# Tiramisu: Black-Box Simulation Extractable NIZKs in the Updatable CRS Model

Karim Baghery[1,2] and Mahdi Sedaghat[1]

[1] imec-COSIC, KU Leuven, Leuven, Belgium
[2] University of Tartu, Tartu, Estonia
karim.baghery@kuleuven.be, ssedagha@esat.kuleuven.be

**Abstract.** In CRYPTO'18, Groth et al. introduced the *updatable* CRS model that allows bypassing the trust in the setup of NIZK arguments. Zk-SNARKs are the well-known family of NIZK arguments that are ubiquitously deployed in practice. In applications that achieve *universal composability*, e.g. Hawk [S&P'16], Gyges [CCS'16], Ouroboros Crypsinous [S&P'19], the underlying SNARK is lifted by the COCO framework [Kosba et al.,2015] to achieve Black-Box Simulation Extractability (BB-SE). The COCO framework is designed in the standard CRS model, consequently, the BB-SE NIZK arguments built with it need a trusted setup phase. In a promising research direction, recently subversion-resistant and updatable SNARKs are proposed that can eliminate/bypass the needed trust in schemes. However, none of the available subversion-resistant/updatable schemes can achieve BB-SE, as Bellare et al.'s result from ASIACRYPT'16 shows that achieving simultaneously Sub-ZK (ZK without trusting a third party) and BB extractability is impossible.

In this paper, we propose Tiramisu [3], as a construction to build BB-SE NIZK arguments in the *updatable* CRS model. Similar to the COCO, Tiramisu is suitable for modular use in larger cryptographic systems and allows building BB-SE NIZK arguments, but with *updatable* parameters. In the cost of one time CRS update, Tiramisu gets arround the mentioned impossibility result by Bellare et al. Namely, by one time updating the CRS, all the parties eliminate the trust on a third-party and the protocol satisfies ZK and BB-SE in the *updatable* CRS model. Meanwhile, we define a variation of public-key cryptosystems with updatable keys, suitable for the updatable CRS model, and present an efficient construction based on the El-Gamal cryptosystem which can be of independent interest. We instantiate Tiramisu and present efficient BB-SE zk-SNARKs with updatable parameters that can be used in protocols like Hawk, Gyges, Ouroboros Crypsinous while allowing the end-users to update the parameters and eliminate the needed trust.

**Keywords:** zk-SNARKs, updatable CRS, Black-Box Simulation Extractability, C∅C∅ framework, UC-Security

[3] In Italian, Tiramisu literally means "pull me up, lift me up", or more literally "pull it up". This work is done during a self-quarantine period of authors to reduce the spread of COVID-19.

# 1 Introduction

Zero-Knowledge (ZK) [GMR89] proof systems, particularly Non-Interactive Zero-Knowledge (NIZK) arguments [BFM88] are one of the elegant tools in the modern cryptography that due to their impressive advantages and practical efficiency, they are ubiquitously deployed in practical applications [BCG+14,KMS+16,JKS16,KKKZ19]. A NIZK proof system allows a party $\mathsf{P}$ (called prover) to non-interactively prove the truth of a statement to another party $\mathsf{V}$ (called verifier) without leaking any information about his/her secret inputs. For instance, they allow a prover $\mathsf{P}$ to convince a verifier $\mathsf{V}$ that for a (public) statement $\mathsf{x}$, he/she knows a (secret) witness $\mathsf{w}$ that satisfies a relation $\mathbf{R}$, $(\mathsf{x}, \mathsf{w}) \in \mathbf{R}$, without leaking any information about $\mathsf{w}$.

Typically, a NIZK argument is expected to satisfy, (i) *Completeness*, which implies that an honest prover always convinces an honest verifier (ii) *Soundness*, which ensures that an adversarial prover cannot convince an honest verifier except with negligible probability. (iii) *Zero-Knowledge* (ZK), which guarantees that an honestly generated proof does not reveal any information about the (secret) witness $\mathsf{w}$. ZK is the desired notion for the prover, and to prove ZK in an argument, one needs to construct a new algorithm called *simulator* $\mathsf{Sim}$ that without getting access to the witness $\mathsf{w}$, but some secret information (related to public parameters) or some extra power, can generate *simulated* proofs that are indistinguishable from the *real* ones. On the other hand, soundness is the desired notion for verifier as it does not allow the prover to cheat. However, in most of the practical cases, it is shown that bare *soundness* is not sufficient and it needs either to be amplified [KMS+16] or the protocol needs to be supported by other cryptographic primitives [BCG+14]. To deal with such concerns, different constructions are proposed that either satisfy one of the following notions, one of which is an amplified variation of soundness. (iv) *Simulation Soundness*, (SS), which ensures that an adversarial prover cannot convince an honest verifier, even if he has seen polynomially time simulated proofs (generated by $\mathsf{Sim}$), except with negligible probability. (v) *Knowledge Soundness* (KS), which guarantees that an adversarial prover cannot convince an honest verifier, unless he *knows* a witness $\mathsf{w}$ for statement $\mathsf{x}$ such that $(\mathsf{x}, \mathsf{w}) \in \mathbf{R}$. (vi) *Simulation Extractability* (SE) (a.k.a. *Simulation Knowledge Soundness*), which guarantees that an adversarial prover cannot convince an honest verifier, even if he has seen polynomially time simulated proofs, unless he *knows* a witness $\mathsf{w}$ for statement $\mathsf{x}$.

The term *knowledge* in notions KS (in item v) and SE (in item vi) means that a successful (adversarial) prover should *know* a witness. In constructions, the concept of *knowing* is formalized by showing that there exists an extraction algorithm $\mathsf{Ext}$, which can extract the witness $\mathsf{w}$ (from either the prover or the proof) in either *non-Block-Box* (nBB) or *Black-Box* (BB) manner. Typically, nBB extraction can result in more efficient constructions, as it allows $\mathsf{Ext}_{\mathcal{A}}$ to get access to the source-code and random coins of the adversary $\mathcal{A}$. Although the constructions that obtain BB extractability are less efficient, they provide stronger security guarantees, as there exists a universal extractor $\mathsf{Ext}$ for *any* $\mathcal{A}$. The term *simulation* in notions SS (in item iv) and SE (in item vi) guarantees that the proofs are non-malleable and an adversary cannot change an old (simulated) proof to a new one such that the verifier will accept it. The notion SE provides the strongest security and also implies non-malleability of proofs as defined in [DDO$^+$01]. Moreover, in [Gro06], it is shown that SE is a sufficient requirement for a NIZK argument to be deployed in a Universally Composability (UC) protocol [Can01].

**NIZK Arguments and zk-SNARKs in the CRS Model.** In the Common Reference String (CRS) model [BFM88], the construction of NIZK arguments requires a trusted setup phase that outputs some public parameters, known as CRS, and shares with the parties. During the last two decades, there has been considerable progress in constructing CRS-based NIZK arguments. Based on the underlying assumptions, they are constructed either using falsifiable or non-falsifiable assumptions [Nao03]. Although the early constructions were mostly based on falsifiable assumptions, they were inefficient and impractical, e.g. Groth-Sahai proofs [GS08].

Following this fact, at the beginning of the last decade, a line of research initiated that focused on constructing NIZK arguments with shorter proofs and more efficient verification. This direction, finally led to a very efficient family of NIZK arguments, called zero-knowledge Succinct Non-interactive ARgument of Knowledge (zk-SNARK) [Gro10,Lip12,PHGR13,BCTV13,Gro16,GM17,BG18], [AB19,Lip19,BPR20]. Zk-SNARKs have *succinct* proofs and efficient verification [4]. Their efficiency mainly comes from the fact that they all are constructed based on non-falsifiable assumptions (e.g. knowledge assumptions [Dam91]) that allow *succinct* proofs and nBB extractability. Meanwhile, in 2011, Gentry and Wichs's impossibility result [GW11] confirmed that NIZK arguments with *succinct* proofs cannot be built based on falsifiable assumptions. Beside *succinct* proofs (e.g. 3 group elements), all initial zk-SNARKs were designed to achieve completeness, ZK and KS (in item v) with nBB extraction [Gro10,Lip12,PHGR13,BCTV13,Gro16]. KS does not guarantee the non-malleability proofs, so in practice to prevent man-in-the-middle attacks, users needed to support the protocol with extra primitives. For instance, the cryptocurrency Zcash [BCG$^+$14] makes extra efforts with hash functions to guarantee the non-malleability of transactions and proofs. Following this concern, in 2017, Groth and Maller [GM17] presented a zk-SNARK that can achieve SE (in item vi) with nBB extractability, consequently guarantees non-malleability of proofs. Recent works in this direction have led to more efficient schemes with the same security guarantees [BG18,AB19,Lip19,BKSV20,BPR20].

**Mitigating the trust in the setup phase of zk-SNARKs.** Due to constructing zk-SNARKs in the CRS model, both the prover and verifier are required to trust the CRS generator. As a common approach to mitigate the trust, in 2015, Ben Sasson et al. [BCG$^+$15] proposed an efficient MPC protocol that can be used to sample CRS of the majority of pairing-based zk-SNARKs. While using

[4] In 2018, zk-SNARKs were listed as one of "10 Breakthrough Technologies of 2018", published by MIT technology review. Available on `https://www.technologyreview.com/lists/technologies/2018/`.

such MPC protocols for CRS generation, both prover and verifier need to trust *1 out of n* parties, instead of trusting a single party entirely, where $n$ denotes the number of parties participated in the MPC protocol [BGM17,BGG19,ABL$^+$19].

In a different research direction, in 2016, Bellare et al. [BFS16] studied the security of CRS-based NIZK arguments in the face of subverted parameters and presented some negative and positive results. They first defined (vii) *Subversion-Soundness*, (Sub-SND), which ensures that the protocol guarantees soundness (in item ii) even if $\mathcal{A}$ has generated the CRS, and (viii) *Subversion-ZK*, (Sub-ZK), which ensures that the protocol guarantees ZK (in item iii) even if $\mathcal{A}$ has generated the CRS. Then, they showed some of the new definitions are not compatible and we cannot construct NIZK arguments that would achieve Sub-SND together with (standard) ZK, and also constructions that will satisfy Sub-ZK together with BB extractability (either KS or SE with BB extraction). Considering those results, two works [ABLZ17,Fuc18] showed that most of pairing-based zk-SNARKs [PHGR13,BCTV13,Gro16], can be lifted to achieve Sub-ZK (in item viii) and KS (in item v) with *nBB* extraction (nBB-KS). Then, Baghery [Bag19b] showed that using the folklore OR technique [BG90] any zk-SNARK that satisfies Sub-ZK and nBB-KS, can be lifted to achieve Sub-ZK and SE (in item vi) with nBB extraction (nBB-SE). The latest result showed that one can construct NIZK arguments that the prover does not need to trust the CRS generator to achieve ZK, and the construction achieves nBB-SE.

Meanwhile, as an extension to the MPC approach and subversion security, in 2018 Groth et al. [GKM$^+$18] introduced a new variation of the CRS model, called *updatable* CRS model. In this model, the CRS is updatable and both prover and verifier can update the CRS and bypass the needed trust in a third party. Groth et al. first defined, (ix) *Updatable KS*, (U-KS), which ensures that the protocol guarantees KS (in item v) as long as the initial CRS generation or one of CRS updates is done by an honest party, and (x) *Updatable ZK*, (U-ZK), which ensures that the protocol guarantees ZK as long as the initial CRS generation or one of CRS updates is done by an honest party [5] . Then, they presented a zk-SNARK that can achieve Sub-ZK and U-KS with nBB extraction (U-nBB-KS). Namely, the prover achieves ZK without trusting the CRS generator and the verifier achieves nBB-KS without trusting the CRS generator but by one-time CRS updating. Recent constructions in this direction have better efficiency [MBKM19,GWC19]. In this direction, recently, Abdolmaleki, Ramacher, and Slamanig [ARS20a] presented a construction, called Lamassu, and showed that using a similar folklore OR technique [BG90,DS16,Bag19b] any zk-SNARK that satisfies Sub-ZK (in item viii) and U-nBB-KS (in item ix), can be lifted to achieve Sub-ZK and U-nBB-SE. (xi) U-nBB-SE ensures that the protocol guarantees SE (in item vi) with nBB extraction as long as the initial CRS generation or one of CRS updates is done by an honest party. Considering the

[5] Note that, as also shown in Lemma 2 in [GKM$^+$18], Sub-ZK is a stronger notion than U-ZK, as in Sub-ZK $\mathcal{A}$ has generated the CRS, while the later achieves ZK if the initial CRS generation or at least one of CRS updates is done honestly.

impossibility of achieving Sub-ZK and BB extraction at the same time [BFS16], such constructions (either with universal string [ARS20a] or with two-phase update [BGM17,BG18,BKSV20,BPR20]) achieve the strongest notion with nBB extraction, but still they cannot be deployed in UC-protocols directly.

**Using zk-SNARKs in UC-Protocols.** During the last decade, zk-SNARKs have made huge impact in various applications [PHGR13,BCG$^+$14,KMS$^+$16], [KMS$^+$16,JKS16,Woo14,KKKZ19]. Some of those protocols including privacy-preserving smart contract systems Hawk [KMS$^+$16] and Gyges [JKS16], and private proof-of-stake system Ouroboros Crypsinous [KKKZ19] are constructed to achieve UC-security [Can01]. Thus, the composability of the deployed zk-SNARK was imperative in designing the main protocol.

Universal Composability or UC is a very powerful security guarantee in constructing cryptographic primitives and protocols, which is proposed by Canetti in [Can01]. A UC primitive/protocol does not interfere with other primitives/protocols and can be composed arbitrary number of times with other protocols. To prove that a cryptographic primitive achieves UC-security, one needs to show that the target primitive securely realizes the ideal functionality defined for that primitive [Can01]. In 2006, Groth [Gro06] showed that a NIZK argument that can achieve BB-SE can realize the ideal NIZK-functionality $\mathcal{F}_{\mathsf{NIZK}}$ [GOS06]. This basically showed that to be able to use NIZK arguments in UC-protocols, the target NIZK argument should achieve BB-SE. Following this result, in 2015 Kosba at al. [KZM$^+$15] proposed a framework called C∅C∅ along with several constructions that allows lifting a sound NIZK argument to a BB-SE NIZK argument, such that the lifted version can be deployed in UC-protocols. In summary, given a sound NIZK argument for language $\mathbf{L}$, the C∅C∅ defines a new extended language $\mathbf{L}'$ appended with some primitives and returns a NIZK argument that can achieve BB-SE. The strongest construction of the C∅C∅ framework is reviewed in App. A.

Unfortunately, the default security of zk-SNARKs is insufficient to be directly deployed in UC-protocols. The reason is that zk-SNARK achieves nBB extraction and the extractor $\mathsf{Ext}_{\mathcal{A}}$ requires access to the source code and random coins of $\mathcal{A}$, while in UC-secure NIZK arguments, the simulator of *ideal-world* should be able to simulate corrupted parties. To do so, the simulator should be able to extract witnesses without getting access to the source code of the environment's algorithm. Due to this fact, all those UC-secure applications that use zk-SNARKs [KMS$^+$16,JKS16,KKKZ19], use C∅C∅ to lift the underlying zk-SNARK to achieve BB-SE, equivalently UC-security [Gro06]. Note that the lifted zk-SNARKs that achieve BB-SE are not *witness* succinct any more, but they still are *circuit* succinct.

**Problem statement.** Currently several popular UC-secure protocols are using BB-SE zk-SNARKs, which are built with the C∅C∅ framework, so require a trusted setup phase. In this research, we consider if it is possible to construct an alternative to the C∅C∅ framework but in the *updatable* CRS model, such that, similarly we will be able to build BB-SE zk-SNARKs but with *updatable* parameters such that all the parties can bypass the imposed trust in the setup phase by individually updating the CRS elements.

**Our Contributions.** The core of our results is, presenting Tiramisu as an alternative to the C∅C∅ framework but in the *updatable* CRS model. Technically speaking, Tiramisu allows one to build BB-SE NIZK arguments with updatable parameters such that the parties in the protocol can update the parameters themselves instead of trusting a third party. The construction is suitable for modular use in larger cryptographic protocols, which aim to build BB-SE NIZK arguments, while avoiding to trust the parameter generators.

