

Subversion-Resistant Commitment Schemes: Definitions and Constructions

Karim Baghery^{1,2} [0000-0001-7213-8496]

¹ imec-COSIC, KU Leuven, Leuven, Belgium

² University of Tartu, Tartu, Estonia
karim.baghery@kuleuven.be

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, (equivocal) commitment schemes require a setup phase which is supposed to be done by a third trusted party. Recently, various news is reported about the 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 and breaking users' privacy, given the trapdoor [Hae19,LPT19]. Motivated by such news and recent studies on subversion-resistance of various cryptographic primitives, this research studies the security of commitment schemes in the presence of a maliciously chosen commitment key. To attain a clear understanding of achievable security, we define a variety 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, reducing trust, CRS model

1 Introduction

The notion of commitment [Blu81] is one of the fundamental and widely used concepts in cryptography. A commitment scheme allows a committer to create a commitment c to a secret value of m , and later open the commitment c in a verifiable manner [GQ88,Ped92]. The procedure of generating c is called *committing* phase, and revealing or giving a proof-of-knowledge of message m and some secret information used in committing phase (precisely, randomnesses) called *opening*. In the Common Reference String (CRS) 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 (NI) commitment schemes [Nao91,CIO98,HILL99]. During last decades, we have seen various elegant NI 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 by maliciously (or incorrectly) generated public parameters or modifying cryptographic protocol specifications to embed backdoors, with intent to violate the 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, which allows breaking security of the system without being detected. Indeed, the used commitment scheme has a trapdoor that having access to that, one can alter the votes or can break voters' privacy. So, given such a trapdoor, a malicious party can do an undetectable vote manipulation by an authority who sets up the mixing network¹.

To deal with such concerns, a particular subfield of cryptography was proposed which is known as kleptography [YY96,YY97,Sim83,Sim85,RTYZ16] that allows an adversary to replace a cryptographic algorithm with an altered version with intend of subverting its security. To construct practical systems secure against such adversary, one needs to develop protocols that can guarantee security against parameter subversion. Meaning that new cryptographic systems should provide their pre-defined security with trusted parameters, but even if the parameters subverted, the system can guarantee a level of security. As a common way, some works [GO07,GGJS11,KKZZ14] use MPC protocols for malicious parameter generation which mitigates the trust on the setup phase. Initiated by Bellare et al. [BPR14], recently subversion security has gotten considerable attention with focus on different cryptographic primitives including symmetric encryption schemes [BPR14], signature schemes [AMV15], non-interactive zero-knowledge proofs [BFS16,ABLZ17,Fuc18,Bag19], and public-key encryption schemes [ABK18]. Each of them considers achievable security in a particular family of primitives under subverted parameters. NI commitment schemes in the CRS model are another prominent family of primitives that require a trusted setup phase [Ped92,DF02,GOS06,Gro09,Gro10,Lip12]. As such commitment schemes are deployed in various areas of cryptography, so their security is not only important on itself but also security of other practical systems relies on it (e.g. guaranteeing the security of shuffling in mix-net of SwissPost-Scytl). Thus, their security under subverting public commitment key can have a crucial effect on the security of the bigger systems.

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

Our Contribution. We study the resistance of NI commitment schemes in the case of subverting commitment key and present definitions, negative results, along with some Subversion-Resistant (Sub-R) 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. To get a clear understanding of achievable security, we first present new variations of current definitions, that are defined to guarantee the security of committers and verifiers even if the setup phase of a commitment scheme is subverted. Presenting subversion-resistant definitions for commitment schemes is the first contribution of this paper.

Recall that in the CRS model, an equivocal NI commitment scheme Π_{com} (e.g. [Gro10,Lip12]) is expected to satisfy, 1) Hiding: It is hard for any PPT adversary \mathcal{A} , given an honestly generated commitment key ck , to generate two messages $m_0 \neq m_1$ from message space \mathcal{M} such that \mathcal{A} can distinguish between their corresponding commitments c_0 and c_1 . 2) Binding: It is hard for any PPT adversary \mathcal{A} , given an honestly generated commitment key ck , to come up with a collision $(c, m_0, op_0, m_1, op_1)$, such that op_0 and op_1 are valid opening values of two different pre-images $m_0 \neq m_1$ for c , 3) Equivocality: Given the trapdoor tk associated with ck , it is possible to create a fake commitment that can be opened successfully. Equivocality implies hiding, as a commitment is indistinguishable from an equivocal commitment which can be opened to any message.

Commitment schemes with subverted parameters. We modify original definitions of commitments in [Gro09,Gro10,Lip12] and present a variation of them for Sub-R equivocal commitments. The key change in new definitions is that the adversary generates ck . When \mathcal{A} generates ck , it can retain some trapdoors tk as a "backdoor" associated with ck . In section 3.1, we formalize the following requirements for Sub-R commitments: 1) Sub-hiding: (subversion hiding) Even if a PPT \mathcal{A} generates ck , if the ck is *well-formed*³, it is hard for \mathcal{A} to gener-

³ Intuitively, the generated commitment key ck should have a well-defined structure.

Table 1: Summary of results. Each row refers to achievability of selected notions.

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

ate $m_0, m_1 \in \mathcal{M}$ s.t. \mathcal{A} can distinguish between their corresponding commitments c_0 and c_1 where $(c_0, \text{op}_0) \leftarrow \text{Com}(\text{ck}, m_0; r)$ and $(c_1, \text{op}_1) \leftarrow \text{Com}(\text{ck}, m_1; r)$.
 2) Sub-binding: (subversion binding) Even if a PPT \mathcal{A} generates ck , if the ck is *well-formed*, it is hard for \mathcal{A} to come up with a collision $(c, m_0, \text{op}_0, m_1, \text{op}_1)$, s.t. op_0 and op_1 are valid opening values of two different pre-images $m_0 \neq m_1$ for c .
 3) Sub-equivocality: (subversion equivocality) Even if a PPT \mathcal{A} generates ck , the scheme *still* should satisfy equivocality.

The relations between standard and new notions are shown in Fig. 1. Subversion-resistant variations imply the standard ones, as in the standard cases the setup phase is trusted. For instance, sub-equivocality implies equivocality, and as already mentioned equivocality implies hiding, as a commitment is indistinguishable from an equivocal commitment that can be opened to any message.

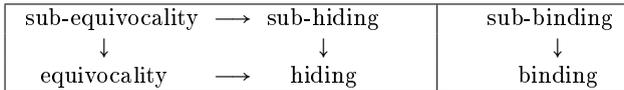


Fig. 1: Relation between current and new defined subversion-resistant notions.

