ \mathsf{CKVer}(\mathsf{gk}, \mathsf{ck}) = 1, \\ (c, \mathsf{ek}) \leftarrow \mathsf{Com}^*_{\mathsf{tk}}(\mathsf{ck}), \\ \mathsf{op} \leftarrow \mathsf{Equiv}_{\mathsf{tk}}(c, m, \mathsf{ek}) : \mathcal{A}(c, \mathsf{op}) = 1\end{array}\right]$$

where $m \in \mathcal{M}_{\mathsf{ck}}$, and $\mathsf{Sim}.\mathcal{A}$ is the simulator of key generation. One can observe that sub-equivocality implies sub-hiding and standard equivocality. We also already know that equivocality implies hiding (more details in Tab. 1).

4 Negative Result: Sub-binding with Equivocability are not Compatible

Currently there are commitment schemes that can achieve hiding, equivocality and binding, e.g. Pedersen commitment schemes [Ped92]. In this section, we aim to consider if we can achieve sub-binding without degrading already existing mentioned properties. We note that achieving sub-binding individually is possible (e.g. by sending plain message) but such scheme will not guarantee equivocality. Here our main target is the practically-interested cases. So, as a first practical scenario, we consider possibility of achieving sub-binding and equivocality at the same time. Meaning that we aim to keep already existing property equivocality and additionally achieve sub-binding. We show that equivocality and sub-binding are not compatible and this is impossible. This impossibility result implies we cannot construct a commitment scheme that can achieve sub-binding and equivocality at the same time. Intuitively, if we consider a non-interactive commitment scheme Π_{com} which guarantees equivocality

and sub-binding at the same time, we show that an adversary can use the simulator of setup phase $\mathsf{Sim.KGen}$, and algorithms $\mathsf{Com}^*_{\mathsf{tk}}$ and $\mathsf{Equiv}_{\mathsf{tk}}$ and break the sub-binding property of the commitment scheme, unless the verification of commitment scheme is *trivial*. By trivial we mean, the verification algorithm can decide about validity of commitment c and opening information op on its own.

Let Π_{com} be a non-interactive commitment scheme which guarantees sub-binding and equivocality. A *commitment-opening instance generator* COG is a polynomial algorithm that on input group description gk returns a pair $(c,(m,\mathsf{op}))$, where c is a commitment, m is a committed message, and op is an opening information (e.g. randomness). Here c is a challenge commitment that may or may not be a `valid` commitment for (m,op), and (m,op) should be `valid` opening (it means $\mathsf{OpenVer}_{\mathsf{ck}}$ will accept them) if c is a `valid` commitment. Let DP be an algorithm (decision procedure) that on inputs gk and c returns a Boolean, showing whether or not it thinks c is `valid` commitment. Now, consider experiment DEC as below associated to a commitment-opening instance generator COG, verification algorithm $\mathsf{OpenVer}_{\mathsf{ck}}$ and decision procedure DP,

Experiment $\mathsf{DEC}_{\mathsf{COG},\mathsf{OpenVer}_{\mathsf{ck}},\mathsf{DP}}(\mathsf{gk})$:

$(c,(m,\mathsf{op})) \leftarrow\!\!\$\ \mathsf{COG}(\mathsf{gk})$; $d_1 \leftarrow \mathsf{OpenVer}_{\mathsf{ck}}(c,(m,\mathsf{op}))$;
If (c is `valid` and $d_1 = \mathsf{false}$) then return false;
$d_0 \leftarrow\!\!\$\ \mathsf{DP}(\mathsf{gk},c)$;
Return $d_0 \neq d_1$;

Let $\mathsf{Adv}^{\mathsf{DEC}}_{\mathsf{COG},\mathsf{OpenVer}_{\mathsf{ck}},\mathsf{DP}}(\mathsf{gk}) = \Pr[\mathsf{DEC}_{\mathsf{COG},\mathsf{OpenVer}_{\mathsf{ck}},\mathsf{DP}}(\mathsf{gk})]$. Now, we say that algorithm DP decides $\mathsf{OpenVer}_{\mathsf{ck}}$ if for every polynomial time COG the function $\mathsf{Adv}^{\mathsf{DEC}}_{\mathsf{COG},\mathsf{OpenVer}_{\mathsf{ck}},\mathsf{DP}}(\mathsf{gk})$ is negligible in λ. We say that verification by $\mathsf{OpenVer}_{\mathsf{ck}}$ is trivial if there is a polynomial time algorithm DP that decides $\mathsf{OpenVer}_{\mathsf{ck}}$. Intuitively, in experiment DEC, think of COG as an adversary trying to make DP fail. The experiment returns true when COG succeeds, meaning that DP returns the wrong decision (as the experiment returns $d_0 \neq d_1$). A technical point is if COG generates a `valid` commitment c, the game forces it to lose if (m,op) are not `valid` opening. Thus we are asking that DP is able to decide validity of commitment c in polynomial time for challenge commitment c that can be efficiently generated with valid (m,op) if the commitment is `valid`.

Now lets go through the negative result. The result is established as a theorem in the rest, but to get the intuition behind the procedure, one may notice that the definition of equivocality states given trapdoor tk of commitment key ck, one can use algorithms $\mathsf{Com}^*_{\mathsf{tk}}$, $\mathsf{Equiv}_{\mathsf{tk}}$ and crate a fake commitment and an opening which are indistinguishable from a honestly generated commitment and opening. On the other side, sub-binding requires that given the trapdoor tk of the commitment key ck, an adversary should not be able to double open, which is not compatible with the definition of equivocality. Because given trapdoor tk, the adversary of sub-binding can use algorithms $\mathsf{Com}^*_{\mathsf{tk}}$ and $\mathsf{Equiv}_{\mathsf{tk}}$ provided by equivocality and break the sub-binding by generating a fake commitment and two different valid openings. This is the main idea behind the following theorem.

Theorem 1 (Impossibility of Sub-binding Along with Equivocability). *Assume Π_{com} be a non-interactive commitment scheme satisfying equivocality and sub-binding. Then the verification* $\mathsf{OpenVer}_{\mathsf{ck}}$ *is trivial.*

Proof. Equivocality of commitment scheme Π_{com} implies that given tk generated as $(\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{Sim.KGen}(\mathsf{gk})$, for an arbitrary $m \leftarrow \mathcal{M}_{\mathsf{ck}}$, there are two algorithms $\mathsf{Com}^*_{\mathsf{tk}}$ and $\mathsf{Equiv}_{\mathsf{tk}}$ that can generate acceptable (c, op) as the following: $(c, \mathsf{ek}) \leftarrow \mathsf{Com}^*_{\mathsf{tk}}(\mathsf{ck})$, $\mathsf{op} \leftarrow \mathsf{Equiv}_{\mathsf{tk}}(c, m, \mathsf{ek})$. Without loss of generality, in the rest, we consider a notion of equivocality which states that given the trapdoor tk, the algorithm $\mathsf{Equiv}_{\mathsf{tk}}$ can open *any valid* commitments, instead of *only* commitments output by $\mathsf{Com}^*_{\mathsf{tk}}$. This is sort of a stronger notion of equivocality. By this in mind, consider the following decision procedure DP,

Algorithm $\mathsf{DP}(\mathsf{gk}, c)$

$(\mathsf{ck}, \mathsf{tk}) \leftarrow_{\$} \mathsf{Sim.KGen}(\mathsf{gk})$; $(c, \mathsf{ek}) \leftarrow \mathsf{Com}^*_{\mathsf{tk}}(\mathsf{ck})$; $m \leftarrow \mathcal{M}_{\mathsf{ck}}$, $\mathsf{op} \leftarrow \mathsf{Equiv}_{\mathsf{tk}}(c, m, \mathsf{ek})$;
Returns $\mathsf{OpenVer}_{\mathsf{ck}}(c, m, \mathsf{op})$