To construct Tiramisu , we start with the C∅C∅'s strongest construction and lift it to a construction that works in the updatable CRS model. Meanwhile, to attain fast practical performance, we consider the state-of-the-art constructions proposed in the updatable CRS model and show that we can simplify the construction of C∅C∅ and still achieve the same goal, particularly in the updatable CRS model. Technically speaking, the strongest construction of the C∅C∅ framework, gets a sound NIZK argument for the language $\mathbf{L}$ and lifts it to a new NIZK argument for the extended language $\mathbf{L}'$, that can achieve BB-SE. The language $\mathbf{L}'$ is an extension of $\mathbf{L}$ appended with some necessary and sufficient primitives, including an encryption scheme to encrypt the witness and a Pseudo-Random Function (PRF) along with a commitment scheme that commits to the secret key of the PRF (more details in App. A and Sec. 4). In composing Tiramisu , we show that considering recent developments in building NIZK arguments with updatable CRS, namely due to the existence of nBB-SE NIZK arguments with updatable CRS (either with a two-phase updatable CRS [Gro16,BGM17,BG18,BKSV20,BPR20] or with a universal updatable string [GKM$^+$18,ARS20a,ARS20c]) we can simplify the definition of $\mathbf{L}'$ by removing the commitment and PRF and construct more efficient BB-SE NIZK arguments, additionally with *updatable* parameters. We show that, Tiramisu also can be added as a layer on top of the construction proposed in [ARS20a], called Lamassu, and act as a generic compiler to lift any sound NIZK argument to an U-BB-SE NIZK argument in the updatable CRS model. However, we show that the arguments built with this approach are inefficient in comparison with the ones built with ad-hoc schemes. Fig. 1 illustrates how one can use C∅C∅ and Tiramisu to build BB-SE NIZK arguments in the *standard* and *updatable* CRS models, respectively. Similar to C∅C∅ framework, Tiramisu results in NIZK arguments whose proof size and verification time are (quasi-)linear in the *witness* size, that is an unavoidable requirement for UC security [Can01], but still are independent of the size of the circuit, which encodes $\mathbf{L}'$.

Constructing Tiramisu shows that one can bypass a known negative result in the standard CRS model. In [BFS16], Bellare et al. observed that achieving Sub-ZK and BB extractability is impossible at the same time. As BB extractability requires the simulator to create a CRS with a trapdoor it withholds, then it can extract the witness from a valid proof. But Sub-ZK requires that even if $\mathcal{A}$ generates the CRS, it should not be able to learn about the witnesses from the proof. However, if a NIZK argument achieves BB extractability, an adversary

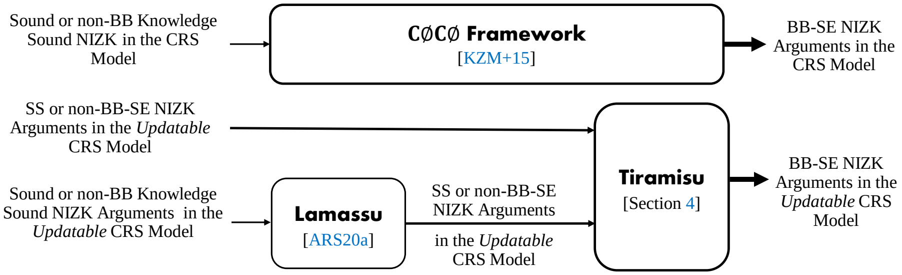
Sound or non-BB Knowledge Sound NIZK in the CRS Model
CØCØ Framework
[KZM+15]
BB-SE NIZK Arguments in the CRS Model
SS or non-BB-SE NIZK Arguments in the *Updatable* CRS Model
Tiramisu
[Section 4]
BB-SE NIZK Arguments in the *Updatable* CRS Model
Sound or non-BB Knowledge Sound NIZK Arguments in the *Updatable* CRS Model
Lamassu
[ARS20a]
SS or non-BB-SE NIZK Arguments
in the *Updatable* CRS Model

Fig. 1: Using CØCØ and TIRAMISU to build BB-SE NIZK arguments in the *standard* and *updatable* CRS models. TIRAMISU can be instantiated with either ad-hoc or lifted constructions [BGM17,BG18,BPR20,ARS20a,ARS20c].

$$\begin{tikzcd}
\text{Sub-ZK} \arrow[rr] \arrow[dr] & & \text{Upd-ZK} \arrow[dl] \\
& \text{ZK} &
\end{tikzcd}
\qquad
\begin{tikzcd}
& \text{Upd-BB-SE} \arrow[r] \arrow[dl] \arrow[d] & \text{Upd-nBB-SE} \arrow[d] \arrow[dr] & \\
\text{Upd-BB-KS} \arrow[dr] & \text{BB-SE} \arrow[r] \arrow[d] & \text{nBB-SE} \arrow[d] & \text{Upd-nBB-KS} \arrow[dl] \\
& \text{BB-KS} \arrow[r] & \text{nBB-KS} &
\end{tikzcd}$$

Fig. 2: Relations between various notions achieved with TIRAMISU (with Gray background) and other subversion-resistant /updatable NIZKs. ZK: Zero-Knowledge, BB: Black-Box, nBB: non-BB, SE: Simulation Extractability, KS: Knowledge Soundness.

(CRS subvertor) can generate the CRS like the simulator. So it has the trapdoor and can also extract the witness and break Sub-ZK. Considering the above negative result, TIRAMISU achieves the best possible combination with downgrading Sub-ZK (in item viii) to U-ZK (in item x) while achieving updatable BB extractability (either U-BB-SE or U-BB-KS). Fig. 2 illustrates the relation between the notions U-ZK, U-BB-SE, and U-BB-KS that are achieved in constructions built with TIRAMISU and other notions that typically are achieved in other subversion-resistant or updatable NIZK arguments.

TIRAMISU employs a semantically secure public-key cryptosystem with *updatable keys* that is defined in this work. It is demonstrated that such cryptosystems can be constructed from key-homomorphic encryption schemes [AHI11]. A variation of El-Gamal cryptosystem [ElG84] instantiated in the pairing-based groups which fulfill the requirements of a cryptosystem with updatable keys is presented. The new syntax can be interesting in its own right, particularly for building other primitives in the updatable CRS model, e.g. commit-and-proof systems [CFQ19], Quasi-Adaptive NIZK arguments [DGP$^+$19] or subversion-resistant commitment schemes [Bag20]. In [CHK03], Canetti, Halevi, and Katz defined forward-secure public-key encryption schemes that also support updating the secret key. More precisely, in a forward-secure encryption scheme, secret keys are updated on a regular basis such that exposure to the secret key for a given time period does not enable an adversary to break the cryptosystem for any prior time period. However, in their setting all updates are supposed to be

Table 1: A comparison of Tiramisu with related works that achieve a flavor of ZK and SE. ZK: Zero-knowledge, SE: Simulation Extractable, U: Updatable, S: Subversion, nBB: non-Black-Box, BB: Black-Box. ✓: Achieves, ×: Does not achieve.

| | Zero-Knowledge | | | Simulation Extractability | | | |
|---|---|---|---|---|---|---|---|
| | ZK | U-ZK | S-ZK | nBBSE | BBSE | U-nBBSE | U-BBSE |
| Tiramisu | ✓ | ✓ | × | ✓ | ✓ | ✓ | ✓ |
| C∅C∅ [KZM$^+$15,Bag19a] | ✓ | × | × | ✓ | ✓ | × | × |
| [GM17,BG18] | ✓ | × | × | ✓ | × | × | × |
| [Bag19b,Lip19,BPR20] | ✓ | ✓ | ✓ | ✓ | × | × | × |
| [BGM17,BG18,ARS20a] | ✓ | ✓ | ✓* | ✓ | × | ✓ | × |

*Theorem 4 in [ARS20a] (and Theorem 3 in [ARS20c]) states that their proposed construction Lamassu, can achieve U-ZK and U-nBB-SE, but it can be shown that the same construction can achieve Sub-ZK along with U-nBB-SE which is a stronger combination.

handled by a single party, hence no proof is required to ensure the correctness of key updating. Due to this fact, their definition does not fit our requirements for distributing trust across multiple updaters in the updatable CRS model. In [FMMO19], Fauzi et al. proposed an updateable key cryptosystem as well, but, much as in the previous cases, their variant is weak for our settings and cannot meet our requirements. We naturally extend their notion of *updatability* from *re-randomization* of the public-key under the same secret-key, to *updating* both public and secret keys, and proving correctness of updating similar to other primitives in the updatable CRS model [GKM$^+$18,ARS20a], while keeping the secret key hidden. These components allow us to distribute trust to the key generation phase by enabling parties to update keys without revealing their secret key while providing proof that the updating phase was executed correctly.

Tab. 1 compares the NIZK arguments built with Tiramisu with existing schemes that can achieve a flavor of SE and ZK. Since constructions built with C∅C∅ achieve BB extractability, therefore they cannot achieve S-ZK [6], and the constructions that achieve Sub-ZK [Bag19b,Lip19,ARS20a,ARS20c] can achieve (U-)nBB-SE in the best case.

The rest of the paper is organized as follows; Sec. 2 introduces notations and presents necessary preliminaries for the paper. Sec. 3 defines the syntax of a public-key cryptosystem with updatable keys and presents an efficient variation of the El-Gamal cryptosystem as an instantiation. The proposed construction Tiramisu and its security proofs are described in Sec. 4. In Sec. 5, we present two U-BB-SE NIZK arguments built with Tiramisu and discuss their deployment in UC-secure applications. Finally, we conclude the paper in Sec. 6.

[6] In the abstracts of [ARS20a,ARS20c], authors state that C∅C∅ is compatible with subversion SNARKs, but not compatible with updatable SNARKs. But, following the result of [BFS16], here we discuss and show that the C∅C∅ framework cannot be compatible with subversion SNARKs, while it can be upgraded to be compatible with updatable SNARKs.

# 2 Preliminaries

Next, we summarize our notations along with some preliminaries necessary for the paper. Throughout, we suppose the security parameter of the scheme be $\lambda$ and $\mathsf{negl}(\lambda)$ denotes a negligible function. We use $x \leftarrow\$ X$ to denote $x$ sampled uniformly according to the distribution $X$. Also, we use $[1 \mathinner{..} n]$ to denote the set of integers in range of 1 to $n$.

Let PPT and NUPPT denote probabilistic polynomial-time and non-uniform probabilistic polynomial-time, respectively. For an algorithm $\mathcal{A}$, let $\mathrm{im}(\mathcal{A})$ be the image of $\mathcal{A}$, i.e., the set of valid outputs of $\mathcal{A}$. Moreover, assume $\mathsf{RND}(\mathcal{A})$ denotes the random tape of $\mathcal{A}$, and $r \leftarrow\$ \mathsf{RND}(\mathcal{A})$ denotes sampling of a randomizer $r$ of sufficient length for $\mathcal{A}$'s needs. By $y \leftarrow \mathcal{A}(x;r)$ we mean given an input $x$ and a randomizer $r$, $\mathcal{A}$ outputs $y$. 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 "$y \leftarrow \mathcal{A}(x;r)$, $y' \leftarrow \mathsf{Ext}_{\mathcal{A}}(x;r)$". Two computationally indistinguishable distributions $A$ and $B$ are shown with $A \approx_c B$.

In pairing-based groups, we use additive notation together with the bracket notation, 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 $[a]_1 \bullet [b]_2 = \hat{e}([a]_1, [b]_2)$.

## 2.1 Zk-SNARKs in the Updatable CRS Model

We adopt the definition of NIZK arguments in the updatable CRS model from [GKM+18]. Let $\mathcal{R}$ be a relation generator, such that $\mathcal{R}(1^\lambda)$ returns a polynomial-time decidable binary relation $\mathbf{R} = \{(\mathsf{x}, \mathsf{w})\}$, where $\mathsf{x}$ is the statement and $\mathsf{w}$ is the corresponding witness. We assume one can deduce $\lambda$ from the description of $\mathbf{R}$. The relation generator also outputs auxiliary information $\xi_{\mathbf{R}}$, which both the honest parties and the adversary have access to it. $\xi_{\mathbf{R}}$ can be a value returned by $\mathsf{BGgen}(1^\lambda)$ [Gro16]. Consequently, we also give $\xi_{\mathbf{R}}$ as an input to the honest parties; if needed, one can include an additional auxiliary input to the adversary. Let $\mathbf{L_R} = \{\mathsf{x} : \exists\, \mathsf{w} \mid (\mathsf{x}, \mathsf{w}) \in \mathbf{R}\}$ be an $\mathbf{NP}$-language including all the statements which there exist corresponding witnesses in relation $\mathbf{R}$.

A *NIZK argument* $\Psi_{\mathsf{NIZK}}$ in the updatable CRS model for $\mathcal{R}$ consists of PPT algorithms $(\mathsf{K}_{\vec{\mathsf{crs}}}, \mathsf{CU}, \mathsf{CV}, \mathsf{P}, \mathsf{V}, \mathsf{Sim}, \mathsf{Ext})$, such that:

- $(\vec{\mathsf{crs}}_0, \Pi_{\vec{\mathsf{crs}}_0}) \leftarrow \mathsf{K}_{\vec{\mathsf{crs}}}(\mathbf{R}, \xi_{\mathbf{R}})$: CRS generator is a PPT algorithm that given $(\mathbf{R}, \xi_{\mathbf{R}})$, where $(\mathbf{R}, \xi_{\mathbf{R}}) \in \mathrm{im}(\mathcal{R}(1^\lambda))$, first samples the trapdoors $\vec{\mathsf{ts}}'_0$ and $\vec{\mathsf{te}}'_0$ and then uses them to generate $\vec{\mathsf{crs}}_0$ along with $\Pi_{\vec{\mathsf{crs}}_0}$ as a proof for its well-formedness. Then, stores the trapdoors associated with $\vec{\mathsf{crs}}_0$ including the simulation trapdoor $\vec{\mathsf{ts}}_0 := \vec{\mathsf{ts}}'_0$, and the extraction trapdoor $\vec{\mathsf{te}}_0 := \vec{\mathsf{te}}'_0$. Finally, it returns $(\vec{\mathsf{crs}}_0, \Pi_{\vec{\mathsf{crs}}_0})$ as the output.
- $(\vec{\mathsf{crs}}_i, \Pi_{\vec{\mathsf{crs}}_i}) \leftarrow \mathsf{CU}(\mathbf{R}, \xi_{\mathbf{R}}, \vec{\mathsf{crs}}_{i-1})$: CRS Updating is a PPT algorithm that given the tuple of $(\mathbf{R}, \xi_{\mathbf{R}}, \vec{\mathsf{crs}}_{i-1})$, where $\vec{\mathsf{crs}}_{i-1}$ is an input CRS, returns the pair of

$(\vec{\mathsf{crs}}_i, \Pi_{\vec{\mathsf{crs}}_i})$, where $\vec{\mathsf{crs}}_i$ is the updated CRS and $\Pi_{\vec{\mathsf{crs}}_i}$ is a proof that guarantees the correctness of updating. Note that after each update, the simulation and extraction trapdoors are updated, for instance $\vec{\mathsf{ts}}_i := \vec{\mathsf{ts}}_{i-1} + \vec{\mathsf{ts}}'_i$, and $\vec{\mathsf{te}}_i := \vec{\mathsf{te}}_{i-1} + \vec{\mathsf{te}}'_i$.

• $(\perp, 1) \leftarrow \mathsf{CV}(\vec{\mathsf{crs}}_i, \Pi_{\vec{\mathsf{crs}}_i})$: CRS Verification is a polynomial-time algorithm that given a potentially updated $\vec{\mathsf{crs}}_i$, and $\Pi_{\vec{\mathsf{crs}}_i}$ returns either $\perp$ on the condition that the $\vec{\mathsf{crs}}_i$ is incorrectly formed (and updated) or 1.

• $(\pi, \perp) \leftarrow \mathsf{P}(\mathbf{R}, \xi_{\mathbf{R}}, \vec{\mathsf{crs}}_i, \mathsf{x}, \mathsf{w})$: Prove is a PPT algorithm that for $\mathsf{CV}(\vec{\mathsf{crs}}_i, \Pi_{\vec{\mathsf{crs}}_i}) = 1$, given the tuple of $(\mathbf{R}, \xi_{\mathbf{R}}, \vec{\mathsf{crs}}_i, \mathsf{x}, \mathsf{w})$, such that $(\mathsf{x}, \mathsf{w}) \in \mathbf{R}$, outputs an argument $\pi$. Otherwise, it returns $\perp$.