Next, we consider how much subversion security is achievable in NI commitments. Our key results are summarized in Tab. 1. Each row considers constructing schemes that simultaneously can achieve the indicated notions (by ✓).

Negative result. We first consider whether we can achieve sub-binding along with the current notions, namely *sub-binding*, binding, hiding, and equivocality. The negative result in Tab. 1, indicates that we cannot achieve even standard equivocality and sub-binding at the same time. In Sec. 4, we show that achieving equivocality is in contradiction with achieving sub-binding.

Positive results. Positive 1: By considering the negative result, the next best scenario would be the case that one can achieve all notions but sub-binding. In Thm. 2, we show that this case is possible, and we present a Sub-R commitment scheme that can achieve the notions indicated in the first positive result in Tab. 1. This result is established under the Bilinear Diffie-Hellman Knowledge of Exponents (BDH-KE) assumption, defined in Def. 2, and Γ -Power Symmetric Discrete Logarithm Assumption (Γ -PSDL), defined in Def. 1, assumptions in a group equipped with bilinear map. Positive 2: Next, we consider 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. Second positive result in Tab. 1 which is established in Thm. 3, shows that we can construct such commitment schemes. We show that basically this includes already known results that one can construct a hiding and binding

commitment in the standard model. **Positive 3:** The third scenario is a commonly used case in practice. The scheme already satisfies hiding, equivocalty, and binding and when we consider the case that keys are generated maliciously, it does not break completely, and indeed it still achieves hiding. In Sec. 5.3, we show that with minimal checks, Pedersen [Ped92] commitment scheme can achieve sub-hiding. This result might look redundant, as it is a restricted form of *Positive 1*, but this result is established entirely under standard assumptions.

In many cryptographic systems, it is shown the deployed commitment require equivocalty, especially in minimizing the round complexity of zero-knowledge proofs [BFM88], or even constructing efficient NI zero-knowledge proofs [GS08,Gro10,Lip12]. A direct observation of the first positive result is that sub-equivocalty can decrease the needed trust in such proof systems [Lip12].

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

Future research directions. Here we have focused on three notions hiding, equivocalty, and binding. With a similar approach one may also consider stronger notions such as non-malleability [DIO98,FF00] about commitment schemes. Such research question about NIZK arguments is studied in [Bag19]. We also found constructing commitment schemes in the updatable CRS model [GKM⁺18] as an interesting (future) research question. Such sort of commitment schemes would allow both the committer and the verifier to update the public parameters and bypass the needed trust in the standard CRS model.

2 Preliminaries

Let $\lambda \in \mathbb{N}$ be the security parameter, and 1^λ denotes its unary representation; say $\lambda = 128$. $s \leftarrow_s S$ denotes picking s uniformly random from S . The empty string is shown with $\{\}$, e.g. $\text{ck} = \{\}$. For an algorithm \mathcal{A} , let $\text{im}(\mathcal{A})$ be the image of \mathcal{A} , i.e., the set of valid outputs of \mathcal{A} , let $\text{RND}(\mathcal{A})$ denote the random tape of \mathcal{A} , and let $r \leftarrow_s \text{RND}(\mathcal{A})$ 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 . Note that $\text{Ext}_{\mathcal{A}}$ and \mathcal{A} use internally the same randomness r . We denote by $\text{negl}(\lambda)$ an arbitrary negligible function, and by $\text{poly}(\lambda)$ an arbitrary polynomial function. For a tuple of integers $\Gamma = (\gamma_1, \dots, \gamma_n)$ with $\gamma_i \leq \gamma_{i+1}$, let $(a_i)_{i \in \Gamma} = (a_{\gamma_1}, \dots, a_{\gamma_n})$. We sometimes denote $(a_i)_{i \in [n]}$ as \mathbf{a} . We say that $\Gamma = (\gamma_1, \dots, \gamma_n) \in \mathbb{Z}^n$ is an (n, λ) -nice tuple, if $0 \leq \gamma_1 \leq \dots \leq \gamma_i \leq \gamma_n = \text{poly}(\lambda)$. In games, $\Pr[G : y]$ shows the probability that y happens for the game G .

In pairing-based groups, we use additive notation together with the bracket notation, i.e., in group \mathbb{G}_μ , $[a]_\mu = a[1]_\mu$, where $[1]_\mu$ is a fixed generator of \mathbb{G}_μ . A *bilinear group generator* $\text{BGgen}(1^\lambda)$ 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, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is an efficient non-degenerate bilinear pairing, s.t. $\hat{e}([a]_1, [b]_2) = [ab]_T$. Denote $[a]_1 \bullet [b]_2 = \hat{e}([a]_1, [b]_2)$.

Definition 1 (Γ -Power (Symmetric) Discrete Logarithm Assumption).

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

$$\Pr \left[\begin{array}{l} \text{gk} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2) \leftarrow \text{BGgen}(1^\lambda), \\ x \leftarrow \mathbb{Z}_p : \mathcal{A}(\text{gk}, ([x^l]_1, [x^l]_2)_{l \in \Gamma}) = x \end{array} \right] = \text{negl}(\lambda) .$$

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

Definition 2 (BDH-KE Assumption). We say BGgen is BDH-KE secure for \mathcal{R} if for any λ , $(\mathbf{R}, \xi_{\mathbf{R}}) \in \text{im}(\mathcal{R}(1^\lambda))$, and PPT adversary \mathcal{A} there exists a PPT extractor $\text{Ext}_{\mathcal{A}}$, such that

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

is negligible in λ . In above assumption, $\xi_{\mathbf{R}}$ is the auxiliary information related to the underlying group. This is an asymmetric-pairing version of the original knowledge assumption [Dam92].