Thus, to decide if c is a `valid` commitment, algorithm DP runs the simulator of key generations $\mathsf{Sim.KGen}$ to obtain simulated ck and corresponding trapdoor tk. Then the algorithm uses tk to generate the commitment c and equivocal key ek. Next, it uses ek, c and m and generates opening op, and finally by verification algorithm decides whether $(c, (m, \mathsf{op}))$ are valid commitment and opening. Let COG be any polynomial time commitment-opening generator. We will show that $\mathsf{Adv}^{\mathsf{DEC}}_{\mathsf{COG},\mathsf{OpenVer}_{\mathsf{ck}},\mathsf{DP}}(\mathsf{gk})$ is negligible. This shows verification $\mathsf{OpenVer}_{\mathsf{ck}}$ is trivial. To show $\mathsf{Adv}^{\mathsf{DEC}}_{\mathsf{COG},\mathsf{OpenVer}_{\mathsf{ck}},\mathsf{DP}}(\mathsf{gk})$ is negligible, below we will define polynomial time adversaries A and B such that

$$\mathsf{Adv}^{DEC}_{\mathsf{COG},\mathsf{OpenVer}_{\mathsf{ck}},\mathsf{DP}}(\mathsf{gk}) \leq \mathsf{Adv}^{equivocality}_{\mathsf{COG},\mathsf{OpenVer}_{\mathsf{ck}},A}(\mathsf{gk}) + \mathsf{Adv}^{sub-binding}_{\mathsf{COG},\mathsf{OpenVer}_{\mathsf{ck}},B}(\mathsf{gk})$$

for all $\lambda \in \mathbb{N}$ (note that λ is in description of group gk). By assumption, the commitment scheme satisfies equivocality and sub-binding, so both $\mathsf{Adv}^{equivocality}_{\mathsf{COG},\mathsf{OpenVer}_{\mathsf{ck}},A}(\mathsf{gk})$ and $\mathsf{Adv}^{sub-binding}_{\mathsf{COG},\mathsf{OpenVer}_{\mathsf{ck}},B}(\mathsf{gk})$ are negligible. Thus, $\mathsf{Adv}^{dec}_{\mathsf{COG},\mathsf{OpenVer}_{\mathsf{ck}},\mathsf{DP}}(\mathsf{gk})$ in left side of above inequality is negligible, as desired. Consider experiments $\mathsf{Exp}_0, \mathsf{Exp}_1$ and Exp_2 as described below. Experiments Exp_0 and Exp_1 split up the verification (decision) process depending on whether c is `valid` or not.

Exp_0:
$(c, (m_1, \mathsf{op}_1)) \leftarrow_{\$} \mathsf{COG}(\mathsf{gk})$; $d_1 \leftarrow \mathsf{OpenVer}_{\mathsf{ck}}(c, (m_1, \mathsf{op}_1))$;
$(\mathsf{ck}, \mathsf{tk}) \leftarrow_{\$} \mathsf{Sim.KGen}(\mathsf{gk})$;
$(c, \mathsf{ek}) \leftarrow \mathsf{Com}^*_{\mathsf{tk}}(\mathsf{ck})$; $m_2 \leftarrow \mathcal{M}_{\mathsf{ck}}$; $\mathsf{op}_2 \leftarrow \mathsf{Equiv}_{\mathsf{tk}}(c, m_2, \mathsf{ek})$;
$d_0 \leftarrow \mathsf{OpenVer}_{\mathsf{ck}}(c, m_2, \mathsf{op}_2)$; $b \leftarrow ((c \text{ is not } \texttt{valid}) \wedge (d_0 = \mathsf{true}))$
Return b

$\underline{\mathsf{Exp}_1}$:

$(c, (m_1, \mathsf{op}_1)) \leftarrow_\$ \mathsf{COG}(\mathsf{gk})$; $d_1 \leftarrow \mathsf{OpenVer}_{\mathsf{ck}}(c, (m_1, \mathsf{op}_1))$;
$(\mathsf{ck}, \mathsf{tk}) \leftarrow_\$ \mathsf{Sim.KGen}(\mathsf{gk})$;
$(c, \mathsf{ek}) \leftarrow \mathsf{Com}^*_{\mathsf{tk}}(\mathsf{ck})$; $m_2 \leftarrow \mathcal{M}_{\mathsf{ck}}$; $\mathsf{op}_2 \leftarrow \mathsf{Equiv}_{\mathsf{tk}}(c, m_2, \mathsf{ek})$;
$d_0 \leftarrow \mathsf{OpenVer}_{\mathsf{ck}}(c, m_2, \mathsf{op}_2)$; $b \leftarrow ((d_1 = \mathsf{true}) \wedge (d_0 = \mathsf{false}))$;
Return b

$\underline{\mathsf{Exp}_2}$:

$(c, (m_1, \mathsf{op}_1)) \leftarrow_\$ \mathsf{COG}(\mathsf{gk})$; $d_1 \leftarrow \mathsf{OpenVer}_{\mathsf{ck}}(c, (m_1, \mathsf{op}_1))$; $(\mathsf{ck}, \mathsf{tk}) \leftarrow_\$ \mathsf{KGen}(\mathsf{gk})$;
$m_2 \leftarrow \mathcal{M}_{\mathsf{ck}}$; $(c, \mathsf{op}_2) \leftarrow \mathsf{Com}_{\mathsf{ck}}(m_2; r_2)$;
$d_0 \leftarrow \mathsf{OpenVer}_{\mathsf{ck}}(c, m_2, \mathsf{op}_2)$; $b \leftarrow ((d_1 = \mathsf{true}) \wedge (d_0 = \mathsf{false}))$;
Return b

Experiment Exp_2 switches to the honest key and commitment generations, that can be done as the commitment-opening instance generator COG provided an opening. Experiment DEC returns true iff,

$$\underline{(c \text{ is not } \texttt{valid}) \wedge (d_0 = \mathsf{true})} \text{ OR } \underline{(c \text{ is } \texttt{valid}) \wedge (d_1 = \mathsf{true}) \wedge (d_0 = \mathsf{false})}.$$

The first condition (left one), is equivalent to the case that experiment Exp_0 returns true, and the second condition (right one) is equivalent to $\underline{(d_1 = \mathsf{true}) \wedge (d_0 = \mathsf{false})}$ (as valid commitments always are accepted), which is equivalent to the case when experiment Exp_1 returns true.
Furthermore the conditions are mutually exclusive and cannot both occur at the same time. Therefore, we have

$$\begin{aligned}\mathsf{Adv}^{DEC}_{\mathsf{COG},\mathsf{OpenVer}_{\mathsf{ck}},\mathsf{DP}}(\mathsf{gk}) &= \Pr[\mathsf{Exp}_0] + \Pr[\mathsf{Exp}_1] \\ &= \Pr[\mathsf{Exp}_0] + \Pr[\mathsf{Exp}_2] + (\Pr[\mathsf{Exp}_1] - \Pr[\mathsf{Exp}_2]) \qquad (1)\end{aligned}$$

Notice that by completeness of commitment scheme Π_{com}, we know $\Pr[\mathsf{Exp}_2] = 0$. As a result, the advantage $\mathsf{Adv}^{DEC}_{\mathsf{COG},\mathsf{OpenVer}_{\mathsf{ck}},\mathsf{DP}}(\mathsf{gk})$

$$\mathsf{Adv}^{dec}_{\mathsf{COG},\mathsf{OpenVer}_{\mathsf{ck}},\mathsf{DP}}(\mathsf{gk}) = \Pr[\mathsf{Exp}_0] + (\Pr[\mathsf{Exp}_1] - \Pr[\mathsf{Exp}_2]) \qquad (2)$$