• $(0, 1) \leftarrow \mathsf{V}(\mathbf{R}, \xi_{\mathbf{R}}, \vec{\mathsf{crs}}_i, \mathsf{x}, \pi)$: Verify is a polynomial-time algorithm that for $\mathsf{CV}(\vec{\mathsf{crs}}_i, \Pi_{\vec{\mathsf{crs}}_i}) = 1$, given the set of parameters as $(\mathbf{R}, \xi_{\mathbf{R}}, \vec{\mathsf{crs}}_i, \mathsf{x}, \pi)$, returns either 0 (reject $\pi$) or 1 (accept $\pi$).

• $(\pi) \leftarrow \mathsf{Sim}(\mathbf{R}, \xi_{\mathbf{R}}, \vec{\mathsf{crs}}_i, \vec{\mathsf{ts}}_i, \mathsf{x})$: Simulator is a PPT algorithm that for $\mathsf{CV}(\vec{\mathsf{crs}}_i, \Pi_{\vec{\mathsf{crs}}_i}) = 1$, given the tuple $(\mathbf{R}, \xi_{\mathbf{R}}, \vec{\mathsf{crs}}_i, \vec{\mathsf{ts}}_i, \mathsf{x})$, where $\vec{\mathsf{ts}}_i$ is the simulation trapdoor associated with the latest CRS, namely $\vec{\mathsf{crs}}_i$, outputs a simulated argument $\pi$.

• $(\mathsf{w}) \leftarrow \mathsf{Ext}(\mathbf{R_L}, \xi_{\mathbf{R_L}}, \vec{\mathsf{crs}}_i, \mathsf{x}, \pi, \vec{\mathsf{te}}_i)$: BB Extractor is a polynomial-time algorithm that, given $(\mathbf{R_L}, \xi_{\mathbf{R_L}}, \vec{\mathsf{crs}}_i, \mathsf{x}, \pi, \vec{\mathsf{te}}_i)$ extracts the witness $\mathsf{w}$, where $\vec{\mathsf{te}}_i$ is the extraction trapdoor associated with the latest well-form CRS, namely $\vec{\mathsf{crs}}_i$. In nBB extraction algorithms, the $\vec{\mathsf{te}}_i$ can be the source code and random coins of the adversary.

In the CRS model, a NIZK argument for $\mathcal{R}$ has a tuple of algorithms $(\mathsf{K}_{\vec{\mathsf{crs}}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim}, \mathsf{Ext})$, while subversion-resistant constructions [BFS16] additionally have a $\mathsf{CV}$ algorithm which is used to verify the well-formedness of CRS elements to achieve S-ZK [BFS16,ABLZ17,Fuc18,Bag19b]. But as listed above, in the *updatable* CRS model, a NIZK argument additionally has a $\mathsf{CU}$ algorithm that allows the parties (prover or verifier) to update the CRS elements and inject their own private shares to the CRS elements and avoid trusting a third party.

Below we recall various security requirements that a NIZK argument can satisfy in the *updatable* CRS model [GKM+18,ARS20a] and refer App. B for the standard and subversion-resistant notions.

**Definition 1 (Perfect Updatable Completeness).** *A non-interactive argument $\Psi_{\mathsf{NIZK}}$ is* perfectly updatable complete *for $\mathcal{R}$, if for all $(\mathbf{R}, \xi_{\mathbf{R}}) \in \mathrm{im}(\mathcal{R}(1^\lambda))$, and $(\mathsf{x}, \mathsf{w}) \in \mathbf{R}$, the following probability is 1 on secuirty parameter $\lambda$,*

$$\Pr\left[\begin{array}{l}(\mathbf{R}, \xi_{\mathbf{R}}) \leftarrow \mathcal{R}(1^\lambda), (\vec{\mathsf{crs}}_0, \Pi_{\vec{\mathsf{crs}}_0}) \leftarrow \mathsf{K}_{\vec{\mathsf{crs}}}(\mathbf{R}, \xi_{\mathbf{R}}),\\ (\{\vec{\mathsf{crs}}_j, \Pi_{\vec{\mathsf{crs}}_j}\}_{j=1}^{i}) \leftarrow \mathcal{A}(\mathbf{R}, \xi_{\mathbf{R}}, \vec{\mathsf{crs}}_0), \{\mathsf{CV}(\vec{\mathsf{crs}}_j, \Pi_{\vec{\mathsf{crs}}_j}) = 1\}_{j=0}^{i} :\\ (\mathsf{x}, \pi) \leftarrow \mathsf{P}(\mathbf{R}, \xi_{\mathbf{R}}, \vec{\mathsf{crs}}_i, \mathsf{x}, \mathsf{w}) \wedge \mathsf{V}(\mathbf{R}, \xi_{\mathbf{R}}, \vec{\mathsf{crs}}_i, \mathsf{x}, \pi) = 1\end{array}\right],$$

*where $\Pi_{\vec{\mathsf{crs}}_i}$ is a proof for the correctness of the initial CRS generation or CRS updating. Note that in the above definition and all the following one, $i$ is the index of final update, and without loss of generality, $\mathcal{A}$ can also first generate $\{\vec{\mathsf{crs}}_j\}_{j=0}^{i-1}$ and then an honest updater updates $\vec{\mathsf{crs}}_{i-1}$ to $\vec{\mathsf{crs}}_i$.*

**Definition 2 (Updatable Zero-Knowledge).** *A non-interactive argument* $\Psi_{\mathsf{NIZK}}$ *is* statistically updatable ZK for $\mathcal{R}$, *if for all* $(\mathbf{R}, \xi_{\mathbf{R}}) \in \mathrm{im}(\mathcal{R}(1^\lambda))$, *and for all computationally unbounded* $\mathcal{A}$, $\varepsilon_0^{unb} \approx_\lambda \varepsilon_1^{unb}$, *where* $\varepsilon_b$ *is equal to*

$$\Pr\left[\begin{array}{l}(\mathbf{R}, \xi_{\mathbf{R}}) \leftarrow \mathcal{R}(1^\lambda), ((\vec{\mathsf{crs}}_0, \Pi_{\vec{\mathsf{crs}}_0}) \parallel \vec{\mathsf{ts}}_0 := \vec{\mathsf{ts}}'_0) \leftarrow \mathsf{K}_{\vec{\mathsf{crs}}}(\mathbf{R}, \xi_{\mathbf{R}}),\\ r_s \leftarrow\!\!\$\, \mathsf{RND}(\mathsf{Sub}), ((\{\vec{\mathsf{crs}}_j, \Pi_{\vec{\mathsf{crs}}_j}\}_{j=1}^{i}, \xi_{\mathsf{Sub}}) \parallel \{\vec{\mathsf{ts}}'_j\}_{j=1}^{i})\\ \leftarrow (\mathsf{Sub} \parallel \mathsf{Ext}_{\mathsf{Sub}})(\vec{\mathsf{crs}}_0, \Pi_{\vec{\mathsf{crs}}_0}, r_s) :\\ \{\mathsf{CV}(\vec{\mathsf{crs}}_j, \Pi_{\vec{\mathsf{crs}}_j}) = 1\}_{j=0}^{i} \wedge \mathcal{A}^{\mathsf{O}_b(\cdot,\cdot)}(\mathbf{R}, \xi_{\mathbf{R}}, \xi_{\mathsf{Sub}}, \vec{\mathsf{crs}}_i) = 1\end{array}\right].$$

*Here, the oracle* $\mathsf{O}_0(\mathsf{x}, \mathsf{w})$ *returns* $\perp$ *(reject) if* $(\mathsf{x}, \mathsf{w}) \not\in \mathbf{R}$, *and otherwise it returns* $\mathsf{P}(\mathbf{R}, \xi_{\mathbf{R}}, \vec{\mathsf{crs}}_i, \mathsf{x}, \mathsf{w})$. *Similarly,* $\mathsf{O}_1(\mathsf{x}, \mathsf{w})$ *returns* $\perp$ *(reject) if* $(\mathsf{x}, \mathsf{w}) \not\in \mathbf{R}$, *and otherwise it returns* $\mathsf{Sim}(\mathbf{R}, \xi_{\mathbf{R}}, \vec{\mathsf{crs}}_i, \mathsf{x}, \vec{\mathsf{ts}}_i := \{\vec{\mathsf{ts}}'_j\}_{j=0}^{i})$, *where* $\vec{\mathsf{ts}}_i$ *is the simulation trapdoor associated with* $\vec{\mathsf{crs}}_i$ *that can be computed using* $\{\vec{\mathsf{ts}}'_j\}_{j=0}^{i}$. *We say* $\Psi_{\mathsf{NIZK}}$ *is* perfect updatable ZK for $\mathcal{R}$ *if one requires that* $\varepsilon_0 = \varepsilon_1$.

**Definition 3 (Updatable nBB Knowledge Soundness).** *A non-interactive argument* $\Psi_{\mathsf{NIZK}}$ *is* updatable non-black-box knowledge sound for $\mathcal{R}$, *if for every PPT adversary* $\mathcal{A}$ *and any subvertor* $\mathsf{Sub}$, *there exists a PPT extractor* $\mathsf{Ext}_{\mathcal{A}}$, *and the following probability is* $\mathsf{negl}(\lambda)$,

$$\Pr\left[\begin{array}{l}(\mathbf{R}, \xi_{\mathbf{R}}) \leftarrow \mathcal{R}(1^\lambda), (\vec{\mathsf{crs}}_0, \Pi_{\vec{\mathsf{crs}}_0}) \leftarrow \mathsf{K}_{\vec{\mathsf{crs}}}(\mathbf{R}, \xi_{\mathbf{R}}), r_s \leftarrow\!\!\$\, \mathsf{RND}(\mathsf{Sub}),\\ (\{\vec{\mathsf{crs}}_j, \Pi_{\vec{\mathsf{crs}}_j}\}_{j=1}^{i}, \xi_{\mathsf{Sub}}) \leftarrow \mathsf{Sub}(\vec{\mathsf{crs}}_0, \Pi_{\vec{\mathsf{crs}}_0}, r_s), \{\mathsf{CV}(\vec{\mathsf{crs}}_j, \Pi_{\vec{\mathsf{crs}}_j})\\ = 1\}_{j=0}^{i}, r_{\mathcal{A}} \leftarrow\!\!\$\, \mathsf{RND}(\mathcal{A}), ((\mathsf{x}, \pi) \parallel \mathsf{w}) \leftarrow (\mathcal{A} \parallel \mathsf{Ext}_{\mathcal{A}})\\ (\mathbf{R}, \xi_{\mathbf{R}}, \vec{\mathsf{crs}}_i, \xi_{\mathsf{Sub}}; r_{\mathcal{A}}) : (\mathsf{x}, \mathsf{w}) \not\in \mathbf{R} \wedge \mathsf{V}(\mathbf{R}, \xi_{\mathbf{R}}, \vec{\mathsf{crs}}_i, \mathsf{x}, \pi) = 1\end{array}\right],$$

*Here* $\mathsf{RND}(\mathcal{A}) = \mathsf{RND}(\mathsf{Sub})$, *and* $\Pi_{\vec{\mathsf{crs}}}$ *is a proof for correctness of CRS generation or updating process. In the definition,* $\xi_{\mathbf{R}}$ *can be seen as a common auxiliary input to* $\mathcal{A}$ *and* $\mathsf{Ext}_{\mathcal{A}}$ *that is generated by using a benign [BCPR14] relation generator and* $\xi_{\mathsf{Sub}}$ *can be auxiliary information provided by* $\mathsf{Sub}$ *to* $\mathcal{A}$.

**Definition 4 (Updatable Simulation Soundness).** *A non-interactive argument* $\Psi_{\mathsf{NIZK}}$ *is* updatable simulation soundness for $\mathcal{R}$, *if for any subvertor* $\mathsf{Sub}$, *and every PPT* $\mathcal{A}$, *the following probability is* $\mathsf{negl}(\lambda)$,

$$\Pr\left[\begin{array}{l}(\mathbf{R}, \xi_{\mathbf{R}}) \leftarrow \mathcal{R}(1^\lambda), ((\vec{\mathsf{crs}}_0, \Pi_{\vec{\mathsf{crs}}_0}) \parallel \vec{\mathsf{te}}_0) \leftarrow \mathsf{K}_{\vec{\mathsf{crs}}}(\mathbf{R}, \xi_{\mathbf{R}}),\\ r_s \leftarrow\!\!\$\, \mathsf{RND}(\mathsf{Sub}), (\{\vec{\mathsf{crs}}_j, \Pi_{\vec{\mathsf{crs}}_j}\}_{j=1}^{i}, \xi_{\mathsf{Sub}}) \leftarrow \mathsf{Sub}(\vec{\mathsf{crs}}_0, \Pi_{\vec{\mathsf{crs}}_0}, r_s),\\ \{\mathsf{CV}(\vec{\mathsf{crs}}_j, \Pi_{\vec{\mathsf{crs}}_j}) = 1\}_{j=0}^{i}, (\mathsf{x}, \pi) \leftarrow \mathcal{A}^{\mathsf{O}(.)}(\mathbf{R}, \xi_{\mathbf{R}}, \xi_{\mathsf{Sub}}, \vec{\mathsf{crs}}_i) :\\ (\mathsf{x}, \pi) \not\in Q \wedge \mathsf{x} \not\in \mathbf{L} \ \wedge \mathsf{V}(\mathbf{R}, \xi_{\mathbf{R}}, \vec{\mathsf{crs}}_i, \mathsf{x}, \pi) = 1\end{array}\right],$$

*where* $\Pi_{\vec{\mathsf{crs}}}$ *is a proof for correctness of CRS generation/updating.* $Q$ *is the set of simulated statement-proof pairs generated by* $\mathsf{O}(.)$.

**Definition 5 (Updatable nBB Simulation Extractability).** *A non-interactive argument* $\Psi_{\mathsf{NIZK}}$ *is* updatable non-black-box simulation-extractable for $\mathcal{R}$, *if for every PPT* $\mathcal{A}$ *and any subvertor* $\mathsf{Sub}$*, there exists a PPT extractor* $\mathsf{Ext}_{\mathcal{A}}$*, and the following probability is* $\mathsf{negl}(\lambda)$*,*

$$\Pr\left[\begin{array}{l}(\mathbf{R},\xi_{\mathbf{R}})\leftarrow\mathcal{R}(1^{\lambda}),(\vec{\mathsf{crs}}_0,\mathit{\Pi}_{\vec{\mathsf{crs}}_0})\leftarrow\mathsf{K}_{\vec{\mathsf{crs}}}(\mathbf{R},\xi_{\mathbf{R}}),r_s\leftarrow\$\,\mathsf{RND}(\mathsf{Sub}),\\(\{\vec{\mathsf{crs}}_j,\mathit{\Pi}_{\vec{\mathsf{crs}}_j}\}_{j=1}^{i},\xi_{\mathsf{Sub}})\leftarrow\mathsf{Sub}(\vec{\mathsf{crs}}_0,\mathit{\Pi}_{\vec{\mathsf{crs}}_0},r_s),\\\{\mathsf{CV}(\vec{\mathsf{crs}}_j,\mathit{\Pi}_{\vec{\mathsf{crs}}_j})=1\}_{j=0}^{i},r_{\mathcal{A}}\leftarrow\$\,\mathsf{RND}(\mathcal{A}),\\((\mathsf{x},\pi)\,\|\,\mathsf{w})\leftarrow(\mathcal{A}^{\mathsf{O}(.)}\,\|\,\mathsf{Ext}_{\mathcal{A}})(\mathbf{R},\xi_{\mathbf{R}},\xi_{\mathsf{Sub}},\vec{\mathsf{crs}}_i;r_{\mathcal{A}}):\\(\mathsf{x},\pi)\not\in Q\wedge(\mathsf{x},\mathsf{w})\not\in\mathbf{R}\wedge\mathsf{V}(\mathbf{R},\xi_{\mathbf{R}},\vec{\mathsf{crs}}_i,\mathsf{x},\pi)=1\end{array}\right],$$

*where* $\mathit{\Pi}_{\vec{\mathsf{crs}}}$ *is a proof for correctness of CRS generation or updating. Here,* $\mathsf{RND}(\mathcal{A})=\mathsf{RND}(\mathsf{Sub})$ *and* $Q$ *is the set of simulated statement-proof pairs returned by* $\mathcal{A}$*'s queries to* $\mathsf{O}$*.*

Note that *updatable nBB-SE* implies *updatable nBB knowledge soundness*, as in the former additionally $\mathcal{A}$ is allowed to make query to the proof simulation oracle. It also implies *updatable simulation soundness*, as if there exists a witness $\mathsf{w}$ s.t. $(\mathsf{x},\mathsf{w})\in\mathbf{R}$, therefore the instance $\mathsf{x}$ belongs to the language [Gro06]. Next, we extended the definition of updatable nBB-SE (in definition 5 to the updatable BB-SE (U-BB-SE) which constructions with Tiramisu can achieve.