3 Security of Commitments under Parameters Subversion

Let Setup be a setup algorithm that takes as input λ and outputs some setup information $\text{gk} \leftarrow \text{Setup}(1^\lambda)$. In the basic form, a NI commitment scheme consists of a tuple of polynomial-time algorithms $(\text{KGen}, \text{Com}, \text{Ver})$. We consider equivocal commitments (a.k.a. trapdoor commitments) that consists of algorithms $(\text{KGen}, \text{Com}, \text{Ver}, \text{KGen}^*, \text{Com}^*, \text{Equiv})$. KGen is a PPT algorithm that given gk generates a ck and a trapdoor key tk . As in [Gro09], the gk can describe a finite group over which we are working, or simply the security parameter. We assume all parties have access to gk . The commitment key ck specifies a message space \mathcal{M} , a randomizer space \mathcal{R} and a commitment space \mathcal{C} . 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} . Com takes ck , a message m , a randomizer r and outputs c

and an opening op . Given ck , c , m and op , Ver returns either 1 or 0. In equivocal commitments, given tk , it is possible to open a c to any message. This property is formalized by PPT algorithms Com^* and Equiv , where Com^* takes tk (generated by KGen^*) and outputs an equivocal commitment c and an equivocation key ek . Then, Equiv on inputs ek , c and a message m creates an opening $\text{op} := r$ of c , so that $(c, \text{op}) = \text{Com}(\text{ck}, m; r)$. Security requirements for equivocal commitments under a trusted setup phase are provided in the App. A.1.

Here, we define Sub-R equivocal commitment schemes and add a new algorithm CKVer to the scheme that will be used to verify the well-formedness of ck . Next, we formally define a subversion-resistant commitment scheme and the target goals.

Definition 3 (Subversion-resistant Equivocal Commitments). *A Sub-R equivocal commitment scheme consists of eight algorithms defined as below,*

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

Commitment Key Verification, $0/1 \leftarrow \text{CKVer}(\text{gk}, \text{ck})$: *CKVer is a deterministic algorithm that given gk and ck , returns either 1 or 0;*

Committing, $(c, \text{op}) \leftarrow \text{Com}(\text{ck}, m; r)$: *Outputs a c and an opening op . It specifies a function $\text{Com} : \mathcal{M} \times \mathcal{R} \rightarrow \mathcal{C}$. Given a $m \in \mathcal{M}$, the committer picks an $r \in \mathcal{R}$ and computes the $(c, \text{op}) = \text{Com}(\text{ck}, m; r)$.*

Opening Verification, $0/1 \leftarrow \text{Ver}(\text{ck}, c, m, \text{op})$: *Outputs 1 if $m \in \mathcal{M}$ is the committed message in c with opening op , and returns 0 otherwise.*

Simulation of Key Generation, $(\text{ck}, \text{tk}) \leftarrow \text{KGen}^*(\text{gk})$: *Generates a key ck and associated trapdoor tk . It also specifies spaces \mathcal{M} , \mathcal{R} , and \mathcal{C} .*

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

Trapdoor Opening, $\text{op} \leftarrow \text{Equiv}(\text{ek}, c, m, \text{ek})$: *On inputs ek , c and an m creates an opening $\text{op} := r$ of c , s.t. $(c, \text{op}) = \text{Com}(\text{ck}, m; r)$ and returns op .*

A (subversion-resistant) commitment scheme satisfies *completeness* if for $\text{ck} \leftarrow \text{KGen}(\text{gk})$ and any honestly generated commitment of $m \in \mathcal{M}$, it successfully passes the verification, i.e., $\text{Ver}(\text{ck}, \text{Com}(\text{ck}, m; \text{op}), m, \text{op}) = 1$.

3.1 Notions for Commitments with Subverted Parameters

As mentioned above, the definitions of standard notions for equivocal commitments are given in the App. A.1. In the standard notions, a critical assumption is that the commitment key ck is honestly generated by a trusted party. But as our goal is to consider achievable security when the 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 is compromised and ck is generated by an adversary (or a subverter) rather than

via the honest algorithm KGen prescribed by Π_{com} . Also, in Sub-R commitments there is a new algorithm CKVer to verify well-formedness of ck .

In the following definitions, let Setup be an algorithm that takes as input the security parameter λ and outputs some setup information $\text{gk} \leftarrow \text{Setup}(1^\lambda)$.

Definition 4 (Subversion Hiding (Sub-Hiding)). *A commitment scheme Π_{com} satisfies computationally subversion hiding if for any PPT adversary \mathcal{A} ,*

$$\left| 2 \Pr \left[\begin{array}{l} (\text{ck}, (m_0, m_1)) \leftarrow \mathcal{A}(\text{gk}), b \leftarrow_s \{0, 1\}, \text{CKVer}(\text{gk}, \text{ck}) = 1, \\ r_b \leftarrow_s \mathcal{R}, (c_b, \text{op}_b) \leftarrow \text{Com}(\text{ck}, m_b; r_b), b' \leftarrow \mathcal{A}(c_b) : b' = b \end{array} \right] - 1 \right| = \text{negl}(\lambda) .$$

The scheme is perfectly subversion hiding if the above probability is equal to 0.

By *well-formedness* of ck we mean the CKVer will verify ck successfully.

Definition 5 (Subversion Binding (Sub-Binding)). *A commitment scheme Π_{com} satisfies computationally subversion binding if for any PPT \mathcal{A} ,*

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

The commitment is perfectly subversion binding if the probability is equal to 0.

Intuitively, *subversion binding* states that an adversary \mathcal{A} will not be able to do double open a commitment c , even if it generates the (well-formed) key ck .

Definition 6 (Subversion Equivocality (Sub-Equivocality)). *A commitment scheme Π_{com} satisfies subversion equivocality if for any PPT \mathcal{A} ,*

$$\left| \Pr \left[\begin{array}{l} (\text{ck}, m) \leftarrow \mathcal{A}(\text{gk}), r \leftarrow_s \mathcal{R}, \\ (c, \text{op}) \leftarrow \text{Com}(\text{ck}, m; r) : \\ \mathcal{A}(\text{ck}, c, \text{op}) = 1 \wedge \\ \text{CKVer}(\text{gk}, \text{ck}) = 1 \end{array} \right] - \Pr \left[\begin{array}{l} (\text{ck}, \text{tk}) \leftarrow \text{KGen}^*(\text{gk}), m \leftarrow \mathcal{M} \\ (c, \text{ek}) \leftarrow \text{Com}^*(\text{ck}, \text{tk}), \\ \text{op} \leftarrow \text{Equiv}(\text{ek}, c, m) : \\ \mathcal{A}(\text{ck}, c, \text{op}) = 1 \wedge \\ \text{CKVer}(\text{gk}, \text{ck}) = 1, \end{array} \right] \right| \leq \text{negl}(\lambda) ,$$

where \mathcal{A} outputs $m \in \mathcal{M}$ and KGen^ is a key generator which also returns tk .*

Intuitively, subversion equivocality states that even if \mathcal{A} (a malicious key generator) generates the ck , still the scheme satisfies equivocality. One may notice that sub-equivocality implies sub-hiding and standard equivocality.