Now we construct two adversaries A and B as shown below,

$\underline{\text{Adversary } A^{equivocality}(\mathsf{gk}, \mathsf{ck})}$:	$\underline{\text{Adversary } B^{sub-binding}(\mathsf{gk})}$:
$(c, (m_1, \mathsf{op}_1)) \leftarrow_\$ \mathsf{COG}(\mathsf{gk})$;	$(c, (m_1, \mathsf{op}_1)) \leftarrow_\$ \mathsf{COG}(\mathsf{gk})$;
$d_1 \leftarrow \mathsf{OpenVer}_{\mathsf{ck}}(c, (m_1, \mathsf{op}_1))$;	$(\mathsf{ck}, \mathsf{tk}) \leftarrow_\$ \mathsf{Sim.KGen}(\mathsf{gk})$;
$m_2 \leftarrow \mathcal{M}_{\mathsf{ck}}$; $(c, \mathsf{op}_2) \leftarrow \mathsf{Com}_{\mathsf{ck}}(m_2; \mathsf{op}_2)$;	$(c, \mathsf{ek}) \leftarrow \mathsf{Com}^*_{\mathsf{tk}}(\mathsf{ck})$;
$d_0 \leftarrow \mathsf{OpenVer}_{\mathsf{ck}}(c, m_2, \mathsf{op}_2)$;	$m_2 \leftarrow \mathcal{M}_{\mathsf{ck}}$, $\mathsf{op}_2 \leftarrow \mathsf{Equiv}_{\mathsf{tk}}(c, m_2, \mathsf{ek})$;
If $(d_1 = \mathsf{true}) \wedge (d_0 = \mathsf{false})$ then $b' = 0$;	Return $(\mathsf{ck}, c, (m_1, \mathsf{op}_1), (m_2, \mathsf{op}_2))$
Else $b' = 1$;	
Return b'	

Note that in constructing adversary B we used the assumption that given tk, the algorithm $\mathsf{Equiv}_{\mathsf{tk}}$ can open *any valid* commitment. One could also use a more general approach by invoking $\mathsf{Equiv}_{\mathsf{tk}}$ twice to open the equivocal commitment to two different messages. By considering adversaries A and B we have,

$$\Pr[\mathsf{Exp}_0] \leq \mathsf{Adv}^{sub-binding}_{\mathsf{COG},\mathsf{OpenVer}_{\mathsf{ck}},B}(\mathsf{gk})$$

$$\Pr[\mathsf{Exp}_1] - \Pr[\mathsf{Exp}_2] \leq \mathsf{Adv}^{equivocality}_{\mathsf{COG},\mathsf{OpenVer}_{\mathsf{ck}},A}(\mathsf{gk}).$$

Next, by substituting last two inequalities in equation (2), we get to the target inequality as

$$\mathsf{Adv}^{dec}_{\mathsf{COG},\mathsf{OpenVer}_{\mathsf{ck}},\mathsf{DP}}(\mathsf{gk}) \leq \mathsf{Adv}^{equivocality}_{\mathsf{COG},\mathsf{OpenVer}_{\mathsf{ck}},A}(\mathsf{gk}) + \mathsf{Adv}^{sub-binding}_{\mathsf{COG},\mathsf{OpenVer}_{\mathsf{ck}},B}(\mathsf{gk}).$$

This results the theorem. □

5 Positive Results

As already mentioned, currently there exist hiding, binding, and equivocal commitment schemes when we trust to the setup phase. In this section, we consider if we can construct subversion-resistant commitment schemes. Commitment schemes that without losing current security guarantees, will achieve to some of subversion-resistant notions defined in section 3.2. For instance, can we achieve sub-equivocality without losing the initial properties? We answer this question positively in subsection 5.1, by presenting a commitment scheme that is sub-equivocal and binding under a knowledge assumption in a group with a bilinear map. By considering the negative result in last section, this is the best case one can achieve if they want to retain equivocality, when public commitment key is subverted. In the rest of our analysis, we consider if we can construct commitment schemes which will satisfy sub-binding? By considering, the negative result in section 4, we know that one cannot construct a commitment scheme which can guarantee sub-binding and equivocality at the same time. In subsection 5.2, we show the best we can achieve while retaining sub-binding is sub-hiding; by introducing non-interactive commitment schemes that simultaneously achieving sub-binding and sub-hiding. First positive result in subsection 5.1 provides sub-equivocality and binding under a knowledge assumption. One may ask, can we relax the requirement of sub-equivocality and aim to retain sub-hiding but from weaker assumptions? In subsection 5.3, we show under minimal assumption that there exist hiding, binding, and sub-equivocal commitment schemes, one can achieve sub-hiding.

5.1 Sub-equivocality and Binding

We present the first positive result by constructing a sub-equivocal and binding commitment scheme, which due to the negative result this is the best scenario.

By considering the definition of sub-equivocality (given in Def. 3.2), to achieve sub-equivocality in a commitment scheme, there must be algorithms $\mathsf{Sim}.\mathcal{A}$, $\mathsf{Com}^*_{\mathsf{tk}}$ and $\mathsf{Equiv}_{\mathsf{tk}}$, where $\mathsf{Sim}.\mathcal{A}$ will simulates *malicious* setup phase, next $\mathsf{Com}^*_{\mathsf{tk}}$ and $\mathsf{Equiv}_{\mathsf{tk}}$ will output a fake commitment and the corresponding valid opening. Unlike standard equivocality, the algorithms $\mathsf{Com}^*_{\mathsf{tk}}$ and $\mathsf{Equiv}_{\mathsf{tk}}$ cannot get honestly generated trapdoors of commitment keys, and also they cannot extract the trapdoors from the malicious key generator $\mathcal{A}$ by rewinding, as they do not

have any interaction with $\mathcal{A}$. So instead, we will rely on a knowledge assumption, which states if a malicious key generator can generate certain outputs, it means he knows underlying secret information. *Knowing underlying secret information* is formalized by showing that there exists a non-black-box extraction procedure which allows to extract the underlying secret information from the malicious key generator. Once we extracted the underlying secret information (more precisely tk as the trapdoor of ck) from the malicious key generator, we can give the extracted trapdoors to algorithms $\mathsf{Com}^*_{\mathsf{tk}}$ and $\mathsf{Equiv}_{\mathsf{tk}}$ to generate a pair of fake but acceptable commitment and opening. To guarantee binding, a minimal requirement is that an adversary cannot obtain the trapdoor tk of ck from a honestly generated commitment key.

In this setting, as we consider the case key generator is malicious, so there is an issue initial parameters, e.g. groups description. They cannot be generated similar to the previous case (by the third party), as they can be subverted. This issue is addressed by considering the groups description gk as a part of the scheme specification. More precisely, since group generation is a deterministic and public procedure, so in subversion-resistant commitment schemes all parties will re-execute group generation themselves to obtain gk (similar to the case in subversion-resistant NIZK arguments [BFS16]). Below we recall two assumptions that are used in proof of the first positive result.

Definition 2 (Γ-Power (Symmetric) Discrete Logarithm Assumption). *Let Γ be an (n,λ)-nice tuple for some $n = \mathsf{poly}(\lambda)$. We say a bilinear group generator BGgen is (n,λ)-PDL secure in group $\mathbb{G}_t$ for $t \in \{1,2\}$, if for any PPT adversary A, $\Pr[\mathsf{gk} := (p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,\hat{e},[1]_1,[1]_2) \leftarrow \mathsf{BGgen}(1^\lambda), [1]_t \leftarrow \mathbb{G}_t\backslash\{1\}, x \leftarrow \mathbb{Z}_p : \mathcal{A}(\mathsf{gk};([x^l]_t)_{\epsilon\Gamma})]$ is negligible in λ. Similarly, we say a bilinear group generator BGgen is Γ-PSDL secure, if for any PPT adversary $\mathcal{A}$,*

$$\Pr\begin{bmatrix}\mathsf{gk} := (p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,\hat{e},[1]_1,[1]_2) \leftarrow \mathsf{BGgen}(1^\lambda),\\ x \leftarrow \mathbb{Z}_p : \mathcal{A}(\mathsf{gk},([x^l]_1,[x^l]_2)_{l\epsilon\Gamma}) = x\end{bmatrix} \approx_\lambda \mathsf{negl}(\lambda) \quad .$$

In [Lip12], Lipmaa has proven that the Γ-PSDL assumption holds in the generic group model for any (n,λ)-*nice* tuple Γ given $n = poly(\lambda)$.