**Definition 6 (Updatable Black-Box Simulation Extractability).** *A non-interactive argument* $\Psi_{\mathsf{NIZK}}$ *is* updatable black-box (strong) simulation-extractable *for* $\mathcal{R}$*, if for every PPT* $\mathcal{A}$ *and any subvertor* $\mathsf{Sub}$*, there exists a PPT extractor* $\mathsf{Ext}$ *and the following probability is* $\mathsf{negl}(\lambda)$*,*

$$\Pr\left[\begin{array}{l}(\mathbf{R},\xi_{\mathbf{R}})\leftarrow\mathcal{R}(1^{\lambda}),((\vec{\mathsf{crs}}_0,\mathit{\Pi}_{\vec{\mathsf{crs}}_0})\,\|\,\vec{\mathsf{te}}_0:=\vec{\mathsf{te}}'_0)\leftarrow\mathsf{K}_{\vec{\mathsf{crs}}}(\mathbf{R},\xi_{\mathbf{R}}),\\r_s\leftarrow\$\,\mathsf{RND}(\mathsf{Sub}),((\{\vec{\mathsf{crs}}_j,\mathit{\Pi}_{\vec{\mathsf{crs}}_j}\}_{j=1}^{i},\xi_{\mathsf{Sub}})\,\|\,\{\vec{\mathsf{te}}'_j\}_{j=1}^{i})\\\ \leftarrow(\mathsf{Sub}\,\|\,\mathsf{Ext}_{\mathsf{Sub}})(\vec{\mathsf{crs}}_0,\mathit{\Pi}_{\vec{\mathsf{crs}}_0},r_s),\{\mathsf{CV}(\vec{\mathsf{crs}}_j,\mathit{\Pi}_{\vec{\mathsf{crs}}_j})=1\}_{j=0}^{i},\\r_{\mathcal{A}}\leftarrow\$\,\mathsf{RND}(\mathcal{A}),(\mathsf{x},\pi)\leftarrow\mathcal{A}^{\mathsf{O}(.)}(\mathbf{R},\xi_{\mathbf{R}},\vec{\mathsf{crs}}_i,\xi_{\mathsf{Sub}};r_{\mathcal{A}}),\\\mathsf{w}\leftarrow\mathsf{Ext}(\mathbf{R},\xi_{\mathbf{R}},\vec{\mathsf{crs}}_i;\vec{\mathsf{te}}_i:=\{\vec{\mathsf{te}}'_j\}_{j=0}^{i}):\\(\mathsf{x},\pi)\not\in Q\wedge(\mathsf{x},\mathsf{w})\not\in\mathbf{R}\ \wedge\mathsf{V}(\mathbf{R},\xi_{\mathbf{R}},\vec{\mathsf{crs}}_i,\mathsf{x},\pi)=1\end{array}\right],$$

*where* $\mathit{\Pi}_{\vec{\mathsf{crs}}}$ *is a proof for correctness of CRS generation/updating and* $\vec{\mathsf{te}}_i$ *is the extraction trapdoor associated with the final CRS that can be computed using* $\{\vec{\mathsf{te}}'_j\}_{j=0}^{i}$*. Here,* $\mathsf{RND}(\mathcal{A})=\mathsf{RND}(\mathsf{Sub})$ *and* $Q$ *is the set of the statment and simulated proofs returned by* $\mathsf{O}$*.*

Our definition of U-BB-SE is inspired from the standard definition of BB-SE given in [Gro06]. We notice that *updatable BB simulation extractability* implies *updatable nBB simulation extractability* (given in Def. 5). One can also define Updatable BB Knowledge Soundness (U-BB-KS) as a weaker version of U-BB-SE, where in the former, $\mathcal{A}$ would not have access to the oracle $\mathsf{O}(\cdot)$.

### 2.2 Public-key Cryptosystems

**Definition 7 (Public-key Cryptosystem).** *A public-key cryptosystem $\Psi_{\mathsf{Enc}}$ over the message space of $\mathcal{M}$ and ciphertext space of $\mathcal{C}$, consists of three PPT algorithms defined as follows,*

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KG}(1^\lambda)$*: Key Generation is a PPT that given security parameter* $1^\lambda$ *returns a key-pair* $(\mathsf{pk}, \mathsf{sk})$*.*
- $(c) \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$*: Encryption is a PPT algorithm that given a public-key* $\mathsf{pk}$ *and a message* $m \in \mathcal{M}$*, outputs a ciphertext* $c \in \mathcal{C}$*.*
- $(\perp, m) \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$*: Decryption is a deterministic algorithm that given a ciphertext* $c \in \mathcal{C}$ *and a secret-key* $\mathsf{sk}$*, returns either* $\perp$ *(reject) or* $m \in \mathcal{M}$ *(successful).*

The primary security requirements for an encryption scheme is correctness and INDistinguishability Under Chosen Plaintext Attacks (IND-CPA) that are defined as below.

**Definition 8 (Perfect Correctness).** *A public-key cryptosystem* $\Psi_{\mathsf{Enc}} := (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ *satisfies perfect correctness, if*

$$\Pr\left[(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KG}(1^\lambda), c = \mathsf{Enc}(\mathsf{pk}, m) : \mathsf{Dec}(\mathsf{sk}, c) = m\right] = 1 \ .$$

*where the probability is taken over the randomness of the encryption algorithm.*

**Definition 9 (IND-CPA Security).** *A public-key cryptosystem* $\Psi_{\mathsf{Enc}} := (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ *satisfies IND-CPA, if for all PPT adversaries* $\mathcal{A}$*,*

$$\Pr\left[\begin{matrix}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KG}(1^\lambda), b \leftarrow_\$ \{0,1\}, (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pk}), \\ b' \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, m_b)) : b = b' \end{matrix}\right] \approx_\lambda \frac{1}{2} \ .$$

*El-Gamal Cryptosystem.* One of the known IND-CPA secure cryptosystems is proposed by El-Gamal [ElG84] that its algorithms $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ work as below,

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KG}(1^\lambda)$: Given the security parameter $1^\lambda$, generate an efficient description of a cyclic group $\mathbb{G}$ of order $p$ with generator $g$; sample $\mathsf{sk} \leftarrow_\$ \mathbb{Z}_p^*$ and set $h = g^{\mathsf{sk}}$; return $(\mathsf{pk}, \mathsf{sk}) := ((g, h), \mathsf{sk})$.
- $(c) \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$: Given $\mathsf{pk} := (g, h)$ and a message $m \in \mathcal{M}$, sample a randomness $r \leftarrow_\$ \mathbb{Z}_p^*$ and return $c := (c_1, c_2) := (m \cdot h^r, g^r)$.
- $(\perp, m) \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$: Given $\mathsf{sk}$ and a ciphertext $c := (c_1, c_2) := (m \cdot h^r, g^r)$ returns, $m := c_1 / c_2^{\mathsf{sk}} = m \cdot h^r / g^{\mathsf{sk} r}$.

### 2.3 Key-Homomorphic Cryptosystems

Let $\Psi_{\mathsf{Enc}} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ be a key-homomorphic cryptosystem and the secret and public keys are chosen from the cyclic groups of $(\mathbb{H}, +)$ and $(\mathbb{G}, \cdot)$, respectively.

**Definition 10 (Secret-key to Public-key Homomorphisms [TW14]).** *We say the cryptosystem $\Psi_{\mathsf{Enc}}$ over the message space $\mathcal{M}$ admits a secret-key to public-key homomorphism if there exists a map $\mu : \mathbb{H} \rightarrow \mathbb{G}$ such that:*

- *$\mu$ is a homomorphism. i.e., for all $\mathsf{sk}, \mathsf{sk}' \in \mathbb{H}$, we have $\mu(\mathsf{sk} + \mathsf{sk}') = \mu(\mathsf{sk}) \cdot \mu(\mathsf{sk}')$;*
- *Every output $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KG}(1^\lambda)$, satisfies $\mathsf{pk} = \mu(\mathsf{sk})$.*

In particular, similar to Def. 8, such construction satisfies *completeness* if for a valid secret key $\mathsf{sk}$ output by $\mathsf{KG}$, the probability $\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mu(\mathsf{sk}), m)) \neq m]$ is negligible for all messages $m \in \mathcal{M}$, where the probability is over the coins of $\mathsf{Enc}$. It satisfies *perfect completeness* if the probability is zero.

In the discrete logarithm setting, it is usually the case $\mathsf{sk} \in \mathbb{Z}_p^*$ and $\mathsf{pk} := g^{\mathsf{sk}}$ such that $g$ is the generator of a cyclic group $\mathbb{G}$ of prime order $p$, e.g., for El-Gamal cryptosystem [ElG84].

**Definition 11 (Key-Homomorphic Cryptosystems [AHI11]).** *We say $\Psi_{\mathsf{Enc}}$ over $\mathcal{M}$ and $\mathcal{C}$ satisfies key-homomorphism property, if there exists an efficient algorithm $\mathsf{Adapt}$, that given $\mathsf{pk}$, $c \in \mathcal{C}$, and a shift amount $\Delta \in \mathbb{H}$, it returns a new $\mathsf{pk}'$ and a new ciphertext $c' \in \mathcal{C}$, namely $(\mathsf{pk}', c') \leftarrow \mathsf{Adapt}(\mathsf{pk}, c, \Delta)$, such that for every $\Delta \in \mathbb{H}$, for all $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KG}(1^\lambda)$ and message $m \in \mathcal{M}$ we have,*

$$(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk} \cdot \mu(\Delta), m)) \approx_\lambda (\mathsf{pk}', c') \leftarrow \mathsf{Adapt}(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, m), \Delta)$$

*and the distribution is induced by randomnesses of $\mathsf{Enc}$ and $\mathsf{Adapt}$.*

For instance, an $\mathsf{Adapt}$ algorithm for El-Gamal [ElG84] cryptosystem (that we later use in Sec. 3) can be written below,

- $(\mathsf{pk}', c') \leftarrow \mathsf{Adapt}(\mathsf{pk}, c, \Delta)$: Given $\mathsf{pk} := (g, h := g^{\mathsf{sk}})$, $c := (c_1, c_2) = (mh^r, g^r)$, sample the shift parameter $\Delta \leftarrow\$ \mathbb{Z}_p^*$, and computes $\mathsf{pk}' := (g, h' := h \cdot g^\Delta)$; $c' := (c_1', c_2') = (mh^r \cdot g^{r\Delta}, g^r)$; Return $(\mathsf{pk}', c')$.

### 2.4 Assumptions

**Definition 12 (Bilinear Diffie-Hellman Knowledge of Exponent (BDH-KE) Assumption).** *We say $\mathsf{BGgen}$ is* BDH-KE secure *for relation set $\mathcal{R}$ if for any $\lambda$, $(\mathbf{R}, \xi_{\mathbf{R}}) \in \operatorname{im}(\mathcal{R}(1^\lambda))$, and PPT adversary $\mathcal{A}$ there exists a PPT extractor $\mathsf{Ext}_{\mathcal{A}}$, such that,*

$$\Pr\left[\begin{array}{l}(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2) \leftarrow \mathsf{BGgen}(1^\lambda), r \leftarrow\$ \mathsf{RND}(\mathcal{A}), \\ ([\alpha_1]_1, [\alpha_2]_2 \parallel a) \leftarrow (\mathcal{A} \parallel \mathsf{Ext}_{\mathcal{A}})(\mathbf{R}, \xi_{\mathbf{R}}; r) : \\ [\alpha_1]_1 \bullet [1]_2 = [1]_1 \bullet [\alpha_2]_2 \wedge a \neq \alpha_1\end{array}\right] \approx_\lambda 0 \ .$$

*Where $\xi_{\mathbf{R}}$ is the auxiliary information related to the underlying group.*

The BDH-KE assumption [ABLZ17] is an asymmetric-pairing version of the original knowledge assumption [Dam92].

# 3 Public-Key Cryptosystems with Updatable Keys

As briefly discussed in Sec. 1, one of the key building blocks used in Tiramisu is the cryptosystem schemes with updatable keys that we define in this section. Such definitions recently are proposed for signatures [ARS20a], but to the best of our knowledge this is the first time that this notion is defined for the public-key cryptosystems. In contrast to subversion-resilient encryption schemes [ABK18] that the key-generation phase might be subverted, here we consider the case that the output of the key-generation phase is updatable and parties can update the keys. We aim to achieve the standard security requirements of a cryptosystem as long as either the original key generation or at least one of the updates was done honestly.

## 3.1 Definition and Security Requirements

**Definition 13 (Public-key Cryptosystems with Updatable Keys).** *A public-key cryptosystem $\Psi_{\mathsf{Enc}}$ with updatable keys over the message space of $\mathcal{M}$ and ciphertext space of $\mathcal{C}$, consists of five PPT algorithms* $(\mathsf{KG}, \mathsf{KU}, \mathsf{KV}, \mathsf{Enc}, \mathsf{Dec})$ *that are defined as follows,*

- $(\mathsf{pk}_0, \Pi_{\mathsf{pk}_0}, \mathsf{sk}_0) \leftarrow \mathsf{KG}(1^\lambda)$*: Key Generation is a PPT algorithm that given the security parameter* $1^\lambda$ *returns the corresponding key pair* $(\mathsf{pk}_0, \mathsf{sk}_0)$ *and* $\Pi_{\mathsf{pk}_0}$ *as a proof of correctness.*
- $(\mathsf{pk}_i, \Pi_{\mathsf{pk}_i}) \leftarrow \mathsf{KU}(\mathsf{pk}_{i-1})$*: Key Updating is a PPT algorithm that given a valid (possibly updated) public key* $\mathsf{pk}_{i-1}$ *(first time* $i = 1$*) outputs* $(\mathsf{pk}_i, \Pi_{\mathsf{pk}_i})$*, where* $\mathsf{pk}_i$ *denotes the updated public-key (with a hidden secret-key* $\mathsf{sk}_i$*) and* $\Pi_{\mathsf{pk}_i}$ *is a proof for the correctness of the updating process.*
- $(1, \perp) \leftarrow \mathsf{KV}(\mathsf{pk}_i, \Pi_{\mathsf{pk}_i})$*: Key Verification is a polynomial-time algorithm that given a potentially updated* $\mathsf{pk}_i$ *and* $\Pi_{\mathsf{pk}_i}$*, checks the validity of the updated key. It returns either* $\perp$ *on the condition that the* $\mathsf{pk}_i$ *is incorrectly formed (and updated) otherwise it outputs* $1$.
- $(c) \leftarrow \mathsf{Enc}(\mathsf{pk}_i, m)$*: Encryption is a randomize algorithm that given a potentially updated public key* $\mathsf{pk}_i$ *and a message* $m \in \mathcal{M}$*, outputs a ciphertext* $c \in \mathcal{C}$.
- $(\perp, m') \leftarrow \mathsf{Dec}(\mathsf{sk}_i, c)$*: Decryption is a deterministic algorithm that given a ciphertext* $c \in \mathcal{C}$ *and a potentially updated secret key* $\mathsf{sk}_i$*, returns either* $\perp$ *(reject) or* $m' \in \mathcal{M}$ *(successful). Note that in the standard public-key cryptosystems (and in this definition before any updating)* $sk_i = sk_0$.