Lemma 1. *A commitment scheme that satisfies a security notion with subvertible setup also satisfies the security notion with honest setup.*

Proof. To prove the lemma, we show that an adversary \mathcal{A} against an honest setup can be used to construct an adversary \mathcal{B} against a subvertible setup.

Adversary \mathcal{B} first samples a ck honestly, i.e., $\text{ck} \leftarrow \text{KGen}(\text{gk})$ and checks that $\text{CKVer}(\text{gk}, \text{ck}) = 1$. Next, sends ck to \mathcal{A} and gets the answer and sends it to the challenger. Similarly, follows the rest of experiment and wins the game of subversion security with the same probability as \mathcal{A} wins the standard game. \square

4 Sub-binding with Equivocability are not Compatible

In this section, we consider if we can achieve sub-binding without degrading hiding, binding, equivocality. Achieving sub-binding individually is possible (e.g. by sending a plain message) but such a scheme will not guarantee equivocality. Here we consider the practically-interested cases. We first consider the achievability of sub-binding and (sub-)equivocality at the same time. We show that achieving simultaneously sub-binding and (even) standard equivocality is impossible.

Theorem 1 (Impossibility of Sub-binding along with Equivocality).

There cannot exist a CRS-based commitment scheme $\Pi_{com} = (\text{KGen}, \text{CKVer}, \text{Com}, \text{Ver}, \text{KGen}^, \text{Com}^*, \text{Equiv})$ which can satisfy equivocality and sub-binding at the same time.*

Proof. Sketch. The definition of equivocality (in App. A.1) states that there exists KGen^* that given gk returns (ck, tk) , and given trapdoor tk there exist two algorithms Com^* and Equiv that allow one to create a fake commitment and a valid opening which are indistinguishable from an honestly generated commitment and opening. So, given those algorithms, an adversary of sub-binding can first generate ck and tk honestly. Then, it gives ck and tk as input to Com^* and calculates $(c, \text{ek}) \leftarrow \text{Com}^*(\text{ck}, \text{tk})$. After that, it samples $(m_0, m_1) \in \mathcal{M}$, where $m_0 \neq m_1$ and invokes the algorithm Equiv twice for two different messages, and generates $\text{op}_0 \leftarrow \text{Equiv}(\text{ek}, c, m_0)$ and $\text{op}_1 \leftarrow \text{Equiv}(\text{ek}, c, m_1)$ and sends $(c, (m_0, \text{op}_0), (m_1, \text{op}_1))$ to the challenger of sub-binding game and wins with probability 1, as each of the tuples (m_0, op_0) and (m_1, op_1) are a (distinct) valid opening for c . On the other hand, sub-binding requires that \mathcal{A} should not be able to double open even if he generates the ck . But, one can observe that achieving equivocality implies that given tk one can use Com^* and Equiv and generate two valid opening with different messages which will break sub-binding.

That was the key idea behind the proof, and the full proof is provided in the App. A.2. \square

5 Positive Results

Next, we consider if we can construct subversion-resistant commitment schemes in the CRS model, which without losing current security guarantees will achieve some of the subversion-resistant notions defined in section 3.1. For instance, can we achieve sub-equivocality without losing the initial properties? We answer this question positively in subsection 5.1, by introducing a commitment scheme in the CRS model that can achieve sub-equivocality and binding. By considering the negative result, this is the best case one can achieve if they want to retain equivocality when commitment key is subverted. In the second scenario, we consider if we can construct commitment schemes that will satisfy sub-binding? In subsection 5.2, we show the best we can achieve while retaining sub-binding is sub-hiding; by introducing some already known schemes that simultaneously achieving sub-binding and sub-hiding. The first positive result 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? This is answered positively in subsection 5.3.

5.1 Sub-equivocality and Binding

By considering the definition of sub-equivocality (given in Def. 6), to achieve sub-equivocality in a commitment scheme, there must be algorithms KGen^* , Com^* and Equiv , where KGen^* simulates *malicious* setup phase, and Com^* and Equiv output a fake commitment and the associated valid opening, consequently. In this case, the algorithms Com^* and Equiv cannot get honestly generated trapdoors of ck , and 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 allows extracting trapdoors of ck from a malicious key generator in a non-black-box way. Once we extracted the tk , it will be provided to algorithms Com^* and Equiv to generate a pair of fake but acceptable commitment and opening. Moreover, in the case of a malicious key generator, there is an issue with the setup information gk , e.g. groups description. They cannot be generated as before, as they can be subverted. Similar to subversion-resistant NIZK arguments [BFS16], this issue is addressed by considering the 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 . To guarantee binding, the minimal requirement is that an adversary cannot obtain the tk of ck from a honestly generated ck .

Theorem 2 (A Sub-equivocal and Binding Commitment). *Let BGgen be a bilinear group generator. Then the commitment scheme Π_{com} described in Fig. 2 which is a variation of knowledge commitment scheme introduced in [Gro10, Lip12], 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. Our proposed variation has the same ck as the original scheme, so the proof of (knowledge) binding can be shown straightforwardly from the original scheme, which is done in [Lip12] under the Γ -PSDL assumption in the group \mathbb{G}_t for $t \in \{1, 2\}$.

To prove sub-equivocality, it was shown that the original scheme is equivocal under a trusted setup, namely the setup phase is simulatable, and the algorithms Com^* and Equiv that can generate a fake commitment and valid opening are shown in Fig. 2. In the original scheme, the algorithms Com^* and Equiv get the honestly generated trapdoor tk , but in our case the tk is not trustable anymore.

Let \mathcal{A} be a malicious key generator. To prove sub-equivocality, we first need to show that the setup phase is simulatable. Namely, there exists KGen^* which can produce the full view of key generation by \mathcal{A} . Second, we need to describe two algorithms Com^* and Equiv which given the extracted trapdoor they can produce a fake commitment and a valid opening which are indistinguishable from the real ones. To address the first issue, we construct a non-black-box

Setup, $\mathbf{gk} \leftarrow \text{BGgen}(1^\lambda)$: Given 1^λ , return $\mathbf{gk} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2)$, where p (a large prime) is the order of cyclic Abelian groups $\mathbb{G}_1, \mathbb{G}_2$, and \mathbb{G}_T ; $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is an efficient non-degenerate bilinear pairing.