Definition 3 (BDH-KE Assumption). *We say* GenBP *is* BDH-KE secure *for $\mathcal{R}$ if for any λ, $(\mathbf{R},\xi_{\mathbf{R}}) \in \mathrm{im}(\mathcal{R}(1^\lambda))$, and PPT adversary $\mathcal{A}$ there exists a PPT extractor $\mathsf{Ext}_{\mathcal{A}}$, such that*

$$\mathsf{Adv}^{BDH-KE}_{\mathsf{BGgen},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}} = \Pr\begin{bmatrix}(p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,\hat{e},[1]_1,[1]_2) \leftarrow \mathsf{BGgen}(1^\lambda), r \leftarrow_\$ \mathsf{RND}(\mathcal{A}),\\ ([\alpha_1]_1,[\alpha_2]_2 \,\|\, a) \leftarrow (\mathcal{A} \,\|\, \mathsf{Ext}_{\mathcal{A}})(\mathbf{R},\xi_{\mathbf{R}};r) :\\ [\alpha_1]_1 \bullet [1]_2 = [1]_1 \bullet [\alpha_2]_2 \wedge a \neq \alpha_1\end{bmatrix}$$

is negligible in λ. In above assumption, $\xi_{\mathbf{R}}$ is the auxiliary information related to the underlying group.

The BDH-KE assumption is an asymmetric-pairing version of the original knowledge assumption [Dam92].

A Sub-equivocal and Binding Commitment Scheme: We show the variant of knowledge commitment scheme from [Gro10], defined by [Lip12] (shown in Fig. 1) can achieve sub-equivocality and binding under the BDH-KE and Γ-PDL assumptions along with some well-formedness checking on public key ck.

Theorem 2 (Sub-equivocal and Binding Commitment Scheme). *Let* BGgen *be a bilinear group generator. Then the commitment scheme* Π_{com} *described in Fig. 1 is binding in* $\mathbb{G}_t$ *for* $t \in \{1,2\}$*, under the* Γ*-PDL assumption and also satisfies sub-equivocality under the BDH-KE knowledge assumption.*

Proof. As we did not change the elements in commitment key ck and committing procedure $\mathsf{Com}_{\mathsf{ck}}$, so the proof of binding is as original scheme which is done in [Lip12] under the Γ-PSDL assumption in group $\mathbb{G}_t$ for $t \in \{1,2\}$.

About the proof of sub-equivocality, in [Lip12], it is shown that the original scheme is equivocal (trapdoor) when the commitment key ck is generated by a trusted third party (algorithms $\mathsf{Com}^*_{\mathsf{tk}}$ and $\mathsf{Equiv}_{\mathsf{tk}}$ are shown in Fig. 1). In the original scheme, the algorithms $\mathsf{Com}^*_{\mathsf{tk}}$ and $\mathsf{Equiv}_{\mathsf{tk}}$ get the honestly generated trapdoor tk of the commitment key ck. But in *sub-equivocality* the trapdoor tk is not trustable anymore, as the commitment keys are generated by a malicious third party or generally speaking by a malicious verifier.

Let $\mathcal{A}$ shows a malicious key generator. To prove sub-equivocality, we first need to show that the setup phase is simulatable. Technically speaking, there exists $\mathsf{Sim}.\mathcal{A}$ which can produce the full view of key generation by $\mathcal{A}$. To do so, we need to show there exists an extractor $\mathsf{Ext}_{\mathcal{A}}$ that can extract the trapdoor tk from the malicious key generator $\mathcal{A}$. Second, we need to describe two algorithms $\mathsf{Com}^*_{\mathsf{tk}}$ and $\mathsf{Equiv}_{\mathsf{tk}}$ which given the extracted trapdoor they can produce a fake commitment and a valid opening which are indistinguishable from the honestly generated ones. As the second item already is shown in the original scheme [Lip12], so we just need to show one can construct the extractor $\mathsf{Ext}_{\mathcal{A}}$, and simulate the setup phase. To this aim, recall that the BDH-KE assumption for bilinear groups $\mathbb{G}_1$ and $\mathbb{G}_2$ generated by $[1]_1$ and $[1]_2$, respectively, states that from any algorithm, given the group description and generators, which returns a pair $([a]_1, [a]_2)$, one can efficiently extracts a. In the rest of proof, we show one can construct an efficient extractor under BDH-KE assumption, and extract the trapdoor td from malicious key generator $\mathcal{A}$.

Let a malicious commitment key generator $\mathcal{A}$ outputs $\mathsf{ck} = (\mathsf{ck}_1, \mathsf{ck}_2)$, where $\mathsf{ck}_t \leftarrow \{\left[x^{\lambda_i}\right]_t, \left[\hat{a}x^{\lambda_i}\right]_t\}$ for $i \in [0\,..\,n]$ and $t \in \{1,2\}$, as described in Fig. 1. By considering BDH-KE assumption, and verifications done in commitment key verification algorithm CKVer, we can see that if a malicious key generator manages to output a *well-formed* ck, it must know x and $\hat{a}$. By well-formed ck, we mean it must pass the verifications done by algorithm CKVer [3]. So we can conclude

[3] Note that verification equations such as $[\hat{a}]_1 \bullet [1]_2 = [1]_1 \bullet [\hat{a}]_2$ inside CKVer algorithm comes from the definition of BDH-KE knowledge assumption. So to check the well-formedness of commitment key ck, depending on the underlying knowledge assumption in different commitment schemes, one may construct a CKVer algorithm with different verification equations.

Parameter Generation, $\mathsf{gk} \leftarrow \mathsf{BGgen}(1^\lambda)$: Given security parameter 1^λ, $\mathsf{BGgen}(1^\lambda)$ returns $\mathsf{gk} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2)$, where p (a large prime) is the order of cyclic abelian groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$. Finally, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficient non-degenerate bilinear pairing, s.t. $\hat{e}([a]_1, [b]_2) = [ab]_T$. Denote $[a]_1 \bullet [b]_2 = \hat{e}([a]_1, [b]_2)$.

Key Generation, $\mathsf{ck} \leftarrow \mathsf{KGen}(\mathsf{gk})$: First obtains $\mathsf{gk} \leftarrow \mathsf{BGgen}(1^\lambda)$. Let Γ be an (n, λ)-nice tuple for some $n = \mathsf{poly}(\lambda)$. Samples $\hat{a}, x \leftarrow \mathbb{Z}_p$. Let $t \in \{1, 2\}$. Returns the commitment key $\mathsf{ck} = (\mathsf{ck}_1, \mathsf{ck}_2)$ as $\mathsf{ck}_t \leftarrow \{[x^{\lambda_i}]_t, [\hat{a}x^{\lambda_i}]_t\}$ for $i \in [0..n]$ and the corresponding trapdoor tk as $\mathsf{tk} = x$.

Commitment Key Verification, $0/1 \leftarrow \mathsf{CKVer}(\mathsf{gk}, \mathsf{ck})$: CKVer is a probabilistic algorithm which first obtains gk as $\mathsf{gk} \leftarrow \mathsf{BGgen}(1^\lambda)$, and then given the commitment key ck, does the following verification on elements of ck,

- Checks whether $[\hat{a}]_1 \bullet [1]_2 = [1]_1 \bullet [\hat{a}]_2$
- For $i \in [1..n]$ checks $[x^i]_1 \bullet [1]_2 = [1]_1 \bullet [x^i]_2$
- For $i \in [1..n]$ checks $[\hat{a}x^i]_1 \bullet [1]_2 = [1]_1 \bullet [\hat{a}x^i]_2$
- For $i \in [1..n]$ checks $[\hat{a}]_1 \bullet [x^i]_2 = [1]_1 \bullet [\hat{a}x^i]_2$

and returns 1 if all verifications passed (the ck is correctly formed) or 0 if any of the verifications failed (the ck is incorrectly formed).