Primary requirements for a cryptosystem with updatable keys $\Psi_{\mathsf{Enc}} := (\mathsf{KG}, \mathsf{KU}, \mathsf{KV}, \mathsf{Enc}, \mathsf{Dec})$ can be considered as follows,

**Definition 14 (Perfect Updatable Correctness).** *A cryptosystem $\Psi_{\mathsf{Enc}}$ with updatable keys is perfect updatable correct, if*

$$\Pr\left[\begin{array}{l}(\mathsf{pk}_0, \Pi_{\mathsf{pk}_0}, \mathsf{sk}_0 := \mathsf{sk}'_0) \leftarrow \mathsf{KG}(1^\lambda), r_s \leftarrow\$\, \mathsf{RND}(\mathsf{Sub}),\\ ((\{\mathsf{pk}_j, \Pi_{\mathsf{pk}_j}\}_{j=1}^{i}, \xi_{\mathsf{Sub}}) \parallel \{\mathsf{sk}'_j\}_{j=1}^{i}) \leftarrow\\ (\mathsf{Sub} \parallel \mathsf{Ext}_{\mathsf{Sub}})(\mathsf{pk}_0, \Pi_{\mathsf{pk}_0}, r_s), \{\mathsf{KV}(\mathsf{pk}_j, \Pi_{\mathsf{pk}_j}) = 1\}_{j=0}^{i} :\\ \mathsf{Dec}(\mathsf{sk}_i := \{\mathsf{sk}'_j\}_{j=0}^{i}, \mathsf{Enc}(\mathsf{pk}_i, m)) = m\end{array}\right] = 1 \ .$$

*where $\mathsf{sk}'_j$ is the individual secret-key of each party and $\mathsf{pk}_i$ is the final public-key. The probability is taken over the randomness of the encryption algorithm. Note that $\mathsf{Sub}$ can also first generate $\{\mathsf{pk}_j\}_{j=0}^{i-1}$ and then an honest updater updates $\mathsf{pk}_{i-1}$ to $\mathsf{pk}_i$.*

**Definition 15 (Updatable Key Hiding).** *In a cryptosystem $\Psi_{\mathsf{Enc}}$ with updatable keys, for $(\mathsf{pk}_0, \Pi_{\mathsf{pk}_0}, \mathsf{sk}_0 := \mathsf{sk}'_0) \leftarrow \mathsf{KG}(1^\lambda)$ and $(\mathsf{pk}_i, \Pi_{\mathsf{pk}_i}) \leftarrow \mathsf{KU}(\mathsf{pk}_{i-1})$, we say that $\Pi_{\mathsf{Enc}}$ is updatable key hiding, if one of the following cases holds,*

- *the original $\mathsf{pk}_0$ was honestly generated and $\mathsf{KV}$ algorithm returns $1$, namely $(\mathsf{pk}_0, \Pi_{\mathsf{pk}_0}, \mathsf{sk}_0) \leftarrow \mathsf{KG}(1^\lambda)$ and $\mathsf{KV}(\mathsf{pk}_0, \Pi_{\mathsf{pk}_0}) = 1$,*
- *the original $\mathsf{pk}_0$ verifies successfully with $\mathsf{KV}$ and the key-update was generated honestly once, namely $\mathsf{KV}(\mathsf{pk}_0, \Pi_{\mathsf{pk}_0}) = 1$ and $(\{\mathsf{pk}_j, \Pi_{\mathsf{pk}_j}\}_{j=1}^{i}) \leftarrow \mathsf{KU}(\mathsf{pk}_0)$ such that $\{\mathsf{KV}(\mathsf{pk}_j, \Pi_{\mathsf{pk}_j}) = 1\}_{j=1}^{i}$.*

**Definition 16 (Updatable IND-CPA).** *A public-key cryptosystem $\Psi_{\mathsf{Enc}}$ with updatable keys satisfies updatable IND-CPA, if for all PPT subvertor $\mathsf{Sub}$, for all $\lambda$, and for all PPT adversaries $\mathcal{A}$,*

$$\Pr\left[\begin{array}{l}(\mathsf{pk}_0, \Pi_{\mathsf{pk}_0}, \mathsf{sk}_0 := \mathsf{sk}'_0) \leftarrow \mathsf{KG}(1^\lambda), r_s \leftarrow\$\, \mathsf{RND}(\mathsf{Sub}),\\ (\{\mathsf{pk}_j, \Pi_{\mathsf{pk}_j}\}_{j=1}^{i}, \xi_{\mathsf{Sub}}) \leftarrow \mathsf{Sub}(\mathsf{pk}_0, \Pi_{\mathsf{pk}_0}, r_s), b \leftarrow\$\, \{0,1\},\\ (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pk}_i, \xi_{\mathsf{Sub}}), b' \leftarrow \mathcal{A}(\mathsf{Enc}(\mathsf{pk}_i, m_b)) :\\ \ \{\mathsf{KV}(\mathsf{pk}_j, \Pi_{\mathsf{pk}_j}) = 1\}_{j=0}^{i} \wedge b' = b\end{array}\right] \approx_\lambda \frac{1}{2} \ ,$$

*where $\xi_{\mathsf{Sub}}$ is the auxiliary information which is returned by the subvertor $\mathsf{Sub}$. Note that $\mathsf{Sub}$ can also generate the initial $\mathsf{pk}_0$ and then an honest key updater $\mathsf{KU}$ updates it and outputs $\mathsf{pk}_i$ (associated with the secret key $\mathsf{sk}_i := \{\mathsf{sk}'_j\}_{j=0}^{i}$), and the proof $\Pi_{\mathsf{pk}_i}$ (then we require that $\mathsf{KV}(\mathsf{pk}_0, \Pi_{\mathsf{pk}_0}) = 1$). The $\mathsf{Sub}$ can also first generate $\{\mathsf{pk}_j\}_{j=0}^{i-1}$ and then an honest updater updates $\mathsf{pk}_{i-1}$ to $\mathsf{pk}_i$.*

In the rest, we prove the following theorem which gives a generic approach for building a public-key cryptosystem with updatable keys using the key-homomorphic cryptosystems [AHI11].

**Theorem 1 (Cryptosystem with Updatable Keys).** *Every correct, IND-CPA secure, and key-homomorphic cryptosystem $\Psi_{\mathsf{Enc}}$ with an efficient extractor $\mathsf{Ext}_{\mathsf{Sub}}$, satisfies updatable correctness, updatable key hiding and updatable IND-CPA security.*

*Proof.* To consider the updatable correctness, let the key updating and verification algorithms $\mathsf{KU}$ and $\mathsf{KV}$ be defined as follows,

- $(\mathsf{pk}_i, \Pi_{\mathsf{pk}_i}) \leftarrow \mathsf{KU}(\{\mathsf{pk}_j\}_{j=0}^{i-1})$: Given the public keys $\{\mathsf{pk}_j\}_{j=0}^{i-1}$, where the $\mathsf{pk}_{i-1}$ is the latest updated public-key, act as follows: choose $\Delta \leftarrow\$ \mathbb{H}$; set $\mathsf{sk}'_i := \Delta$, where $\mathsf{sk}'_i$ is the secret key of the updater; set $\mathsf{pk}_i = \mathsf{pk}_{i-1} \cdot \mu(\Delta)$ and $\Pi_{\mathsf{pk}_i} := \mu(\Delta)$; Output $(\mathsf{pk}_i, \Pi_{\mathsf{pk}_i})$, where $\mathsf{pk}_i$ denotes the updated public and $\Pi_{\mathsf{pk}_i}$ is a proof for the correctness of the updating process.
- $(1, \perp) \leftarrow \mathsf{KV}(\{\mathsf{pk}_j, \Pi_{\mathsf{pk}_j}\}_{j=0}^{i})$: Given a potentially updated $\mathsf{pk}_i$ along with previous keys $\{\mathsf{pk}_j\}_{j=0}^{i-1}$, and $\Pi_{\mathsf{pk}_i}$ (along with $\{\Pi_{\mathsf{pk}_j}\}_{j=0}^{i-1}$), returns 1 either if $\mathsf{pk}_i = \mathsf{pk}_0$ or $\mathsf{pk}_i := \mathsf{pk}_{i-1} \cdot \Pi_{\mathsf{pk}_i}$, otherwise it responses by $\perp$.

One can see that $\mathsf{sk}_i := \mathsf{sk}_{i-1} + \Delta := \mathsf{sk}_{i-1} + \mathsf{sk}'_i$, where $\mathsf{sk}_i$ is the secret key associated with $\mathsf{pk}_i$, $\mathsf{sk}_{i-1}$ is the secret key associated with $\mathsf{pk}_{i-1}$, and $\mathsf{sk}'_i := \Delta$ is the secret-key of the updater. Consequently, due to the existence of $\mathsf{Ext}_{\mathsf{Sub}}$, (which allows to extract all the secret keys injected in the key updates by $\mathsf{Sub}$, namely $\{\mathsf{sk}'_j\}_{j=1}^{i}$) the updatable correctness follows from the correctness of $\Psi_{\mathsf{Enc}}$.

Updatable key hiding directly comes from the key-homomorphic property (in Def. 11) of the cryptosystem $\Psi_{\mathsf{Enc}}$, and the algorithms $\mathsf{KU}$ and $\mathsf{KV}$ required in Def. 15 act as defined above.

Next, we prove updatable IND-CPA security by a reduction to the IND-CPA security of the cryptosystem $\Psi_{\mathsf{Enc}}$. Suppose $\mathcal{A}$ is a successful adversary against updatable IND-CPA of $\Psi_{\mathsf{Enc}}$. Namely, let $\mathsf{pk}_0$ be the public-key generated by challenger of $\Psi_{\mathsf{Enc}}$, and $(\{\mathsf{pk}_j, \Pi_{\mathsf{pk}_j}\}_{j=1}^{i})$ be the output of $\mathcal{A}$ on input $\mathsf{pk}_0$. Then, if $\{\mathsf{KV}(\mathsf{pk}_j,$
$\Pi_{\mathsf{pk}_j}) = 1\}_{j=1}^{i}$, so one can use $\mathsf{Ext}_{\mathsf{Sub}}$ to extract $\{\mathsf{sk}'_j\}_{j=1}^{i}$ (the secret keys of $\mathsf{Sub}$ in each update) and also conclude that $\mathsf{pk}_i := \mathsf{pk}_{i-1} \cdot \Pi_{\mathsf{pk}_i}$. Next, for random bit $b \leftarrow\$ \{0, 1\}$ taken by challenger, and $(m_0, m_1)$ taken by $\mathcal{A}$, the challenger sends back $c_b = \mathsf{Enc}(\mathsf{pk}_i, m_b)$ to $\mathcal{A}$ and with non-negligible advantage, $\mathcal{A}$ guesses $b$, correctly.

Now, consider a new adversary $\mathcal{B}$ for IND-CPA of $\Psi_{\mathsf{Enc}}$ that given $\mathsf{pk}_0$ sends it to $\mathcal{A}$ and gets $(\{\mathsf{pk}_j, \Pi_{\mathsf{pk}_j}\}_{j=1}^{i})$ and $(m_0, m_1)$ from $\mathcal{A}$. Then $\mathcal{B}$ sends $(m_0, m_1)$ to the challenger and gets $c_b = \mathsf{Enc}(\mathsf{pk}_0, m_b)$ which is encrypted with $\mathsf{pk}_0$. Next, the adversary $\mathcal{B}$ uses $\mathsf{Ext}_{\mathsf{Sub}}$ and extracts all $\{\mathsf{sk}'_j\}_{j=1}^{i}$ from $\mathcal{A}$ (subvertor $\mathsf{Sub}$) and uses them to compute $\mathsf{sk}$. After that, executes $(\mathsf{pk}_i, c'_b) \leftarrow \mathsf{Adapt}(\mathsf{pk}_0, c_b, \mathsf{sk})$ and sends $c'_b$ (which is encrypted with $\mathsf{pk}_i$) to the adversary $\mathcal{A}$ and gets $b'$. Finally, adversary $\mathcal{B}$ returns the same $b'$ to the challenger and wins the IND-CPA game with the same probability that $\mathcal{A}$ wins the game updatable IND-CPA. The case that first $\mathsf{pk}_{i-1}$ is subverted and then one-time honest updating is done can be shown analogously, which is omitted. □

### 3.2 A Key Updatable Cryptosystem

Next, we show that the El-Gamal cryptosystem [ElG84] instantiated in a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2)$ can be represented as an uptable encryption scheme. In bilinear group based instantiation, in contrast to the standard

El-Gamal encryption (reviewed in Sec. 2.2)), the public key consists of a pair $([x]_1, [x]_2)$. Consequently, the algorithms of new variation can be expressed as follows,

- $(\mathsf{pk}_0, \Pi_{\mathsf{pk}_0}, \mathsf{sk}_0 := \mathsf{sk}'_0) \leftarrow \mathsf{KG}(1^\lambda)$: Given the security parameter $1^\lambda$, first obtain $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2) \leftarrow \mathsf{BGgen}(1^\lambda)$; sample $\mathsf{sk}'_0 \leftarrow\$\, \mathbb{Z}_p^*$ and return the corresponding key pair $(\mathsf{pk}_0, \mathsf{sk}_0) := \left(\left(\mathsf{pk}_0^1, \mathsf{pk}_0^2\right), \mathsf{sk}_0\right) := \left(\left(\left[\mathsf{sk}'_0\right]_1, \left[\mathsf{sk}'_0\right]_2\right), \mathsf{sk}'_0\right)$ and $\Pi_{\mathsf{pk}_0} := \left(\Pi^1_{\mathsf{pk}_0}, \Pi^2_{\mathsf{pk}_0}\right)$ $:= \left(\left[\mathsf{sk}'_0\right]_1, \left[\mathsf{sk}'_0\right]_2\right)$ as a proof of correctness (a.k.a. well-formedness).
- $(\mathsf{pk}_i, \Pi_{\mathsf{pk}_i}) \leftarrow \mathsf{KU}(\mathsf{pk}_{i-1})$: Obtain $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2) \leftarrow \mathsf{BGgen}(1^\lambda)$; then for a given $\mathsf{pk}_{i-1} := \left(\mathsf{pk}^1_{i-1}, \mathsf{pk}^2_{i-1}\right) :=$ $\left(\left[\mathsf{sk}_{i-1}\right]_1, \left[\mathsf{sk}_{i-1}\right]_2\right)$, for $i \geq 1$, sample $\mathsf{sk}'_i \leftarrow\$\, \mathbb{Z}_p^*$ and output,

$$(\mathsf{pk}_i, \Pi_{\mathsf{pk}_i}) := \left(\left(\left[\mathsf{sk}_{i-1} + \mathsf{sk}'_i\right]_1, \left[\mathsf{sk}_{i-1} + \mathsf{sk}'_i\right]_2\right), \left(\left[\mathsf{sk}'_i\right]_1, \left[\mathsf{sk}'_i\right]_2\right)\right),$$

  where $\mathsf{pk}_i := \left(\mathsf{pk}^1_i, \mathsf{pk}^2_i\right)$ denotes the updated public-key associated with the secret key $\mathsf{sk}_i := \mathsf{sk}_{i-1} + \mathsf{sk}'_i$ and $\Pi_{\mathsf{pk}_i} := \left(\Pi^1_{\mathsf{pk}_i}, \Pi^2_{\mathsf{pk}_i}\right) := \left(\left[\mathsf{sk}'_i\right]_1, \left[\mathsf{sk}'_i\right]_2\right)$ is the proof for correctness of the update.
- $(1, \perp) \leftarrow \mathsf{KV}\left(\{\mathsf{pk}_j\}_{j=0}^i, \Pi_{\mathsf{pk}_i}\right)$: Obtain $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2) \leftarrow \mathsf{BGgen}(1^\lambda)$, and then,
  - for $i = j = 0$, by giving the public key $\mathsf{pk}_0 := \left(\mathsf{pk}^1_0, \mathsf{pk}^2_0\right) := ([\mathsf{sk}_0]_1, [\mathsf{sk}_0]_2)$, and the corresponding proof $\Pi_{\mathsf{pk}_0} := \left(\Pi^1_{\mathsf{pk}_0}, \Pi^2_{\mathsf{pk}_0}\right) := \left(\left[\mathsf{sk}'_0\right]_1, \left[\mathsf{sk}'_0\right]_2\right)$, checks that,

    1) $\Pi^1_{\mathsf{pk}_0} \bullet [1]_2 \overset{?}{=} [1]_1 \bullet \mathsf{pk}^2_0$ ,

    2) $[1]_1 \bullet \Pi^2_{\mathsf{pk}_0} \overset{?}{=} \mathsf{pk}^1_0 \bullet [1]_2$ ,