Key Generation, $\mathbf{ck} \leftarrow \text{KGen}(\mathbf{gk})$: Let Γ be an (n, λ) -nice tuple for some $n = \text{poly}(\lambda)$ with $\gamma_i = i$ in the original version, for $i \in [0..n]$. Sample $\hat{a}, x \leftarrow \mathbb{Z}_p$. Let $t \in \{1, 2\}$. Return the key $\mathbf{ck} = (\mathbf{ck}_1, \mathbf{ck}_2)$ where $\mathbf{ck}_t \leftarrow \{[x^i]_t, [\hat{a}x^i]_t\}$ for $i \in [0..n]$ and the corresponding trapdoor \mathbf{tk} as $\mathbf{tk} = x$.

Commitment Key Verification, $0/1 \leftarrow \text{CKVer}(\mathbf{gk}, \mathbf{ck})$: Given \mathbf{gk} and the commitment key \mathbf{ck} , first parse $\mathbf{ck} := (\{[x^i]_1, [\hat{a}x^i]_1\}, \{[x^i]_2, [\hat{a}x^i]_2\})$ for $i \in [0..n]$ and then do the following verification on elements of the \mathbf{ck} ,

- Check whether $[\hat{a}]_1 \bullet [1]_2 = [1]_1 \bullet [\hat{a}]_2$
- For $i \in [1..n]$ check:
 1. $[x^i]_1 \bullet [1]_2 = [1]_1 \bullet [x^i]_2$
 2. $[\hat{a}x^i]_1 \bullet [1]_2 = [1]_1 \bullet [\hat{a}x^i]_2$
 3. $[\hat{a}]_1 \bullet [x^i]_2 = [1]_1 \bullet [\hat{a}x^i]_2$
 4. $[\hat{a}x]_1 \bullet [x^{i-1}]_2 = [1]_1 \bullet [\hat{a}x^i]_2$

and return 1 if all checks passed successfully; otherwise return 0.

Committing, $(c, \text{op}) \leftarrow \text{Com}(\mathbf{ck}, \mathbf{m}; r)$: Given $(\mathbf{ck}, \mathbf{m})$ for $\text{CKVer}(\mathbf{gk}, \mathbf{ck}) = 1$, to commit to $\mathbf{m} = (m_1, m_2, \dots, m_n) \in \mathbb{Z}_p^n$ sample a random $r \leftarrow \mathbb{Z}_p$, and return $(c, \text{op} := r)$ that are defined as follows,

$$c := (c_t^1, c_t^2) = (r [1]_t + \sum_{i=1}^n m_i [x^i]_t, r [\hat{a}]_t + \sum_{i=1}^n m_i [\hat{a}x^i]_t)$$

Opening Verification, $0/1 \leftarrow \text{Ver}(\mathbf{ck}, c, \mathbf{m}, \text{op})$: Given c, \mathbf{m} and $\text{op} = r$, recompute c as original one and check if it is equal to given c and return 0/1.

Simulation of Key Generation, $(\mathbf{ck}, \mathbf{tk}) \leftarrow \text{KGen}^*(\mathbf{gk})$: Use the simulation algorithm $\text{Sim}_{\mathcal{A}}$ in Fig. 4 and generates a key pair $(\mathbf{ck}, \mathbf{tk} := (x, \hat{a}))$.

Trapdoor Committing, $(c, \mathbf{ek}) \leftarrow \text{Com}^*(\mathbf{ck}, \mathbf{tk})$: Given the a key pair $(\mathbf{ck}, \mathbf{tk})$, output an equivocal commitment $c = [r]_t$ where $r \leftarrow \mathbb{Z}_p^2$ and an equivocation key $\mathbf{ek} = (\mathbf{tk}, r)$.

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

Fig. 2: A variation of the commitment scheme of Groth [Gro10] defined by Lipmaa [Lip12] that achieves sub-equivocality and binding. We note that in this setting, $\mathbf{gk} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2)$ is part of the scheme specification, and in practice each party can run deterministic algorithm BGgen and re-obtain \mathbf{gk} .

extraction algorithm $\text{Ext}_{\mathcal{A}}$ that can extract the trapdoor \mathbf{tk} from a malicious key generator \mathcal{A} and simulate the setup phase. 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 extract a . In the rest of the proof, we construct an efficient extractor under BDH-KE assumption which allows us to extract the trapdoor \mathbf{td} from \mathcal{A} .

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

Given source code and random coins of the malicious key generator \mathcal{A} , and some auxiliary information $\xi_{\mathbf{R}}$ it extracts $(x, \hat{a}) \leftarrow \text{Ext}_{\mathcal{A}}(\text{gk}, \text{ck}, \xi_{\mathbf{R}})$ and set $\text{tk} := (x, \hat{a})$; Finally, **Return** tk.

Fig. 3: A BDH-KE assumption based extraction algorithm $\text{Ext}_{\mathcal{A}}$ for the sub-
equivocal commitment scheme described in Fig. 2

Simulator $\text{Sim}_{\mathcal{A}}(\text{gk})$:

$\text{gk} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2, \xi_{\mathbf{R}}) \leftarrow \text{BGgen}(1^\lambda)$; $\text{ck} \leftarrow \mathcal{A}(\text{gk})$; # as in Fig. 2

By executing $\text{CKVer}(\text{gk}, \text{ck})$,

Check whether $[\hat{a}]_1 \bullet [1]_2 = [1]_1 \bullet [\hat{a}]_2$

For $i \in [1..n]$ check:

1. $[x^i]_1 \bullet [1]_2 = [1]_1 \bullet [x^i]_2$
2. $[\hat{a}x^i]_1 \bullet [1]_2 = [1]_1 \bullet [\hat{a}x^i]_2$
3. $[\hat{a}]_1 \bullet [x^i]_2 = [1]_1 \bullet [\hat{a}x^i]_2$
4. $[\hat{a}x]_1 \bullet [x^{i-1}]_2 = [1]_1 \bullet [\hat{a}x^i]_2$

if the checks pass, $\text{tk} := (x, \hat{a}) \leftarrow \text{Ext}_{\mathcal{A}}(\text{gk}, \text{ck}, \xi_{\mathbf{R}})$ # as in Fig. 3

Otherwise $\text{tk} \leftarrow \perp$

Return (ck, tk)

Fig. 4: Simulation of the setup phase in the knowledge commitment scheme
described in Fig. 2.