Committing, $(c, \mathsf{op}) \leftarrow \mathsf{Com}_{\mathsf{ck}}(\mathbf{m}; \mathbf{r})$: First obtains gk as $\mathsf{gk} \leftarrow \mathsf{BGgen}(1^\lambda)$. Then given $(\mathsf{ck}, \mathbf{m})$ for $\mathsf{CKVer}(\mathsf{gk}, \mathsf{ck}) = 1$, to commit to $\mathbf{a} = (\mathbf{m_1}, \mathbf{m_2}, \ldots, \mathbf{m_n}) \in \mathbb{Z}_\mathbf{p}^\mathbf{n}$, the committer samples a random $r \leftarrow_\$ \mathbb{Z}_p$, and returns (c, op) that are defined as follows,

$$c = (c_t^1, c_t^2) = \mathsf{Com}_{\mathsf{ck}}(\mathsf{ck}_t; \mathbf{a}; \mathbf{r}) := (\mathbf{r}[\mathbf{1}]_\mathbf{t} + \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{m_i} \left[\mathbf{x}^{\boldsymbol{\lambda}_\mathbf{i}}\right]_\mathbf{t}, \hat{\mathbf{a}}\mathbf{r}[\mathbf{1}]_\mathbf{t} + \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{m_i} \left[\hat{\mathbf{a}}\mathbf{x}^{\boldsymbol{\lambda}_\mathbf{i}}\right]_\mathbf{t})$$

Opening Verification, $0/1 \leftarrow \mathsf{OpenVer}_{\mathsf{ck}}(c, m, \mathsf{op})$: Given c, $\mathbf{a}$ and r, it recomputes commitment as in committing phase and check if it is equal to c; if so outputs 1, otherwise 0.

Simulation of Key Generation, $(\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{Sim.KGen}(\mathsf{gk})$: Uses the simulation algorithm in Fig. 3 and generates a commitment public key ck and associated trapdoor key tk.

Trapdoor Committing, $(c, \mathsf{ek}) \leftarrow \mathsf{Com}^*_{\mathsf{tk}}(\mathsf{ck})$: Given commitment key ck and tk as input, outputs an equivocal commitment $c = \mathsf{Com}_{\mathsf{ck}}(0; r) = [r]_t$ where $r \leftarrow \mathbb{Z}_p^2$ and an equivocation key $\mathsf{ek} = r$.

Trapdoor Opening, $\mathsf{op} \leftarrow \mathsf{Equiv}_{\mathsf{tk}}(c, m, \mathsf{ek})$: On input equivocation key $\mathsf{ek} = r \in \mathbb{Z}_p^2$, $c \in \mathcal{C}_{\mathsf{ck}}^2$ and messages $\mathbf{a}$ creates an opening $(m, r') = (\mathbf{a}, \mathbf{r} - \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{m_i}\mathbf{x}^{\boldsymbol{\lambda}_\mathbf{i}})$ for any $\mathbf{a}$, so that $c = \mathsf{Com}_{\mathsf{ck}}(\mathbf{a}; \mathbf{r'})$ and returns $\mathsf{op} = r'$,

Fig. 1: A variation of the knowledge commitment scheme of Groth [Gro10] defined by Lipmaa [Lip12] that can achieve sub-equivocality and binding. We note that in this setting, $\mathsf{gk} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2)$ is part of the scheme specification, and in practice each party can run deterministic algorithm BGgen and re-obtain gk.

Extraction algorithm, $\mathsf{tk} \leftarrow \mathsf{Ext}_{\mathcal{A}}(\mathsf{gk}, \mathsf{ck}, \xi_{\mathbf{R}})$:

Given source code and random coins of the malicious key generator $\mathcal{A}$, and some auxiliary information $\xi_{\mathbf{R}}$ it acts as follows.

For $\alpha \in \{x, \hat{a}\}$ do
    $\alpha \leftarrow \mathsf{Ext}_{\mathcal{A}}(\mathsf{gk}, \mathsf{ck}, \xi_{\mathbf{R}})$
$\mathsf{tk} \leftarrow (x, \hat{a})$
Return tk

Fig. 2: Extraction procedure in simulation of setup phase in sub-equivocal commitment scheme described in Fig. 1

Simulator $\mathsf{Sim}.\mathcal{A}(\mathsf{gk})$:

$\mathsf{gk} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2) \leftarrow \mathsf{BGgen}(1^{\lambda})$;
$\mathsf{ck} \leftarrow_{\$} \mathcal{A}(\mathsf{gk})$; # similar to procedure desribed in Fig. 1
By executing $\mathsf{CKVer}(\mathsf{gk}, \mathsf{ck})$,
check if
- $[\hat{a}]_1 \bullet [1]_2 = [1]_1 \bullet [\hat{a}]_2$
- $[x^i]_1 \bullet [1]_2 = [1]_1 \bullet [x^i]_2$, for $i \in [1..n]$
- $[\hat{a}x^i]_1 \bullet [1]_2 = [1]_1 \bullet [\hat{a}x^i]_2$, for $i \in [1..n]$
- $[\hat{a}]_1 \bullet [x^i]_2 = [1]_1 \bullet [\hat{a}x^i]_2$, for $i \in [1..n]$

then $\mathsf{tk} := x, \hat{a} \leftarrow \mathsf{Ext}_{\mathcal{A}}(\mathsf{gk}, \mathsf{ck})$ # using extractor in Fig. 2
Else $\mathsf{tk} \leftarrow \perp$
Return $(\mathsf{ck}, \mathsf{tk})$

Fig. 3: Simulation of setup phase in the commitment scheme described in Fig. 1 [Gro10,Lip12].

there exists a polynomial time extractor $\mathsf{Ext}_{\mathcal{A}}$ that if all the verifications in algorithm CKVer pass for some $\hat{a}$ and x, then the extractor $\mathsf{Ext}_{\mathcal{A}}$ extracts x and $\hat{a}$; as $\mathsf{Adv}^{BDH-KE}_{\mathsf{BGgen},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}$ is negligible. A high-level description of extraction procedure is shown in Fig. 2.

One may notice that the verifications in algorithm CKVer (described in Fig. 1) verify the well-formedness of all elements in commitment key ck, which is necessary for completeness of the commitment scheme as well. By considering this fact, if all the CKVer's paring checks, then the malicious key generator's outputted commitment key ck has the same structure as one output by the KGen algorithm. Then, using extractor $\mathsf{Ext}_{\mathcal{A}}$ one can do the simulation of malicious key generation using algorithm $\mathsf{Sim}.\mathcal{A}$ described in Fig. 3.

Finally, using the extracted trapdoor tk, one can consider the rest of proof as the proof of equivocality given for original scheme, by showing that given the (extracted) trapdoors one can use two algorithms $\mathsf{Com}^{*}_{\mathsf{tk}}$ and $\mathsf{Equiv}_{\mathsf{tk}}$ (described in Fig. 1) and generate a fake commitment and the corresponding valid opening that will be successfully verified by $\mathsf{OpenVer}_{\mathsf{ck}}$.

This completes the proof. □

Batched Commitment Key Verification, $0/1 \leftarrow \mathsf{CKVer}(\mathsf{gk}, \mathsf{ck})$: Batched CKVer is a probabilistic algorithm that, given commitment key ck and public setup information gk (that can be computed locally), does the following verifications on ck elements,

- Parse or recompute $\mathsf{gk} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2) \leftarrow \mathsf{BGgen}(1^\lambda)$;
- Samples three vector of randomnesses with length n as $\boldsymbol{r}, \boldsymbol{s}, \boldsymbol{t} \leftarrow_\$ \{1, \ldots, 2^\lambda\}$;
- If $\left([\hat{a}]_1 + \sum_{i=1}^n r_i \left[x^i\right]_1 + \sum_{i=1}^n s_i \left[\hat{a}x^i\right]_1\right) \bullet [1]_2 + [\hat{a}]_1 \bullet \sum_{i=1}^n t_i \left[x^i\right]_2 = [1]_1 \bullet \left([\hat{a}]_2 + \sum_{i=1}^n r_i \left[x^i\right]_2 + \sum_{i=1}^n s_i \left[\hat{a}x^i\right]_2 + \sum_{i=1}^n t_i \left[\hat{a}x^i\right]_2\right)$, then **return** 1 (the ck is well-formed);
- Else, **return** 0 (the ck is not well-formed);

Fig. 4: Batched CKVer algorithm for sub-equivocal commitment scheme in Fig. 1

Discussions. In summary, the result of Thm. 2 shows that one can construct commitment schemes that allows a committer to achieve equivocality without trusting to the key generator (a third party). But to achieve this, the committer requires to check the well-formedness of commitment keys before using them in algorithm $\mathsf{Com}_{\mathsf{ck}}$. To do so, a committer only needs to execute CKVer algorithm on ck elements. Note that if a committer will use commitment keys several times, he only requires to check the keys once. One may also note that the information-theoretical variant of equivocality hold even for adversarial keys.