    3) $[1]_1 \bullet \Pi^2_{\mathsf{pk}_0} \overset{?}{=} \Pi^1_{\mathsf{pk}_0} \bullet [1]_2$.
  - for $i \geq 1$, by taking public key $\mathsf{pk}_{i-1} := \left(\mathsf{pk}^1_{i-1}, \mathsf{pk}^2_{i-1}\right) := \left(\left[\mathsf{sk}_{i-1}\right]_1, \left[\mathsf{sk}_{i-1}\right]_2\right)$, a potentially updated public key $\mathsf{pk}_i := \left(\mathsf{pk}^1_i, \mathsf{pk}^2_i\right) := \left(\left[\mathsf{sk}_{i-1} + \mathsf{sk}'_i\right]_1, \left[\mathsf{sk}_{i-1} + \mathsf{sk}'_i\right]_2\right)$, and $\Pi_{\mathsf{pk}_i} := \left(\Pi^1_{\mathsf{pk}_i}, \Pi^2_{\mathsf{pk}_i}\right) := \left(\left[\mathsf{sk}'_i\right]_1, \left[\mathsf{sk}'_i\right]_2\right)$, checks that,

    1) $\left(\mathsf{pk}^1_{i-1} + \Pi^1_{\mathsf{pk}_i}\right) \bullet [1]_2 \overset{?}{=} [1]_1 \bullet \mathsf{pk}^2_i$,

    2) $[1]_1 \bullet \left(\mathsf{pk}^2_{i-1} + \Pi^2_{\mathsf{pk}_i}\right) \overset{?}{=} \mathsf{pk}^1_i \bullet [1]_2$ ,

    3) $[1]_1 \bullet \Pi^2_{\mathsf{pk}_i} \overset{?}{=} \Pi^1_{\mathsf{pk}_i} \bullet [1]_2$ ,

  in each case, if all the checks pass, it returns 1, otherwise $\perp$.
- $(c) \leftarrow \mathsf{Enc}(\mathsf{pk}_i, m)$: Obtain $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2) \leftarrow \mathsf{BGgen}(1^\lambda)$ and then given a (potentially updated) public key $\mathsf{pk}_i := ([\mathsf{sk}_i]_1, [\mathsf{sk}_i]_2)$, such

that $\mathsf{sk}_i := \mathsf{sk}_{i-1} + \mathsf{sk}'_i$, and a message $m \in \mathcal{M}$, samples a randomness $r \leftarrow\$ \mathbb{Z}_p^*$ and outputs the corresponding ciphertext as,

$$c := (c_1, c_2) := \left(m \cdot [r\mathsf{sk}_i]_T, [r]_T\right).$$

- $(\perp, m) \leftarrow \mathsf{Dec}(\mathsf{sk}_i, c)$: Obtain $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2) \leftarrow \mathsf{BGgen}(1^\lambda)$ and then given a ciphertext $c \in \mathcal{C}$ and a potentially updated secret key $\mathsf{sk}_i = \mathsf{sk}_{i-1} + \mathsf{sk}'_i$ it returns,

$$\frac{c_1}{c_2^{\mathsf{sk}}} = \frac{m \cdot [r\mathsf{sk}_i]_T}{[r\mathsf{sk}_i]_T} = m.$$

In the proposed construction, for the case that $\{\mathsf{KV}(\{\mathsf{pk}_j\}_{j=0}^i, \Pi_{\mathsf{pk}_i}) = 1\}_{j=0}^i$, under the BDH-KE knowledge assumption (in Def. 12) with checking $[1]_1 \bullet \Pi^2_{\mathsf{pk}_j} \overset{?}{=} \Pi^1_{\mathsf{pk}_j} \bullet [1]_2$ for $0 \leq j \leq i$, there exists an efficient nBB extractor $\mathsf{Ext}_{\mathsf{Sub}}$ that can extract all $\mathsf{sk}'_j$ from the subvertor $\mathsf{Sub}_j$. Note that here we considered the standard version of the El-Gamal cryptosystem, but we could also take its *lifted* version, which encrypts $g^m$ instead of $m$.

## 4 Tiramisu: Constructing BB-SE NIZK Arguments in the Updatable CRS Model

Next, we present Tiramisu, as a construction that allows one to build NIZK arguments in the updatable CRS model which can satisfy BB-SE (in Def. 6) and ZK (in Def. 2). Our main goal is to construct an alternative to the C∅C∅ framework but in the *updatable* CRS model, such that in new constructions the end-users can bypass the blind trust in the setup phase by one-time updating the shared parameters. Our starting point is the strongest construction of the C∅C∅ framework (reviewed in App. A) that gets a sound NIZK argument and lifts it to a BB-SE NIZK argument. To do so, given a language $\mathbf{L}$ with the corresponding $\mathbf{NP}$ relation $\mathbf{R_L}$, the C∅C∅ framework defines a new language $\mathbf{L}'$ such that $((\mathsf{x}, c, \mu, \mathsf{pk}_s, \mathsf{pk}_e, \rho), (r, r_0, \mathsf{w}, s_0)) \in \mathbf{R}_{\mathbf{L}'}$ iff:

$$c = \mathsf{Enc}(\mathsf{pk}_e, \mathsf{w}; r) \wedge \left((\mathsf{x}, \mathsf{w}) \in \mathbf{R_L} \vee (\mu = f_{s_0}(\mathsf{pk}_s) \wedge \rho = \mathsf{Com}(s_0; r_0))\right),$$

where $\{f_s : \{0,1\}^* \to \{0,1\}^\lambda\}_{s \in \{0,1\}^\lambda}$ is a pseudo-random function family, $(\mathsf{KG}_e, \mathsf{Enc}, \mathsf{Dec})$ is a set of algorithms for a semantically secure encryption scheme, $(\mathsf{KG}_s, \mathsf{Sig}_s, \mathsf{Vfy}_s)$ is a one-time signature scheme and $(\mathsf{Com}, \mathsf{Vfy})$ is a perfectly binding commitment scheme.

As a result, given a sound NIZK argument $\Psi_{\mathsf{NIZK}}$ for $\mathcal{R}$ constructed from PPT algorithms $(\mathsf{K}_{\vec{\mathsf{crs}}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim}, \mathsf{Ext})$, the C∅C∅ framework returns a BB-SE NIZK argument $\Psi'_{\mathsf{NIZK}}$ with PPT algorithms $(\mathsf{K}'_{\vec{\mathsf{crs}}}, \mathsf{P}', \mathsf{V}', \mathsf{Sim}', \mathsf{Ext}')$, where $\mathsf{K}'_{\vec{\mathsf{crs}}}$ is the CRS generator for new construction and acts as follows,

- $(\vec{\mathsf{crs}}' \parallel \vec{\mathsf{ts}}' \parallel \vec{\mathsf{te}}') \leftarrow \mathsf{K}'_{\vec{\mathsf{crs}}}(\mathbf{R_L}, \xi_{\mathbf{R_L}})$: Given $(\mathbf{R_L}, \xi_{\mathbf{R_L}})$, sample $(\vec{\mathsf{crs}} \parallel \vec{\mathsf{ts}}) \leftarrow \mathsf{K}_{\vec{\mathsf{crs}}}(\mathbf{R}_{\mathbf{L}'}, \xi_{\mathbf{R}_{\mathbf{L}'}})$; $(\mathsf{pk}_e, \mathsf{sk}_e) \leftarrow \mathsf{KG}_e(1^\lambda)$; $s_0, r_0 \leftarrow\$ \{0,1\}^\lambda$; $\rho := \mathsf{Com}$

$(s_0; r_0)$; and output $(\vec{\mathsf{crs}}' \,\|\, \vec{\mathsf{ts}}' \,\|\, \vec{\mathsf{te}}') := ((\vec{\mathsf{crs}}, \mathsf{pk}_e, \rho) \,\|\, (s_0, r_0) \,\|\, \mathsf{sk}_e)$, where $\vec{\mathsf{crs}}'$ is the CRS of $\Psi'_{\mathsf{NIZK}}$ and $\vec{\mathsf{ts}}'$ and $\vec{\mathsf{te}}'$, respectively, are the simulation trapdoor and extraction trapdoor associated with $\vec{\mathsf{crs}}'$.

Considering the description of algorithm $\mathsf{K}'_{\vec{\mathsf{crs}}}$, to construct an alternative to the C∅C∅ framework but in the *updatable* CRS model, a naive solution is to construct the three primitives above (with *gray* background) in the *updatable* CRS model, and then define a similar language but using the primitives constructed in the updatable CRS model. But, considering the fact that currently there exist efficient NIZK arguments with updatable parameters, a more efficient solution is to simplify the language $\mathbf{L}'$ and construct more efficient BB-SE NIZK arguments with updatable parameters.

Continuing the second solution, since currently there exist some ad-hoc constructions that allow two-phase updating (e.g. [BGM17,BG18,BKSV20,BPR20]) or even a lifting construction to build updatable nBB-SE zk-SNARKs in the updatable CRS model (e.g. [ARS20a,ARS20c]), therefore we simplify the original language $\mathbf{L}'$ defined in C∅C∅ and show that given a simulation sound NIZK argument with updatable parameters we can construct updatable BB-SE NIZK arguments in a more efficient manner than the mentioned naive way. To this end, we use the cryptosystem with updatable keys, defined in Sec. 3.

### 4.1 Construction

Let $\Psi_{\mathsf{Enc}} := (\mathsf{KG}, \mathsf{KU}, \mathsf{KV}, \mathsf{Enc}, \mathsf{Dec})$ be a set of algorithms for a semantically secure cryptosystem with updatable keys $(\mathsf{pk}_i, \mathsf{sk}_i)$. Similar to C∅C∅ framework, we define a new language $\mathbf{L}'$ based on the main language $\mathbf{L}$ corresponding to the input updatable nBB-SE NIZK $\Psi_{\mathsf{NIZK}} := (\mathsf{K}_{\vec{\mathsf{crs}}}, \mathsf{CU}, \mathsf{CV}, \mathsf{P}, \mathsf{V}, \mathsf{Sim}, \mathsf{Ext})$. The language $\mathbf{L}'$ is embedded with the encryption of witness with the *potentially updated* public key $\mathsf{pk}_i$ given in the CRS. Namely, given a language $\mathbf{L}$ with the corresponding $\mathbf{NP}$ relation $\mathbf{R_L}$, we define $\mathbf{L}'$ for a given random element $r \leftarrow\$\, \mathbb{F}_p$, such that $((\mathsf{x}, c, \mathsf{pk}_i), (\mathsf{w}, r)) \in \mathbf{R}_{\mathbf{L}'}$ iff:

$$c = \mathsf{Enc}(\mathsf{pk}_i, \mathsf{w}; r) \wedge (\mathsf{x}, \mathsf{w}) \in \mathbf{R_L}.$$

The intuition behind $\mathbf{L}'$ is to enforce the $\mathsf{P}$ to encrypt its witness with a potentially updated public key $\mathsf{pk}_i$, given in the CRS, and send the ciphertext $c$ along with a *simulation sound* proof. Consequently, in proving BB-SE, the updated $\mathsf{sk}_i$ of the defined cryptosystem $\Psi_{\mathsf{Enc}}$ is given to the $\mathsf{Ext}$, which makes it possible to extract the witness in a *black-box* manner. By sending the encryption of witnesses, the proof will not be *witness* succinct anymore, but still, it is succinct in the size of the circuit that encodes $\mathbf{L}'$.

In security proofs, we show that due to updatable simulation soundness (in Def. 4) of the underlying NIZK argument $\Psi_{\mathsf{NIZK}}$, the *updatable IND-CPA* security (in Def. 16) and perfect *updatable completeness* (in Def. 14) of $\Psi_{\mathsf{Enc}}$ is sufficient to achieve BB-SE in the updatable NIZK argument $\Psi'_{\mathsf{NIZK}}$ for the language $\mathbf{L}'$. By considering new language $\mathbf{L}'$, the modified construction $\Psi'_{\mathsf{NIZK}} := (\mathsf{K}'_{\vec{\mathsf{crs}}}, \mathsf{CU}', \mathsf{CV}', \mathsf{P}', \mathsf{V}', \mathsf{Sim}', \mathsf{Ext}')$ for $\mathbf{L}'$ can be written as in Fig. 3.

**CRS and trapdoor generation,** $(\vec{\mathsf{crs}}'_0, \Pi'_{\vec{\mathsf{crs}}_0}) \leftarrow \mathsf{K}'_{\vec{\mathsf{crs}}}(\mathbf{R_L}, \xi_{\mathbf{R_L}})$**:** Given $(\mathbf{R_L}, \xi_{\mathbf{R_L}}) \in \mathrm{im}(\mathcal{R}(1^\lambda))$ act as follows: execute key generation of $\Psi_{\mathsf{Enc}}$ as $(\mathsf{pk}_0, \Pi_{\mathsf{pk}_0}, \mathsf{sk}_0 := \hat{\mathsf{sk}}_0) \leftarrow \mathsf{KG}(1^\lambda)$; run CRS generator of NIZK argument $\Psi_{\mathsf{NIZK}}$ and sample $(\vec{\mathsf{crs}}_0, \Pi_{\vec{\mathsf{crs}}_0}, \vec{\mathsf{ts}}_0 := \hat{\vec{\mathsf{ts}}}_0) \leftarrow \mathsf{K}_{\vec{\mathsf{crs}}}(\mathbf{R_{L'}}, \xi_{\mathbf{R_{L'}}})$, where $\vec{\mathsf{ts}}_0$ is the simulation trapdoor associated with $\vec{\mathsf{crs}}_0$; set $(\vec{\mathsf{crs}}'_0 \parallel \Pi'_{\vec{\mathsf{crs}}_0} \parallel \vec{\mathsf{ts}}'_0 \parallel \vec{\mathsf{te}}'_0) := ((\vec{\mathsf{crs}}_0, \mathsf{pk}_0) \parallel (\Pi_{\vec{\mathsf{crs}}_0}, \Pi_{\mathsf{pk}_0}) \parallel \vec{\mathsf{ts}}_0 \parallel \mathsf{sk}_0)$; where $\Pi'_{\vec{\mathsf{crs}}_0}$ is the proof of well-formedness of $\vec{\mathsf{crs}}'_0$, $\vec{\mathsf{ts}}'_0$ is the simulation trapdoor associated with $\vec{\mathsf{crs}}'_0$, and $\vec{\mathsf{te}}'_0$ is the extraction trapdoor associated with $\vec{\mathsf{crs}}'_0$; Return $(\vec{\mathsf{crs}}'_0, \Pi'_{\vec{\mathsf{crs}}_0})$.

**CRS Updating,** $(\vec{\mathsf{crs}}'_i, \Pi'_{\vec{\mathsf{crs}}_i}) \leftarrow \mathsf{CU}'(\mathbf{R_L}, \xi_{\mathbf{R_L}}, \vec{\mathsf{crs}}'_{i-1})$**:** Given $(\mathbf{R_L}, \xi_{\mathbf{R_L}}) \in \mathrm{im}(\mathcal{R}(1^\lambda))$, and $\vec{\mathsf{crs}}'_{i-1}$ as an input CRS, act as follows: Parse $\vec{\mathsf{crs}}'_{i-1} := (\vec{\mathsf{crs}}_{i-1}, \mathsf{pk}_{i-1})$; execute $(\vec{\mathsf{crs}}_i, \Pi_{\vec{\mathsf{crs}}_i}) \leftarrow \mathsf{CU}(\mathbf{R_L}, \xi_{\mathbf{R_L}}, \vec{\mathsf{crs}}_{i-1})$; run $(\mathsf{pk}_i, \Pi_{\mathsf{pk}_i}) \leftarrow \mathsf{KU}(\mathsf{pk}_{i-1})$; set $(\vec{\mathsf{crs}}'_i \parallel \Pi'_{\vec{\mathsf{crs}}_i}) := ((\vec{\mathsf{crs}}_i, \mathsf{pk}_i) \parallel (\Pi_{\vec{\mathsf{crs}}_i}, \Pi_{\mathsf{pk}_i}))$, where $\Pi'_{\vec{\mathsf{crs}}_i}$ is the proof of well-formedness of $\vec{\mathsf{crs}}'_i$; Return $(\vec{\mathsf{crs}}'_i, \Pi'_{\vec{\mathsf{crs}}_i})$. Note that after each update, the simulation and extraction trapdoors are updated, for instance $\vec{\mathsf{ts}}'_i := \vec{\mathsf{ts}}'_{i-1} + \hat{\vec{\mathsf{ts}}}_i$, and $\vec{\mathsf{te}}'_i := \vec{\mathsf{te}}'_{i-1} + \hat{\vec{\mathsf{te}}}_i := \mathsf{sk}'_{i-1} + \hat{\mathsf{sk}}_i$.