Let \mathcal{A} outputs $\text{ck} = (\text{ck}_1, \text{ck}_2)$, where $\text{ck}_t \leftarrow \{[x^i]_t, [\hat{a}x^i]_t\}$ for $i \in [0..n]$ and $t \in \{1, 2\}$, as described in Fig. 2. By considering BDH-KE assumption, and verifications done in CKVer , one can observe that if a malicious key generator \mathcal{A} manages to output a *well-formed* ck , it must know x and \hat{a} . By well-formed ck , we mean it must pass all checks in CKVer ⁴. So it implies that there exists a polynomial time extractor $\text{Ext}_{\mathcal{A}}$ that if all the verifications in CKVer pass for some \hat{a} and x , then the $\text{Ext}_{\mathcal{A}}$ can extract x and \hat{a} ; as $\text{Adv}_{\text{BGgen}, \mathcal{A}, \text{Ext}_{\mathcal{A}}}^{\text{BDH-KE}}$ is negligible. A high-level description of the extraction procedure is shown in Fig. 3. After using the extractor $\text{Ext}_{\mathcal{A}}$, one can simulate a malicious key generation using algorithm $\text{Sim}_{\mathcal{A}}$ described in Fig. 4.

Finally, using the extracted trapdoor tk , one can consider the rest of proof as the proof of equivocality given in the original scheme [Lip12], by showing that given the (extracted) trapdoors one can use two algorithms Com^* and Equiv (described in Fig. 2) and generate a fake commitment and the corresponding valid opening that will be successfully verified by Ver . \square

⁴ Note that verifications such as $[\hat{a}]_1 \bullet [1]_2 = [1]_1 \bullet [\hat{a}]_2$ inside CKVer comes from the definition of the BDH-KE. 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.

Remark 1. In practice, executing the 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.

Below, we proposed a batched version of the proposed CKVer. For the sub-equivocal commitment scheme given in Fig. 2, to execute CKVer, one needs to compute $6n+2$ parings (note that right hand of some verifications are the same). But with with batched CKVer algorithm in Fig. 5, one can verify ck with only 4 parings and $8n$ exponentiations, that for large values of n , this takes considerably less time. This can also be optimized by using the same randomness for the different equations, that would allow to save $2n$ exponentiations.

Batched Commitment Key Verification, $0/1 \leftarrow \text{CKVer}(\text{gk}, \text{ck})$: Batched CKVer is an efficient 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 $\text{gk} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2) \leftarrow \text{BGGen}(1^\lambda)$;
- Samples three vector of randomnesses with length n as $\mathbf{r}, \mathbf{s}, \mathbf{t}, \mathbf{q} \leftarrow_{\$} \{1, \dots, 2^\lambda\}$;
- If
$$\begin{pmatrix} [\hat{a}]_1 + \sum_{i=1}^n r_i [x^i]_1 + \sum_{i=1}^n s_i [\hat{a}x^i]_1 \\ \sum_{i=1}^n t_i [x^i]_2 + [\hat{a}x]_1 \end{pmatrix} \bullet [1]_2 + [\hat{a}]_1 \bullet \begin{pmatrix} [\hat{a}x^i]_1 \\ \sum_{i=1}^n q_i [x^{i-1}]_2 \end{pmatrix} = [1]_1 \bullet \begin{pmatrix} [\hat{a}]_1 \\ \sum_{i=1}^n q_i [\hat{a}x^i]_2 \end{pmatrix}$$
 then **return** 1 (the ck is well-formed);
- Else, **return** 0 (the ck is not well-formed);

Fig. 5: Batched CKVer algorithm for sub-equivocal commitment scheme in Fig. 2

5.2 Sub-binding and Sub-hiding

Next, we discuss the second positive result. Let $\Pi_{\text{com}}^{2\text{-party}} = (\text{KGen}, \text{Com}, \text{Ver})$ be a commitment that does not require a particular setup and the output of KGen can be ignored. This includes all classical commitments that guarantee hiding and binding and do not require a setup. In other words, all the commitments that only need to choose some public parameters that can be agreed between both parties, e.g., agreeing on the order and generator of the underlying group or a particular secure and collision resistant hash function family.

We show that such hiding and binding commitment schemes also guarantee sub-hiding and sub-binding. Intuitively, one can see that in such case (e.g. $\text{ck} = \{\}$) there is no risk of subverting ck.

Lemma 2. *Let $\Pi_{\text{com}}^{2\text{-party}} = (\text{KGen}, \text{Com}, \text{Ver})$ be a commitment scheme that does not require a particular setup phase. If $\Pi_{\text{com}}^{2\text{-party}}$ satisfies binding and hiding, it also guarantees sub-binding and sub-hiding.*

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

$$\Pr \left[\begin{array}{l} \text{gk} \leftarrow \text{Setup}(1^\lambda), (\text{ck}, c, (m_0, \text{op}_0), (m_1, \text{op}_1)) \leftarrow \mathcal{A}(\text{gk}) : \\ \text{CKVer}(\text{gk}, \text{ck}) = 1 \wedge (m_0 \neq m_1) \\ \wedge (\text{Ver}(\text{ck}, c, m_0, \text{op}_0) = 1) \wedge (\text{Ver}(\text{ck}, c, m_1, \text{op}_1) = 1) \end{array} \right] = 1 - \text{negl}(\lambda) .$$

By considering the fact that in a $\Pi_{\text{com}}^{2\text{-party}}$ commitment, so its commitment key can be generated by either \mathcal{A} or the honest KGen. So in above game, one can substitute malicious key generator \mathcal{A} with an honest KGen, meaning that

$$\Pr \left[\begin{array}{l} \text{gk} \leftarrow \text{Setup}(1^\lambda), \text{ck} \leftarrow \text{KGen}(\text{gk}), \\ (c, (m_0, \text{op}_0), (m_1, \text{op}_1)) \leftarrow \mathcal{A}(\text{gk}, \text{ck}) : \text{CKVer}(\text{gk}, \text{ck}) = 1 \wedge \\ (m_0 \neq m_1) \wedge (\text{Ver}(\text{ck}, c, m_0, \text{op}_0) = 1) \wedge (\text{Ver}(\text{ck}, c, m_1, \text{op}_1) = 1) \end{array} \right] = 1 - \text{negl}(\lambda) .$$

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

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

$$\left| 2 \Pr \left[\begin{array}{l} \text{gk} \leftarrow \text{Setup}(1^\lambda), (\text{ck}, (m_0, m_1)) \leftarrow \mathcal{A}(\text{gk}), \\ b \leftarrow_{\$} \{0, 1\}, \text{CKVer}(\text{gk}, \text{ck}) = 1, r_b \leftarrow_{\$} \mathcal{R}, \\ (c_b, \text{op}_b) \leftarrow \text{Com}(\text{ck}, m_b; r_b), b' \leftarrow \mathcal{A}(c_b) : b' = b \end{array} \right] - 1 \right| = 1 - \text{negl}(\lambda) .$$