Remark 1. In practice, executing CKVer algorithm on long commitment keys might take considerable time. In such cases, to make CKVer more efficient, one can use batching techniques [BGR98,HHK+17] to speed up the verification.

Particularly for the sub-equivocal commitment scheme given in Fig. 1, in order to execute CKVer, one needs to compute $5n + 2$ parings (note that right hand of verifications in fourth item, already are computed in third item). But with batching techniques, one can write a batched form of CKVer algorithm as in Fig. 4 and decrease the number of paring equations considerably. For instance, with batched CKVer algorithm in Fig. 4, one can verify public commitment keys of sub-equivocal commitment scheme in Fig. 1 with only 3 parings and $6n$ exponentiations, that for large values of n, this takes considerably less time.

5.2 Sub-binding and Sub-hiding

In this section, we show the second positive result which states that we can construct commitment schemes that achieve sub-hiding and sub-binding at the same time, but not equivocality.

Let $\Pi_{com}^{2\text{-}party} = (\mathsf{KGen}, \mathsf{Com}_{\mathsf{ck}}, \mathsf{OpenVer}_{\mathsf{ck}})$ be a commitment scheme that its KGen is *trivial*, meaning that committer and verifier can ignore its output. We prove that if there exists such hiding and binding commitment scheme, then it also guarantees sub-hiding and sub-binding. Generally speaking, this covers commitment schemes that do not require (particular) setup phase, except choosing some system parameters that can be agreed in between both parties jointly, e.g.

agreeing on order and generator of the underlying group or a particular standard hash function. Intuitively, one can see that if ck is trivial, e.g. it is empty $\mathsf{ck} = \{\}$, so there is no risk of subverting. So, basically the setup-less commitment schemes belong to this family.

Lemma 1. *Let $\Pi_{com}^{2-party}$ be a commitment scheme with trivial ck. If $\Pi_{com}^{2-party}$ satisfies binding and hiding, it also guarantees sub-binding and sub-hiding.*

Proof. Let $\mathcal{A}$ be a sub-binding adversary, meaning that

$$\Pr\begin{bmatrix}(\mathsf{ck}, c, (m_0, \mathsf{op}_0), (m_1, \mathsf{op}_1)) \leftarrow \mathcal{A}(\mathsf{gk}) : \\ \mathsf{CKVer}(\mathsf{gk}, \mathsf{ck}) = 1 \wedge (\mathsf{OpenVer}_{\mathsf{ck}}(c, m_0, \mathsf{op}_0) = 1) \wedge \\ (\mathsf{OpenVer}_{\mathsf{ck}}(c, m_1, \mathsf{op}_1) = 1) \wedge (m_0 \neq m_1)\end{bmatrix} \approx_\lambda 1 - \mathsf{negl}(\lambda) \quad .$$

Now by considering the fact that in a $\Pi_{com}^{2-party}$ commitment scheme, meaning that it has trivial ck, so its commitment key can be generated by either adversary or the honest KGen. So in above game we can substitute malicious key generator $\mathcal{A}$ in the setup phase with a honest KGen, meaning that

$$\Pr\begin{bmatrix}\mathsf{ck} \leftarrow \mathsf{KGen}(\mathsf{gk}), (c, (m_0, \mathsf{op}_0), (m_1, \mathsf{op}_1)) \leftarrow \mathcal{A}(\mathsf{ck}) : \\ \mathsf{CKVer}(\xi, \mathsf{ck}) = 1 \wedge (\mathsf{OpenVer}_{\mathsf{ck}}(c, m_0, \mathsf{op}_0) = 1) \wedge \\ (\mathsf{OpenVer}_{\mathsf{ck}}(c, m_1, \mathsf{op}_1) = 1) \wedge (m_0 \neq m_1)\end{bmatrix} \approx_\lambda 1 - \mathsf{negl}(\lambda) \quad ,$$

which gives us a new successful adversary for binding of the commitment scheme $\Pi_{com}^{2-party}$. As a result, if $\Pi_{com}^{2-party}$ guarantees binding, so it is also sub-binding.

Similarly, let $\mathcal{A}$ be a sub-hiding adversary, meaning that

$$\Pr\begin{bmatrix}(\mathsf{ck}, (m_0, m_1)) \leftarrow \mathcal{A}(\mathsf{gk}), b \leftarrow\!\!\$\ \{0,1\}, \mathsf{CKVer}(\mathsf{gk}, \mathsf{ck}) = 1, \\ (c_b, \mathsf{op}_b) \leftarrow \mathsf{Com}_{\mathsf{ck}}(\mathsf{ck}, m_b), b' \leftarrow \mathcal{A}(\mathsf{ck}, c_b) : b' = b\end{bmatrix} \approx_\lambda 1 - \mathsf{negl}(\lambda) \quad .$$

Similar to previous case, by considering the property of a $\Pi_{com}^{2-party}$ commitment scheme, we can substitute malicious key generator $\mathcal{A}$ in the setup phase with an honest KGen, which results,

$$\Pr\begin{bmatrix}\mathsf{ck} \leftarrow \mathsf{KGen}(\mathsf{gk}), (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{ck}), b \leftarrow\!\!\$\ \{0,1\}, \mathsf{CKVer}(\mathsf{gk}, \mathsf{ck}) = 1, \\ (c_b, \mathsf{op}_b) \leftarrow \mathsf{Com}_{\mathsf{ck}}(\mathsf{ck}, m_b), b' \leftarrow \mathcal{A}(\mathsf{ck}, c_b) : b' = b\end{bmatrix} \approx_\lambda 1 - \mathsf{negl}(\lambda) \quad ,$$

that gives us a new successful adversary for hiding of the commitment scheme $\Pi_{com}^{2-party}$. Hence, if $\Pi_{com}^{2-party}$ guarantees binding, so it is also sub-binding. Note that generally when key generation is done honestly, CKVer always returns 1. □

Theorem 3 (Sub-hiding and Sub-binding Commitment Schemes). *Under some standard assumptions (e.g. DDH, CDH and etc), there exist commitment schemes that archive sub-hiding and sub-binding.*

Proof. Basically all setup-less commitment schemes (e.g. the ones constructed with standard hash functions) that can guarantee hiding and binding are a

$\Pi_{com}^{2-party}$ commitment scheme. As a result, by considering the result of Lemma 1, all of them can also guarantee sub-hiding and sub-binding.

As an interesting example, recently, Lombardi and Schaeffer [LS19] observed and proved that any Key Agreement (KA) protocol satisfying perfect completeness (i.e. if both parties are honest, the final agreed keys are the same), can be used to construct a non-interactive commitment scheme which guarantees (perfectly) binding and (computationally) hiding and does not need a particular setup phase. The implication works for all KA protocols independent of number of rounds in the protocol. Intuitively, they observed that the transcript of a KA protocol can be considered as a commitment to the final agreed key [4]. The commitment schemes constructed from such KA protocols can also be considered as a $\Pi_{com}^{2-party}$ commitment scheme, and consequently a sub-binding and sub-hiding commitment scheme [LS19]. □

5.3 Binding, Equivocality and Sub-hiding

We now consider the last positive result in Tab. 1 which states that we can have a commitment scheme that achieves hiding, equivocality, binding, and sub-hiding at the same time. This result might look redundant in comparison with first positive result. Since by considering the relation between standard and new notions of a commitment scheme, described in Tab. 1, one can see that this result can be concluded from first positive result, as sub-equivocality implies sub-hiding and equivocality. But the point is that the proof of first positive result is established under the knowledge assumption BDH-KE (a non-falsifiable assumption), but here we show that one can still achieve sub-hiding under standard assumptions (falsifiable assumptions) by requiring that there exist hiding, binding and equivocal commitment schemes.