**CRS Verify,** $(\perp, 1) \leftarrow \mathsf{CV}'(\vec{\mathsf{crs}}'_i, \Pi_{\vec{\mathsf{crs}}'_i})$**:** Given $\vec{\mathsf{crs}}'_i := (\vec{\mathsf{crs}}_i, \mathsf{pk}_i)$, and $\Pi_{\vec{\mathsf{crs}}'_i} := (\Pi_{\vec{\mathsf{crs}}_i}, \Pi_{\mathsf{pk}_i})$ act as follows: if $\mathsf{CV}(\vec{\mathsf{crs}}_i, \Pi_{\vec{\mathsf{crs}}_i}) = 1$ and $\mathsf{KV}(\mathsf{pk}_i, \Pi_{\mathsf{pk}_i}) = 1$ return 1 (i.e., the updated $\vec{\mathsf{crs}}'_i$ is correctly formed), otherwise $\perp$.

**Prover,** $(\pi', \perp) \leftarrow \mathsf{P}'(\mathbf{R_L}, \xi_{\mathbf{R_L}}, \vec{\mathsf{crs}}'_i, \mathsf{x}, \mathsf{w})$**:** Parse $\vec{\mathsf{crs}}'_i := (\vec{\mathsf{crs}}_i, \mathsf{pk}_i)$; Return $\perp$ if $(\mathsf{x}, \mathsf{w}) \notin \mathbf{R_L}$; sample $r \leftarrow\$ \{0,1\}^\lambda$; compute encryption of witnesses $c = \mathsf{Enc}(\mathsf{pk}_i, \mathsf{w}; r)$. Then execute prover $\mathsf{P}$ of the input NIZK argument $\Psi_{\mathsf{NIZK}}$ and generate $\pi \leftarrow \mathsf{P}(\mathbf{R_{L'}}, \xi_{\mathbf{R_{L'}}}, \vec{\mathsf{crs}}_i, (\mathsf{x}, c, \mathsf{pk}_i), (\mathsf{w}, r))$; and output $\pi' := (c, \pi)$.

**Verifier,** $(0, 1) \leftarrow \mathsf{V}'(\mathbf{R_L}, \xi_{\mathbf{R_L}}, \vec{\mathsf{crs}}'_i, \mathsf{x}, \pi')$**:** Parse $\vec{\mathsf{crs}}'_i := (\vec{\mathsf{crs}}_i, \mathsf{pk}_i)$ and $\pi' := (c, \pi)$; call verifier of the input NIZK argument $\Psi_{\mathsf{NIZK}}$ as $\mathsf{V}(\mathbf{R_{L'}}, \xi_{\mathbf{R_{L'}}}, \vec{\mathsf{crs}}_i, (\mathsf{x}, c, \mathsf{pk}_i), \pi)$ and returns 1 if $((\mathsf{x}, c, \mathsf{pk}_i), (\mathsf{w}, r)) \in \mathbf{R_{L'}}$, otherwise it responses by 0.

**Simulator,** $(\pi') \leftarrow \mathsf{Sim}'(\mathbf{R_L}, \xi_{\mathbf{R_L}}, \vec{\mathsf{crs}}'_i, \mathsf{x}, \vec{\mathsf{ts}}'_i)$**:** Parse $\vec{\mathsf{crs}}'_i := (\vec{\mathsf{crs}}_i, \mathsf{pk}_i)$ and $\vec{\mathsf{ts}}'_i := \vec{\mathsf{ts}}_i$; sample $z, r \leftarrow\$ \{0,1\}^\lambda$; compute $c = \mathsf{Enc}(\mathsf{pk}_i, z; r)$; execute simulator of the input NIZK argument $\Psi_{\mathsf{NIZK}}$ and generate $\pi \leftarrow \mathsf{Sim}(\mathbf{R_{L'}}, \xi_{\mathbf{R_{L'}}}, \vec{\mathsf{crs}}_i, (\mathsf{x}, c, \mathsf{pk}_i), \vec{\mathsf{ts}}_i)$; and output $\pi' := (c, \pi)$.

**Extractor,** $(\mathsf{w}) \leftarrow \mathsf{Ext}'(\mathbf{R_L}, \xi_{\mathbf{R_L}}, \vec{\mathsf{crs}}'_i, \vec{\mathsf{te}}'_i, \mathsf{x}, \pi')$**:** Parse $\pi' := (c, \pi)$ and $\vec{\mathsf{te}}'_i := \mathsf{sk}_i$; extract $\mathsf{w} \leftarrow \mathsf{Dec}(\mathsf{sk}_i, c)$; output $\mathsf{w}$.

Fig. 3: TIRAMISU : a construction for building BB-SE NIZK argument $\Psi'_{\mathsf{NIZK}}$ with updatable parameters.

## 4.2 Efficiency

In the rest, we highlight the key efficiency parameters of BB-SE NIZK arguments built with TIRAMISU .

Considering new language $\mathbf{L}'$, in new argument $\Psi'_{\mathsf{NIZK}}$ the CRS generation (CRS updating and CRS verification) of the input argument $\Psi_{\mathsf{NIZK}}$ will be done for a larger instance, and one also needs to generate (update and verify) the key pairs of the updatable public-key cryptosystem. The corresponding circuit of the newly defined language $\mathbf{L}'$, expands by the number of constraints needed for the encryption function. Recall that the language $\mathbf{L}'$ is an appended form of language $\mathbf{L}$ by encryption of witnesses. However, due to our simplifications in defining language $\mathbf{L}'$, the overhead in Tiramisu will be less than the case one uses the C∅C∅ framework. Meanwhile, as we later show in Sec.5 the efficiency of final constructions severely depends on the input NIZK argument.

The prover of the new construction $\Psi'_{\mathsf{NIZK}}$ needs to generate a proof for new language $\mathbf{L}'$ that would require extra computations. The proofs will be the proof of input nBB-SE updatable NIZK argument $\Psi_{\mathsf{NIZK}}$ appended with the ciphertext $c$ which leads to having proofs linear in *witness* size but still succinct in the *circuit* size. It is a known result that having proofs linear in witness size is an undeniable fact to achieve BB extraction and UC-security [Can01,GW11].

As the verifier is unchanged, so the verification of new constructions will be the same as NIZK $\Psi_{\mathsf{NIZK}}$ but for a larger statement.

### 4.3 Security Proof

**Theorem 2 (Perfect Updatable Completeness).** *If the input NIZK argument $\Psi_{\mathsf{NIZK}}$ guarantees perfect updatable completeness for the language $\mathbf{L}$, and the public-key cryptosystem $\Psi_{\mathsf{Enc}}$ be perfectly updatable correct, then the NIZK argument constructed in Sec. 4 for language $\mathbf{L}'$, is perfectly updatable complete.*

*Proof.* By considering the construction in Fig. 3, and the fact that both $\Psi_{\mathsf{NIZK}}$ and $\Psi_{\mathsf{Enc}}$ are *perfectly updatable correct* (given in Def. 1 and Def. 14), one can conclude the statement. Namely, if $(\vec{\mathsf{crs}}'_0, \Pi'_{\vec{\mathsf{crs}}_0}) \leftarrow \mathsf{K}'_{\vec{\mathsf{crs}}}(\mathbf{R_L}, \xi_{\mathbf{R_L}})$, $(\vec{\mathsf{crs}}'_i, \Pi'_{\vec{\mathsf{crs}}_i}) \leftarrow \mathsf{CU}'(\mathbf{R_L}, \xi_{\mathbf{R_L}}, \{\vec{\mathsf{crs}}'_j\}_{j=0}^{i-1})$ and $\left(\{\mathsf{CV}(\vec{\mathsf{crs}}'_j, \Pi_{\vec{\mathsf{crs}}'_j}) = 1\}_{j=0}^{i} \wedge (\mathsf{x}, \mathsf{w}) \in \mathbf{R}_L\right)$, then with probability 1, $\mathsf{V}'(\mathbf{R_L}, \xi_{\mathbf{R_L}}, \vec{\mathsf{crs}}'_i, \mathsf{x}, \mathsf{P}'(\mathbf{R_L}, \xi_{\mathbf{R_L}}, \vec{\mathsf{crs}}'_i, \mathsf{x}, \mathsf{w})) = 1$. □

**Theorem 3 (Computationally Updatable Zero-Knowledge).** *If the input NIZK argument $\Psi_{\mathsf{NIZK}}$ guarantees (perfect) zero-knowledge, and the public-key cryptosystem $\Psi_{\mathsf{Enc}}$ is updatable IND-CPA and satisfies updatable key hiding, then the NIZK argument constructed in Sec. 4 for language $\mathbf{L}'$ satisfies computational updatable ZK.*

*Proof.* Note that the *updatable ZK* property (in Def. 2) of the input NIZK argument $\Psi_{\mathsf{NIZK}}$ along with the *updatable* completeness of the encryption scheme $\Psi_{\mathsf{Enc}}$ imply that for one-time honest CRS generation, namely $(\vec{\mathsf{crs}}'_0, \Pi'_{\vec{\mathsf{crs}}_0}, \vec{\mathsf{ts}}'_0 := \vec{\mathsf{ts}}_0) \leftarrow \mathsf{K}'_{\vec{\mathsf{crs}}}(\mathbf{R_L}, \xi_{\mathbf{R_L}})$, and arbitrary time *acceptable* [7] (possibly malicious) CRS updating $(\{\vec{\mathsf{crs}}'_j, \Pi'_{\vec{\mathsf{crs}}_j}\}_{j=1}^{i}, \xi_{\mathsf{Sub}}) \leftarrow \mathsf{Sub}(\vec{\mathsf{crs}}'_0, \Pi'_{\vec{\mathsf{crs}}_0}, r_s)$, there exists an nBB extraction algorithm $\mathsf{Ext}_{\mathsf{Sub}}$, that given access to the source code and random coins of $\mathsf{Sub}$, under a knowledge assumption, can extract $\{\vec{\mathsf{ts}}_j\}_{j=1}^{i}$, namely $\{\vec{\mathsf{ts}}_j \leftarrow \mathsf{Ext}_{\mathsf{Sub}}(\vec{\mathsf{crs}}'_j, \Pi'_{\vec{\mathsf{crs}}_j}, r_s)\}_{j=1}^{i-1}$. So, given the $\vec{\mathsf{ts}}_0$ provided by the honest CRS generator (or an updater) along with the extracted trapdoors $\{\vec{\mathsf{ts}}_j\}_{j=1}^{i}$ from subvertor, the simulator $\mathsf{Sim}$ of argument $\Psi'_{\mathsf{NIZK}}$ can compute $\vec{\mathsf{ts}}'_i$ using $\{\vec{\mathsf{ts}}_j\}_{j=0}^{i}$ (i.e. $\vec{\mathsf{ts}}'_i = \sum_{j=0}^{i} \vec{\mathsf{ts}}_j$) and simulate the proofs as described in Fig. 3, where $\vec{\mathsf{ts}}'_i$ is the simulation trapdoor associated with final CRS $\vec{\mathsf{crs}}'_i$.

[7] By acceptable, we mean $\mathsf{CV}'$ accepts them, namely $\{\mathsf{CV}'(\vec{\mathsf{crs}}'_j, \Pi'_{\vec{\mathsf{crs}}_j}) = 1\}_{j=0}^{i}$.

Next, we write a series of hybrid experiments starting from an experiment that encrypts a random value and uses the $\mathsf{Sim}$, and finally getting to an experiment that uses the real prover. While moving on between the experiments, we show that they all are indistinguishable two-by-two. Consider the following experiments,

$\underline{\mathsf{EXP}_1^{zk}}$(simulator):

- *Setup:* $(\mathsf{pk}_0, \Pi_{\mathsf{pk}_0}, \mathsf{sk}_0) \leftarrow \mathsf{KG}(1^\lambda)$, $(\vec{\mathsf{crs}}_0, \Pi_{\vec{\mathsf{crs}}_0}, \vec{\mathsf{ts}}_0) \leftarrow \mathsf{K}_{\vec{\mathsf{crs}}}(\mathbf{R}_{\mathbf{L}'}, \xi_{\mathbf{R}_{\mathbf{L}'}})$, $r_s \leftarrow\$ \, \mathsf{RND}(\mathsf{Sub})$, $(\{\mathsf{pk}_j, \Pi_{\mathsf{pk}_j}\}_{j=1}^{i}, \xi_{\mathsf{Sub}}) \leftarrow \mathsf{Sub}(\mathsf{pk}_0, \Pi_{\mathsf{pk}_0}, r_s)$, $(\{(\vec{\mathsf{crs}}_j, \Pi_{\vec{\mathsf{crs}}_j})\}_{j=1}^{i} \parallel \{\vec{\mathsf{ts}}_j\}_{j=1}^{i}) \leftarrow (\mathsf{Sub} \parallel \mathsf{Ext}_{\mathsf{Sub}})(\vec{\mathsf{crs}}_0, \Pi_{\vec{\mathsf{crs}}_0}, r_s)$, Return $(\vec{\mathsf{crs}}'_i \parallel \Pi'_{\vec{\mathsf{crs}}_i} \parallel \vec{\mathsf{ts}}'_i) := ((\vec{\mathsf{crs}}_i, \mathsf{pk}_i) \parallel (\Pi_{\vec{\mathsf{crs}}_i}, \Pi_{\mathsf{pk}_i}) \parallel \{\vec{\mathsf{ts}}_j\}_{j=0}^{i})$;
- *Define function* $\mathsf{O}(\mathsf{x}, \mathsf{w})$ : Abort **if** $(\mathsf{x}, \mathsf{w}) \not\in \mathbf{R}_{\mathbf{L}}$; Abort **if** for any $j \in [0 \,..\, i]$, $\mathsf{CV}(\vec{\mathsf{crs}}_j, \Pi_{\vec{\mathsf{crs}}_j}) \neq 1$; Abort **if** for any $j \in [0 \,..\, i]$, $\mathsf{KV}(\mathsf{pk}_j, \Pi_{\mathsf{pk}_j}) \neq 1$; Sample $z, r \leftarrow \{0,1\}^\lambda$; $c = \mathsf{Enc}(\mathsf{pk}_i, z; r)$; $\pi \leftarrow \mathsf{Sim}(\mathbf{R}_{\mathbf{L}'}, \xi_{\mathbf{R}_{\mathbf{L}'}}, \vec{\mathsf{crs}}_i, (\mathsf{x}, c, \mathsf{pk}_i), \vec{\mathsf{ts}}'_i)$;
- $b \leftarrow \mathcal{A}^{\mathsf{O}(\mathsf{x},\mathsf{w})}(\vec{\mathsf{crs}}'_i, \Pi'_{\vec{\mathsf{crs}}_i})$;
  **return** $b$; **fi**

$\underline{\mathsf{EXP}_2^{zk}}$(simulator with witness):

- *Setup:* The same as in experiment $\mathsf{EXP}_1^{zk}$.
- *Define function* $\mathsf{O}(\mathsf{x}, \mathsf{w})$ : Abort **if** $(\mathsf{x}, \mathsf{w}) \not\in \mathbf{R}_{\mathbf{L}}$; Abort **if** for any $j \in [0 \,..\, i]$, $\mathsf{CV}(\vec{\mathsf{crs}}_j, \Pi_{\vec{\mathsf{crs}}_j}) \neq 1$; Abort **if** for any $j \in [0 \,..\, i]$, $\mathsf{KV}(\mathsf{pk}_j, \Pi_{\mathsf{pk}_j}) \neq 1$; Sample $r \leftarrow \{0,1\}^\lambda$; $c = \mathsf{Enc}(\mathsf{pk}_i, \mathsf{w}; r)$; $\pi \leftarrow \mathsf{Sim}(\mathbf{R}_{\mathbf{L}'}, \xi_{\mathbf{R}_{\mathbf{L}'}}, \vec{\mathsf{crs}}_i, (\mathsf{x}, c, \mathsf{pk}_i), \vec{\mathsf{ts}}'_i)$;
- $b \leftarrow \mathcal{A}^{\mathsf{O}(\mathsf{x},\mathsf{w})}(\vec{\mathsf{crs}}'_i, \Pi'_{\vec{\mathsf{crs}}_i})$;
  **return** $b$; **fi**

**Lemma 1.** *If the cryptosystem* $\Psi_{\mathsf{Enc}}$ *deployed in the above games satisfies updatable IND-CPA (in Def. 16) and updatable key hiding (in Def. 15), then we have* $\Pr[\mathsf{EXP}_2^{zk}] \approx_c \Pr[\mathsf{EXP}_1^{zk}]$.