Again, by considering the property of a $\Pi_{\text{com}}^{2\text{-party}}$ commitment, one can substitute malicious key generator \mathcal{A} in the setup phase with an honest KGen, which results,

$$\left| 2 \Pr \left[\begin{array}{l} \text{gk} \leftarrow \text{Setup}(1^\lambda), \text{ck} \leftarrow \text{KGen}(\text{gk}), (m_0, m_1) \leftarrow \mathcal{A}(\text{gk}), \\ b \leftarrow_{\$} \{0, 1\}, \text{CKVer}(\text{gk}, \text{ck}) = 1, r_b \leftarrow_{\$} \mathcal{R}, \\ (c_b, \text{op}_b) \leftarrow \text{Com}(\text{ck}, m_b; r_b), b' \leftarrow \mathcal{A}(c_b) : b' = b \end{array} \right] - 1 \right| = 1 - \text{negl}(\lambda) .$$

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

Theorem 3 (Sub-hiding and Sub-binding Commitment Schemes). *Under some standard assumptions, there exist commitment schemes that achieve sub-hiding and sub-binding.*

Proof. Basically all classic commitment schemes that do not require a particular setup phase and guarantee hiding and binding are a $\Pi_{\text{com}}^{2\text{-party}}$ commitment scheme. For instance, a commitment scheme built using a family of collision-resistant hash functions⁵. As a result, by considering the result of Lemma 2, all of them can also guarantee sub-hiding and sub-binding. \square

⁵ A sample construction is available on <https://cs.nyu.edu/courses/fall108/G22.3210-001/lect/lecture14.pdf>

5.3 Binding, Equivocality and Sub-hiding

Finally, we consider the last positive result in Tab. 1 which states that we can have a commitment to achieving hiding, equivocality, binding, and sub-hiding at the same time. In this result, we show that one can still achieve sub-hiding under standard assumptions by requiring that there exist hiding, binding, and equivocal commitment schemes.

Pedersen Commitment Scheme Achieves Sub-hiding. The Pedersen commitment scheme [Ped92] can guarantee sub-hiding property with minimal checking. The committer only needs to run the CKVer algorithm to verify ck before using the key for committing, and the check for this scheme is quite simple. Basically a committer needs to check whether both $g \neq 0$ and $h \neq 0$ before using $ck = (g, h)$.

Theorem 4 (Subversion-Resistant Pedersen Commitment). *The Pedersen commitment scheme with checking $g \neq 0$ and $h \neq 0$, satisfies hiding, equivocal, binding and sub-hiding under the discrete logarithm assumption in \mathbb{G} .*

Proof. For the sub-hiding property, once CKVer(gk, 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 the original scheme, given trapdoor tk of the commitment key ck, one can generate a fake commitment and the corresponding valid opening. \square

Acknowledgment. This work was supported in part by the Estonian Research Council grant PRG49, by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001120C0085, and by Cyber Security Research Flanders with reference number VR20192203. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the ERC, DARPA, the US Government or Cyber Security Research Flanders. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

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.

- ABLZ17. Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, and Michal Zajac. A subversion-resistant SNARK. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 3–33. Springer, Heidelberg, December 2017.
- 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>.
- Bag19. Karim Baghery. Subversion-resistant simulation (knowledge) sound NIZKs. In Martin Albrecht, editor, *17th IMA International Conference on Cryptography and Coding*, volume 11929 of *LNCS*, pages 42–63. Springer, Heidelberg, December 2019.
- 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.
- 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.
- 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.
- 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.
- Fuc18. Georg Fuchsbauer. Subversion-zero-knowledge SNARKs. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 315–347. Springer, Heidelberg, March 2018.
- 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.
- GKM⁺18. Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Heidelberg, August 2018.
- 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 transmission 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 against-vote 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.
- 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 Perloth, 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. Clipping: 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, *EURO-CRYPT'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.
- 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.

A Appendix

A.1 Standard Notions for Equivocal Commitment Schemes

An equivocal commitment scheme $\Pi_{com} = (\text{KGen}, \text{Com}, \text{Ver}, \text{KGen}^*, \text{Com}^*, \text{Equiv})$ satisfies hiding, binding and equivocality that are defined below.

Definition 7 (Hiding). *A commitment scheme Π_{com} is computationally hiding if for any PPT adversary \mathcal{A} ,*

$$\left| 2 \Pr \left[\begin{array}{l} \text{gk} \leftarrow \text{Setup}(1^\lambda), \text{ck} \leftarrow \text{KGen}(\text{gk}), \\ (m_0, m_1) \leftarrow \mathcal{A}(\text{ck}), b \leftarrow_s \{0, 1\}, r_b \leftarrow_s \mathcal{R}, \\ (c_b, \text{op}_b) \leftarrow \text{Com}(\text{ck}, m_b; r_b), b' \leftarrow \mathcal{A}(\text{ck}, c_b) : b' = b \end{array} \right] - 1 \right| = \text{negl}(\lambda) .$$

The commitment is *perfectly* hiding if the above probability is equal to 0.

Definition 8 (Binding). *A commitment scheme Π_{com} is computationally binding if for any PPT adversary \mathcal{A} ,*

$$\Pr \left[\begin{array}{l} \text{gk} \leftarrow \text{Setup}(1^\lambda), \text{ck} \leftarrow \text{KGen}(\text{gk}), \\ (c, (m_0, \text{op}_0), (m_1, \text{op}_1)) \leftarrow \mathcal{A}(\text{ck}) : (m_0 \neq m_1) \wedge \\ (\text{Ver}(\text{ck}, c, m_0, \text{op}_0) = 1) \wedge (\text{Ver}(\text{ck}, c, m_1, \text{op}_1) = 1) \end{array} \right] = \text{negl}(\lambda) .$$

The commitment is *perfectly* binding if the above probability is equal to 0.