Pedersen Commitment Scheme Achieves Sub-hiding. We show the Pedersen commitment scheme [Ped92] can guarantee sub-hiding property with minimal checking. More accurately, the committer only needs to run the CKVer algorithm to verify ck before using the key for committing, and luckily the checking for this scheme is quite simple. A variation of Pedersen commitment scheme that archives sub-hiding under standard assumptions can be expressed as a tuple of algorithms $\Pi_{com}^{sub-ped} = (\mathsf{KGen}, \mathsf{CKVer}, \mathsf{Com}_{\mathsf{ck}}, \mathsf{OpenVer}_{\mathsf{ck}}, \mathsf{Sim.KGen}, \mathsf{Com}^{*}_{\mathsf{tk}}, \mathsf{Equiv}_{\mathsf{tk}})$ described in Fig. 5. Note that we are interested to active new notion by retaining already achieved ones. It is worth to mention that, since here we assume the commitment scheme achieves equivocality by trusting to the key generator, so we used $\mathsf{Sim.KGen}$ instead of $\mathsf{Sim}.\mathcal{A}$ from the algorithms listed in $\Pi_{com}^{sub-ped}$.

[4] Let $\Pi_{\mathsf{KA}}^{A,B}$ be a *complete* KA protocol between parties A and B, with random inputs x_A and y_B, respectively. The committing phase of the by-result commitment scheme will be as the following: on input a bit b, random inputs x_A and y_B for an execution of $\Pi_{\mathsf{KA}}^{A,B}$; compute the transcript τ of the KA protocol $\Pi_{\mathsf{KA}}^{A,B}$ using x_A and x_B, and return c as commitment, $c := (\tau, b \oplus k)$, where k is the final agreed key [LS19].

Parameter Generation, $\mathsf{gk} \leftarrow \mathsf{BGgen}(1^\lambda)$**:** Given security parameter 1^λ, generates the description of a group $\mathbb{G}$ of prime order p and return $\mathsf{gk} := (\mathbb{G}, p)$.

Key Generation, $\mathsf{ck} \leftarrow \mathsf{KGen}(\mathsf{gk})$**:** Generates two generators $(g, h) \leftarrow \mathbb{G}^2$, where h also can be taken as $h := g^{\mathsf{sk}}$, where in such case sk would be the secret trapdoor of the commitment key ck. We let $\mathsf{ck} := (g, h)$;

Commitment Key Verification, $0/1 \leftarrow \mathsf{CKVer}(\mathsf{gk}, \mathsf{ck})$**:** Given setup information gk and commitment key ck, check if both $g, h \in \mathbb{G}$ and returns 1 if both $g \neq 0$ and $h \neq 0$, else return 0.

Committing, $(c, \mathsf{op}) \leftarrow \mathsf{Com}_{\mathsf{ck}}(m; r)$**:** Given (ck, m) for $\mathsf{CKVer}(\mathsf{gk}, \mathsf{ck}) = 1$, where $m \in \mathbb{Z}_p$, it samples a randomness $r \leftarrow \mathbb{Z}_p$ and computes the commitment $(c, \mathsf{op}) := (g^m h^r, r)$.

Opening Verification, $0/1 \leftarrow \mathsf{OpenVer}_{\mathsf{ck}}(c, m, \mathsf{op})$**:** Outputs 1 if $g^m h^r = c$, else 0;

Simulation of Key Generation, $(\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{Sim.KGen}(\mathsf{gk})$**:** Generates two generators $g \leftarrow \mathbb{G}$ and $h = g^{\mathsf{sk}}$, where $\mathsf{sk} \leftarrow \mathbb{Z}_p$ would be the secret trapdoor of the commitment key ck. We let $(\mathsf{ck}, \mathsf{tk}) := ((g, h), \mathsf{sk})$;

Trapdoor Committing, $(c, \mathsf{ek}) \leftarrow \mathsf{Com}^*_{\mathsf{tk}}(\mathsf{ck})$**:** Given commitment key ck and tk as input, outputs an equivocal commitment $c = \mathsf{Com}_{\mathsf{ck}}(0; s) = g^s$ where $s \leftarrow \mathbb{Z}_p$ and an equivocation key $\mathsf{ek} = s$.

Trapdoor Opening, $r \leftarrow \mathsf{Equiv}_{\mathsf{tk}}(c, m)$**:** On input equivocation key $\mathsf{ek} = s \in \mathbb{Z}_p$, $c \in \mathcal{C}_{\mathsf{ck}}$ and messages m creates an opening $r \leftarrow (s - m) \cdot \mathsf{sk}^{-1}$, such that $c = \mathsf{Com}_{\mathsf{ck}}(m; r)$, and returns r.

Fig. 5: Pedersen commitment scheme with sub-hiding.

Theorem 4 (Subversion-Resistant Pedersen Commitment). *The Pedersen commitment scheme with the construction described in Fig. 5 is hiding, equivocal, binding and sub-hiding under the discrete logarithm assumption in* $\mathbb{G}$.

Proof. For the sub-hiding property, once $\mathsf{CKVer}(\mathsf{gk}, \mathsf{ck})$ returned 1, we conclude that both g and h are non-zero group elements, so one can notice that upon random choice of $r \in \mathbb{Z}_p$, for any $m \in \mathbb{Z}_p$, $c = g^m h^r$ is uniformly distributed over $\mathbb{G}$. For the binding property, as the non-subversion resistant version, one can observe that given openings (r_0, r_1) for a commitment c to distinct messages (m_0, m_1), the relation $g^{m_0} h^{r_0} = g^{m_1} h^{r_1}$ leads to $h = g^{\frac{m_0 - m_1}{r_1 - r_0}}$, which gives the discrete logarithm of h in base g. Intuitively, if the discrete logarithm problem is hard, the commitment scheme is (computationally) binding. For equivocality, as original scheme we can see that given trapdoor tk of the commitment key ck, one can generate a fake commitment and the corresponding valid opening using $\mathsf{Com}^*_{\mathsf{tk}}$ and $\mathsf{Equiv}_{\mathsf{tk}}$ described in Fig. 5. This results the theorem. □

To sum up, the Thm. 4 shows that, even if a malicious key generator generates the commitment key ck for Pedersen commitment scheme, still committer can achieve hiding (sub-hiding) by some simple checking. But to guarantee equivocality, still committer needs to trust the key generator or use a commitment scheme as described in 1.

Acknowledgment. This work was supported in part by the Estonian Research Council grant PRG49.

References

ABK18. Benedikt Auerbach, Mihir Bellare, and Eike Kiltz. Public-key encryption resistant to parameter subversion and its realization from efficiently-embeddable groups. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 348–377. Springer, Heidelberg, March 2018.

AMV15. Giuseppe Ateniese, Bernardo Magri, and Daniele Venturi. Subversion-resilient signatures: Definitions, constructions and applications. Cryptology ePrint Archive, Report 2015/517, 2015. `http://eprint.iacr.org/2015/517`.

BBG+13. James Ball, Julian Borger, Glenn Greenwald, et al. Revealed: how us and uk spy agencies defeat internet privacy and security. *The Guardian*, 6:2–8, 2013.

BCG+14. Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014.

BFM88. Manuel Blum, Paul Feldman, and Silvio Micali. Non-Interactive Zero-Knowledge and Its Applications. In *STOC 1988*, pages 103–112, Chicago, Illinois, USA, May 2–4, 1988. ACM Press.

BFS16. Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. NIZKs with an untrusted CRS: Security in the face of parameter subversion. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 777–804. Springer, Heidelberg, December 2016.

BGR98. Mihir Bellare, Juan A. Garay, and Tal Rabin. Batch verification with applications to cryptography and checking. In Claudio L. Lucchesi and Arnaldo V. Moura, editors, *LATIN 1998*, volume 1380 of *LNCS*, pages 170–191. Springer, Heidelberg, April 1998.

BL07. Ahto Buldas and Sven Laur. Knowledge-binding commitments with applications in time-stamping. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 150–165. Springer, Heidelberg, April 2007.

Blu81. Manuel Blum. Coin flipping by telephone. In Allen Gersho, editor, *CRYPTO'81*, volume ECE Report 82-04, pages 11–15. U.C. Santa Barbara, Dept. of Elec. and Computer Eng., 1981.