Definition 9 (Equivocality). A commitment scheme $\Pi_{com} = (\text{KGen}, \text{Com}, \text{Ver}, \text{KGen}^*, \text{Com}^*, \text{Equiv})$ is *equivocal* if there exist PPT algorithms Com^* and Equiv that given the trapdoor of the commitment key, can come up with a fake commitment and a valid opening s.t. they would be indistinguishable from the real ones. More formally, for $\text{gk} \leftarrow \text{Setup}(1^\lambda)$, for any PPT adversary \mathcal{A} ,

$$\left| \Pr \left[\begin{array}{l} \text{ck} \leftarrow \text{KGen}(\text{gk}), \\ m \leftarrow \mathcal{A}(\text{ck}), r \leftarrow_s \mathcal{R}, \\ (c, \text{op}) \leftarrow \text{Com}(\text{ck}, m; r) : \\ \mathcal{A}(\text{ck}, c, \text{op}) = 1 \end{array} \right] - \Pr \left[\begin{array}{l} (c, \text{tk}) \leftarrow \text{KGen}^*(\text{gk}), \\ m \leftarrow \mathcal{A}(\text{ck}), \\ (c, \text{ek}) \leftarrow \text{Com}^*(\text{ck}, \text{tk}), \\ \text{op} \leftarrow \text{Equiv}(\text{ek}, c, m) : \\ \mathcal{A}(\text{ck}, c, \text{op}) = 1 \end{array} \right] \right| \leq \text{negl}(\lambda)$$

where \mathcal{A} outputs $m \in \mathcal{M}$.

Equivocality implies hiding, as a commitment is indistinguishable from an equivocal commitment that can be opened to any message [Gro09].

A.2 Full Proof of Theorem 1

Proof. In the rest, we say the verification of a commitment scheme is *trivial* if the verification algorithm can decide about the validity of commitment c and opening information op on its own. Let Π_{com} be a NI commitment scheme in the CRS model 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 $(c, (m, \text{op}))$, where c is a commitment, m is a committed message, and op is an opening information. Here c is a challenge commitment that may or may not be a **valid** commitment for (m, op) , and (m, op) should be a **valid** opening (it means Ver 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 Ver and decision procedure DP ,

Experiment $\text{DEC}_{\text{COG}, \text{Ver}, \text{DP}}(\text{gk})$:

$(c, (m, \text{op})) \leftarrow_s \text{COG}(\text{gk}); d_1 \leftarrow \text{Ver}(\text{ck}, c, (m, \text{op}));$
 If $(c$ is **valid** and $d_1 = \text{false})$ then return **false**;
 $d_0 \leftarrow_s \text{DP}(\text{gk}, c);$
Return $d_0 \neq d_1$;

Let $\text{Adv}_{\text{COG}, \text{Ver}, \text{DP}}^{\text{DEC}}(\text{gk}) = \Pr[\text{DEC}_{\text{COG}, \text{Ver}, \text{DP}}(\text{gk})]$. Now, we say that algorithm DP decides Ver if for every polynomial time COG the function $\text{Adv}_{\text{COG}, \text{Ver}, \text{DP}}^{\text{DEC}}(\text{gk})$ is $\text{negl}(\lambda)$. We say that verification by Ver is *trivial* if there is a polynomial time algorithm DP that decides Ver . 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 the 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**.

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

Algorithm DP(gk, c)

$(\text{ck}, \text{tk}) \leftarrow_{\$} \text{KGen}^*(\text{gk}); (c, \text{ek}) \leftarrow \text{Com}^*(\text{ck}, \text{tk}); m \leftarrow \mathcal{M}, \text{op} \leftarrow \text{Equiv}(\text{ek}, c, m, \text{ek});$

Returns $\text{Ver}(\text{ck}, c, m, \text{op})$

Thus, to decide if c is a **valid** commitment, algorithm DP runs the simulator of key generations 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 $(c, (m, \text{op}))$ are valid commitment and opening. Let COG be any polynomial time commitment-opening generator. We will show that $\text{Adv}_{\text{COG, Ver, DP}}^{\text{DEC}}(\text{gk})$ is negligible. This shows verification Ver is trivial. To show $\text{Adv}_{\text{COG, Ver, DP}}^{\text{DEC}}(\text{gk})$ is negligible, below we will define polynomial-time adversaries A and B such that

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

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 $\text{Adv}_{\text{COG, Ver, A}}^{\text{equivocality}}(\text{gk})$ and $\text{Adv}_{\text{COG, Ver, B}}^{\text{sub-binding}}(\text{gk})$ are negligible. Thus, $\text{Adv}_{\text{COG, Ver, DP}}^{\text{dec}}(\text{gk})$ in left side of above inequality is negligible, as desired. Consider experiments $\text{Exp}_0, \text{Exp}_1$ 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.

Exp₀:

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

Exp₁:

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

Exp₂:

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

Experiment Exp₂ 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,

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

The first condition (left one), is equivalent to the case that experiment Exp₀ returns true, and the second condition (right one) is equivalent to $\underline{(d_1 = \text{true}) \wedge (d_0 = \text{false})}$ (as valid commitments always are accepted), which is equivalent to the case when experiment Exp₁ returns true.

Furthermore the conditions are mutually exclusive and cannot both occur at the same time. Therefore, we have

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

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

$$\text{Adv}_{\text{COG}, \text{Ver}, \text{DP}}^{\text{dec}}(\text{gk}) = \Pr[\text{Exp}_0] + (\Pr[\text{Exp}_1] - \Pr[\text{Exp}_2]) \quad (2)$$

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

Adversary $A^{\text{equivocality}}(\text{gk}, \text{ck})$:

$(c, (m_1, \text{op}_1)) \leftarrow_{\S} \text{COG}(\text{gk});$
 $d_1 \leftarrow \text{Ver}(\text{ck}, c, (m_1, \text{op}_1));$
 $m_2 \leftarrow \mathcal{M}; (c, \text{op}_2) \leftarrow \text{Com}(\text{ck}, m_2; \text{op}_2);$
 $d_0 \leftarrow \text{Ver}(\text{ck}, c, m_2, \text{op}_2);$
 If $(d_1 = \text{true}) \wedge (d_0 = \text{false})$ then $b' = 0$;
 Else $b' = 1$;
 Return b'

Adversary $B^{\text{sub-binding}}(\text{gk})$:

$(c, (m_1, \text{op}_1)) \leftarrow_{\S} \text{COG}(\text{gk});$
 $(\text{ck}, \text{tk}) \leftarrow_{\S} \text{KGen}^*(\text{gk});$
 $(c, \text{ek}) \leftarrow \text{Com}^*(\text{ck}, \text{tk});$
 $m_2 \leftarrow \mathcal{M}, \text{op}_2 \leftarrow \text{Equiv}(\text{ek}, c, m_2, \text{ek});$
 Return $(\text{ck}, c, (m_1, \text{op}_1), (m_2, \text{op}_2))$

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

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

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

Next, by substituting last two inequalities in equation (2), we get to

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

This results the theorem. □