BPR14. Mihir Bellare, Kenneth G. Paterson, and Phillip Rogaway. Security of symmetric encryption against mass surveillance. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 1–19. Springer, Heidelberg, August 2014.

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

CGGN17. Matteo Campanelli, Rosario Gennaro, Steven Goldfeder, and Luca Nizzardo. Zero-knowledge contingent payments revisited: Attacks and payments for services. Cryptology ePrint Archive, Report 2017/566, 2017. `http://eprint.iacr.org/2017/566`.

CIO98. Giovanni Di Crescenzo, Yuval Ishai, and Rafail Ostrovsky. Non-Interactive and Non-Malleable Commitment. In Jeffrey Scott Vitter, editor, *STOC 1998*, pages 141–150, Dallas, Texas, USA, May 23–26, 1998.

Dam90. Ivan Damgård. On the existence of bit commitment schemes and zero-knowledge proofs. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 17–27. Springer, Heidelberg, August 1990.

Dam92. Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 445–456. Springer, Heidelberg, August 1992.

DF02. Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 125–142. Springer, Heidelberg, December 2002.

DGP+19. Vanesa Daza, Alonso González, Zaira Pindado, Carla Ràfols, and Javier Silva. Shorter quadratic QA-NIZK proofs. In *Public-Key Cryptography - PKC 2019 - 22nd IACR International Conference on Practice and Theory of Public-Key Cryptography, Beijing, China, April 14-17, 2019, Proceedings, Part I*, pages 314–343, 2019.

DIO98. Giovanni Di Crescenzo, Yuval Ishai, and Rafail Ostrovsky. Non-interactive and non-malleable commitment. In *30th ACM STOC*, pages 141–150. ACM Press, May 1998.

EGL85. Shimon Even, Oded Goldreich, and Abraham Lempel. A Randomized Protocol for Signing Contracts. *Communications of the ACM*, 28(6):637–647, June 1985.

EHK+13. Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, August 2013.

FF00. Marc Fischlin and Roger Fischlin. Efficient non-malleable commitment schemes. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 413–431. Springer, Heidelberg, August 2000.

FLZ16. Prastudy Fauzi, Helger Lipmaa, and Michal Zajac. A shuffle argument secure in the generic model. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 841–872. Springer, Heidelberg, December 2016.

FMMO18. Prastudy Fauzi, Sarah Meiklejohn, Rebekah Mercer, and Claudio Orlandi. Quisquis: A new design for anonymous cryptocurrencies. *IACR Cryptology ePrint Archive*, 2018:990, 2018.

Fuc17. Georg Fuchsbauer. Subversion-zero-knowledge SNARKs. Cryptology ePrint Archive, Report 2017/587, 2017. `http://eprint.iacr.org/2017/587`.

Gab19. Ariel Gabizon. On the security of the BCTV pinocchio zk-snark variant. *IACR Cryptology ePrint Archive*, 2019:119, 2019.

GGJS11. Sanjam Garg, Vipul Goyal, Abhishek Jain, and Amit Sahai. Bringing people of different beliefs together to do UC. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 311–328. Springer, Heidelberg, March 2011.

GL89. Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *21st ACM STOC*, pages 25–32. ACM Press, May 1989.

GL07. Jens Groth and Steve Lu. Verifiable shuffle of large size ciphertexts. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 377–392. Springer, Heidelberg, April 2007.

GMW87. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In *STOC 1987*, pages 218–229, New York City, 25–27 May 1987.

GMW91. Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *Journal of the ACM*, 38(3):691–729, 1991.

GO07. Jens Groth and Rafail Ostrovsky. Cryptography in the multi-string model. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 323–341. Springer, Heidelberg, August 2007.

GOS06. Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 97–111. Springer, Heidelberg, August 2006.

GQ88. Louis C. Guillou and Jean-Jacques Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both trasmission and memory. In C. G. Günther, editor, *EUROCRYPT'88*, volume 330 of *LNCS*, pages 123–128. Springer, Heidelberg, May 1988.

Gre14. Glenn Greenwald. *No place to hide: Edward Snowden, the NSA, and the US surveillance state*. Macmillan, 2014.

Gro05. Jens Groth. Non-interactive zero-knowledge arguments for voting. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *ACNS 05*, volume 3531 of *LNCS*, pages 467–482. Springer, Heidelberg, June 2005.

Gro09. Jens Groth. Homomorphic trapdoor commitments to group elements. Cryptology ePrint Archive, Report 2009/007, 2009. `http://eprint.iacr.org/2009/007`.

Gro10. Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Heidelberg, December 2010.

GS08. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008.

Hae19. Rolf Haenni. Swiss post public intrusion test: Undetectable attack againstvote integrity and secrecy `https://e-voting.bfh.ch/app/download/7833162361/PIT2.pdf?t=1552395691`. 2019.

HHK+17. Gottfried Herold, Max Hoffmann, Michael Klooß, Carla Ràfols, and Andy Rupp. New techniques for structural batch verification in bilinear groups with applications to groth-sahai proofs. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1547–1564. ACM Press, October / November 2017.

HILL99. Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. 1999.

KKZZ14. Jonathan Katz, Aggelos Kiayias, Hong-Sheng Zhou, and Vassilis Zikas. Distributing the setup in universally composable multi-party computation. In Magnús M. Halldórsson and Shlomi Dolev, editors, *33rd ACM PODC*, pages 20–29. ACM, July 2014.

KMS+16. Ahmed E. Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE Symposium on Security and Privacy*, pages 839–858. IEEE Computer Society Press, May 2016.

Lip12. Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor,

TCC 2012, volume 7194 of *LNCS*, pages 169–189. Springer, Heidelberg, March 2012.

Lip16. Helger Lipmaa. Prover-Efficient Commit-And-Prove Zero-Knowledge SNARKs. In David Pointcheval, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *AFRICACRYPT 2016*, volume 9646 of *LNCS*, pages 185–206, Fes, Morocco, April 13–15, 2016. Springer, Heidelberg.

LPT19. Sarah Jamie Lewis, Olivier Pereira, and Vanessa Teague. Trapdoor commitments in the swisspost e-voting shuffle proof, `https://people.eng.unimelb.edu.au/vjteague/SwissVote`. 2019.

LS19. Alex Lombardi and Luke Schaeffer. A note on key agreement and non-interactive commitments. *IACR Cryptology ePrint Archive*, 2019:279, 2019.

MP12. Mohammad Mahmoody and Rafael Pass. The curious case of non-interactive commitments - on the power of black-box vs. non-black-box use of primitives. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 701–718. Springer, Heidelberg, August 2012.

Nao91. Moni Naor. Bit Commitment using Pseudorandom Generators. *J. Cryptology*, 4(2):151–158, 1991.

Ped92. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992.

PLS13. Nicole Perlroth, Jeff Larson, and Scott Shane. Nsa able to foil basic safeguards of privacy on web. *The New York Times*, 5, 2013.

RTYZ16. Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Cliptography: Clipping the power of kleptographic attacks. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 34–64. Springer, Heidelberg, December 2016.

Sim83. Gustavus J. Simmons. The prisoners' problem and the subliminal channel. In David Chaum, editor, *CRYPTO'83*, pages 51–67. Plenum Press, New York, USA, 1983.

Sim85. Gustavus J. Simmons. The subliminal channel and digital signature. In Thomas Beth, Norbert Cot, and Ingemar Ingemarsson, editors, *EUROCRYPT'84*, volume 209 of *LNCS*, pages 364–378. Springer, Heidelberg, April 1985.

Wik09. Douglas Wikström. A commitment-consistent proof of a shuffle. In Colin Boyd and Juan Manuel González Nieto, editors, *ACISP 09*, volume 5594 of *LNCS*, pages 407–421. Springer, Heidelberg, July 2009.

Yao82. Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91, 1982.

YY96. Adam Young and Moti Yung. The dark side of "black-box" cryptography, or: Should we trust capstone? In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 89–103. Springer, Heidelberg, August 1996.

YY97. Adam Young and Moti Yung. The prevalence of kleptographic attacks on discrete-log based cryptosystems. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 264–276. Springer, Heidelberg, August 1997.