*Proof.* The updatable key hiding properties of the cryptosystem $\Psi_{\mathsf{Enc}}$ guarantees that $\mathsf{pk}_0 \approx_\lambda \mathsf{pk}_i$, and the *updatable* IND-CPA of $\Psi_{\mathsf{Enc}}$ implies that no PT algorithm can distinguish an oracle that encrypts $z \leftarrow \{0,1\}^\lambda$ and uses the simulator $\mathsf{Sim}$ from the case that it encrypts witness $\mathsf{w}$ and uses $\mathsf{Sim}$.

$\underline{\mathsf{EXP}_3^{zk}}$(prover):

- *Setup:* The same as in experiment $\mathsf{EXP}_1^{zk}$ and $\mathsf{EXP}_2^{zk}$.
- *Define function* $\mathsf{O}(\mathsf{x}, \mathsf{w})$ : Abort **if** $(\mathsf{x}, \mathsf{w}) \not\in \mathbf{R}_{\mathbf{L}}$; Abort **if** for any $j \in [0 \,..\, i]$, $\mathsf{CV}(\vec{\mathsf{crs}}_j, \Pi_{\vec{\mathsf{crs}}_j}) \neq 1$; Abort **if** for any $j \in [0 \,..\, i]$, $\mathsf{KV}(\mathsf{pk}_j, \Pi_{\mathsf{pk}_j}) \neq 1$; Sample $r \leftarrow \{0,1\}^\lambda$; $c = \mathsf{Enc}(\mathsf{pk}_i, \mathsf{w}; r)$;
  $\pi \leftarrow \mathsf{P}(\mathbf{R}_{\mathbf{L}'}, \xi_{\mathbf{R}_{\mathbf{L}'}}, \vec{\mathsf{crs}}_i, (\mathsf{x}, c, \mathsf{pk}_i), (\mathsf{w}, r))$;

– $b \leftarrow \mathcal{A}^{\mathsf{O(x,w)}}(\vec{\mathsf{crs}}'_i, \Pi'_{\vec{\mathsf{crs}}_i})$;
 **return** $b$; **fi**

**Lemma 2.** *If the NIZK argument* $\Psi_{\mathsf{NIZK}}$ *used in above experiments satisfies updatable ZK, the for two experiments* $\mathsf{EXP}_3^{zk}$ *and* $\mathsf{EXP}_2^{zk}$ *we have* $\Pr[\mathsf{EXP}_3^{zk}] \approx_c \Pr[\mathsf{EXP}_2^{zk}]$.

*Proof.* The *updatable* ZK of the NIZK argument $\Psi_{\mathsf{NIZK}}$ implies that the real proof (generated by prover) in experiment $\mathsf{EXP}_3^{zk}$ is indistinguishable from the simulated proof (generated by simulator) in experiment $\mathsf{EXP}_2^{zk}$.

This completes proof of the theorem. As Lemmas 1 and 2 show that $\Pr[\mathsf{EXP}_1^{zk}] \approx \Pr[\mathsf{EXP}_2^{zk}]$ and $\Pr[\mathsf{EXP}_2^{zk}] \approx \Pr[\mathsf{EXP}_3^{zk}]$, respectively. Since the indistinguishability of experiments is transitive then we can conclude, $\Pr[\mathsf{EXP}_1^{zk}] \approx_c \Pr[\mathsf{EXP}_3^{zk}]$. □

**Theorem 4 (Updatable Black-Box Simulation Extractability).** *If the input NIZK argument* $\Psi_{\mathsf{NIZK}}$ *guarantees updatable correctness, updatable simulation soundness and updatable zero-knowledge, and the public-key cryptosystem* $\Psi_{\mathsf{Enc}}$ *satisfies updatable perfect correctness, updatable key hiding, and updatable IND-CPA, then the NIZK argument constructed in Sec. 4 for language* $\mathbf{L}'$ *satisfies updatable BB simulation extractability.*

*Proof.* Recall that the notion of updatable BB-SE guarantees that for a one time honest CRS generation/updating, even if $\mathcal{A}$ has seen an arbitrary number of simulated proofs, he cannot come up with a *fresh* valid proof unless he knows the witness. The concept of knowing is formalized by showing that there exists a BB extraction algorithm $\mathsf{Ext}$ that given extraction trapdoor generated in the setup phase, it can extract the witness from the proof. In this setting, the decryption function of cryptosystem $\Psi_{\mathsf{Enc}}$ plays the role of the mentioned $\mathsf{Ext}$, such that given the extraction trapdoor $\vec{\mathsf{te}}'_i$ (secret key) associated with the final public key $\mathsf{pk}_i$, can decrypt a valid $c$ and obtain $\mathsf{w}$. The key idea behind our construction is that in order to provide $\vec{\mathsf{te}}'_i$ to the $\mathsf{Ext}$, and $\vec{\mathsf{ts}}'_i$ to the $\mathsf{Sim}$, we use the extraction algorithm $\mathsf{Ext}_{\mathsf{Sub}}$ constructed in the setup phase of $\Psi_{\mathsf{Enc}}$ and $\Psi_{\mathsf{NIZK}}$ to extract the simulation and extraction trapdoors from the *untrusted* key generator or key updaters (maximum $i$ parties) and then along with honestly sampled simulation and extraction trapdoors (without loss of generality, $\mathsf{sk}_0$ and $\vec{\mathsf{ts}}_0$) calculate $\vec{\mathsf{te}}'_i := \{\mathsf{sk}_j\}_{j=0}^{i}$, (e.g. $\vec{\mathsf{te}}'_i = \sum_{j=0}^{i} \mathsf{sk}_j$), and $\vec{\mathsf{ts}}'_i := \{\vec{\mathsf{ts}}_j\}_{j=0}^{i}$, (e.g. $\vec{\mathsf{ts}}'_i = \sum_{j=0}^{i} \vec{\mathsf{ts}}_j$), and finally provide them to the $\mathsf{Ext}$ and $\mathsf{Sim}$. Note that, the updatable correctness of the cryptosystem $\Psi_{\mathsf{Enc}}$ and the updatable ZK of the NIZK argument $\Psi_{\mathsf{NIZK}}$ guarantee the existence of such $\mathsf{Ext}_{\mathsf{Sub}}$ for both primitives that allow to extract the extraction trapdoors $\{\mathsf{sk}_j\}_{j=1}^{i}$ and the simulation trapdoors $\{\vec{\mathsf{ts}}_j\}_{j=1}^{i}$ from $i$ malicious CRS updaters. Next, we go through a sequence of hybrid experiences which pairwise are indistinguishable. Starting from the actual experiment of BB-SE, consider the following experiments,

$\underline{\mathsf{EXP}_1^{BB-SE}}$(simulator):

- *Setup:* $(\mathsf{pk}_0, \Pi_{\mathsf{pk}_0}, \mathsf{sk}_0) \leftarrow \mathsf{KG}(1^\lambda)$, $(\vec{\mathsf{crs}}_0, \Pi_{\vec{\mathsf{crs}}_0}, \vec{\mathsf{ts}}_0) \leftarrow \mathsf{K}_{\vec{\mathsf{crs}}}(\mathbf{R}_{\mathbf{L}'}, \xi_{\mathbf{R}_{\mathbf{L}'}})$, $r_s \leftarrow\$\ \mathsf{RND}(\mathsf{Sub})$, $(\{\mathsf{pk}_j, \Pi_{\mathsf{pk}_j}\}_{j=1}^i \| \{\mathsf{sk}_j\}_{j=1}^i) \leftarrow (\mathsf{Sub} \| \mathsf{Ext}_{\mathsf{Sub}})(\mathsf{pk}_0, \Pi_{\mathsf{pk}_0}, r_s)$, $(\{\vec{\mathsf{crs}}_j, \Pi_{\vec{\mathsf{crs}}_j}\}_{j=1}^i \| \{\vec{\mathsf{ts}}_j\}_{j=1}^i) \leftarrow (\mathsf{Sub} \| \mathsf{Ext}_{\mathsf{Sub}})(\vec{\mathsf{crs}}_0, \Pi_{\vec{\mathsf{crs}}_0}, r_s)$, Return $(\vec{\mathsf{crs}}'_i \| \Pi'_{\vec{\mathsf{crs}}_i} \| \vec{\mathsf{ts}}'_i \| \vec{\mathsf{te}}'_i) := ((\vec{\mathsf{crs}}_i, \mathsf{pk}_i) \| (\Pi_{\vec{\mathsf{crs}}_i}, \Pi_{\mathsf{pk}_i}) \| \{\vec{\mathsf{ts}}_j\}_{j=0}^i \| \{\mathsf{sk}_j\}_{j=0}^i)$; where $\vec{\mathsf{ts}}'_i$ and $\vec{\mathsf{te}}'_i$ are the simulation and extraction trapdoors associate with $\vec{\mathsf{crs}}'_i$.
- *Define function* $\mathsf{O}(\mathsf{x})$: Abort **if** for any $j \in [0\mathinner{..}i]$, $\mathsf{CV}(\vec{\mathsf{crs}}_j, \Pi_{\vec{\mathsf{crs}}_j}) \neq 1$; Abort **if** for any $j \in [0\mathinner{..}i]$, $\mathsf{KV}(\mathsf{pk}_j, \Pi_{\mathsf{pk}_j}) \neq 1$; Sample $r, z \leftarrow \{0,1\}^\lambda$; $c = \mathsf{Enc}(\mathsf{pk}_i, z; r)$; $\pi \leftarrow \mathsf{Sim}(\mathbf{R}_{\mathbf{L}'}, \xi_{\mathbf{R}_{\mathbf{L}'}}, \vec{\mathsf{crs}}_i, (\mathsf{x}, c, \mathsf{pk}_i), \vec{\mathsf{ts}}'_i)$; Output $\pi' := (c, \pi)$
- $(\mathsf{x}, \pi') \leftarrow \mathcal{A}^{\mathsf{O}(\mathsf{x})}(\vec{\mathsf{crs}}'_i, \vec{\mathsf{te}}'_i)$;
- Parse $\pi' := (c, \pi)$; extract witness $\mathsf{w} \leftarrow \mathsf{Dec}(c, \vec{\mathsf{te}}'_i)$;
- **if** $((\mathsf{x}, \pi') \notin Q) \wedge\ (\mathsf{V}(\mathbf{R}_{\mathbf{L}'}, \xi_{\mathbf{R}_{\mathbf{L}'}}, \vec{\mathsf{crs}}'_i, (\mathsf{x}, c, \mathsf{pk}_i), \pi) = 1) \wedge ((\mathsf{x}, \mathsf{w}) \notin \mathbf{R}_{\mathbf{L}})$ : **return** 1.
  where $Q$ is the set of statement-proof pairs returned by $\mathsf{O}(\mathsf{x})$. **fi**

$\underline{\mathsf{EXP}_2^{BB-SE}}$(simulator while encrypting $\mathsf{w}$):

- *Setup:* The same as in experiment $\mathsf{EXP}_1^{BB-SE}$.
- *Define function* $\mathsf{O}(\mathsf{x})$: Abort **if** for any $j \in [0\mathinner{..}i]$, $\mathsf{CV}(\vec{\mathsf{crs}}_j, \Pi_{\vec{\mathsf{crs}}_j}) \neq 1$; Abort **if** for any $j \in [0\mathinner{..}i]$, $\mathsf{KV}(\mathsf{pk}_j, \Pi_{\mathsf{pk}_j}) \neq 1$; Sample $r \leftarrow \{0,1\}^\lambda$; $c = \mathsf{Enc}(\mathsf{pk}_i, \mathsf{w}; r)$; $\pi \leftarrow \mathsf{Sim}(\mathbf{R}_{\mathbf{L}'}, \xi_{\mathbf{R}_{\mathbf{L}'}}, \vec{\mathsf{crs}}_i, (\mathsf{x}, c, \mathsf{pk}_i), \vec{\mathsf{ts}}'_i)$; Output $\pi' := (c, \pi)$
- $(\mathsf{x}, \pi') \leftarrow \mathcal{A}^{\mathsf{O}(\mathsf{x})}(\vec{\mathsf{crs}}'_i, \vec{\mathsf{te}}'_i)$;
- Parse $\pi' := (c, \pi)$; extract witness $\mathsf{w} \leftarrow \mathsf{Dec}(c, \vec{\mathsf{te}}'_i)$;
- **if** $((\mathsf{x}, \pi') \notin Q) \wedge\ (\mathsf{V}(\mathbf{R}_{\mathbf{L}'}, \xi_{\mathbf{R}_{\mathbf{L}'}}, \vec{\mathsf{crs}}'_i, (\mathsf{x}, c, \mathsf{pk}_i), \pi) = 1) \wedge ((\mathsf{x}, \mathsf{w}) \notin \mathbf{R}_{\mathbf{L}})$ : **return** 1.
  where $Q$ is the set of statement-proof pairs generated by $\mathsf{O}(\mathsf{x})$. **fi**

**Lemma 3.** *If the cryptosystem* $\Psi_{\mathsf{Enc}}$ *used in above experiments be updatable IND-CPA and updatable key hiding, then for two experiments* $\mathsf{EXP}_2^{BB-SE}$ *and* $\mathsf{EXP}_1^{BB-SE}$*, we have* $\Pr[\mathsf{EXP}_2^{BB-SE}] \leq \Pr[\mathsf{EXP}_1^{BB-SE}] + \mathsf{negl}(\lambda)$ .

*Proof.* Due to updatable key hiding of $\Psi_{\mathsf{Enc}}$, $\mathsf{pk}_0 \approx_\lambda \mathsf{pk}_i$. The updatable IND-CPA property of $\Psi_{\mathsf{Enc}}$ implies that after a one-time honest key generation/updating, no polynomial-time algorithm (adversary) can distinguish an oracle that encrypts randomly chosen value $z$ with the public key $\mathsf{pk}_i$ and uses the simulator $\mathsf{Sim}$, from the case that it encrypts the true witness $\mathsf{w}$ with the $\mathsf{pk}_i$ and again uses the simulator $\mathsf{Sim}$, even if it has updated the public-key $\mathsf{pk}_i$ arbitrary times, i.e. $(i-1)$ times.

$\underline{\mathsf{EXP}_3^{BB-SE}}$(prover):

- *Setup:* The same as in experiment $\mathsf{EXP}_1^{BB-SE}$ and $\mathsf{EXP}_2^{BB-SE}$.

- *Define function* $\mathsf{O}(\mathsf{x})$: Abort **if** for any $j \in [0\,..\,i]$, $\mathsf{CV}(\vec{\mathsf{crs}}_j, \mathit{\Pi}_{\vec{\mathsf{crs}}_j}) \neq 1$; Abort **if** for any $j \in [0\,..\,i]$, $\mathsf{KV}(\mathsf{pk}_j, \mathit{\Pi}_{\mathsf{pk}_j}) \neq 1$; Sample $r \leftarrow \{0,1\}^\lambda$; $c = \mathsf{Enc}(\mathsf{pk}_i, \mathsf{w}; r)$; $\pi \leftarrow \mathsf{P}(\mathbf{R}_{\mathbf{L}'}, \xi_{\mathbf{R}_{\mathbf{L}'}}, \vec{\mathsf{crs}}_i, (\mathsf{x}, c, \mathsf{pk}_i), (\mathsf{w}, r))$; Output $\pi' := (c, \pi)$
- $(\mathsf{x}, \pi') \leftarrow \mathcal{A}^{\mathsf{O}(\mathsf{x})}(\vec{\mathsf{crs}}'_i, \vec{\mathsf{te}}'_i)$;
- Parse $\pi' := (c, \pi)$; extract witness $\mathsf{w} \leftarrow \mathsf{Dec}(c, \vec{\mathsf{te}}'_i)$;
- **if** $((\mathsf{x}, \pi') \not\in Q) \wedge (\mathsf{V}(\mathbf{R}_{\mathbf{L}'}, \xi_{\mathbf{R}_{\mathbf{L}'}}, \vec{\mathsf{crs}}'_i, (\mathsf{x}, c, \mathsf{pk}_i), \pi) = 1) \wedge ((\mathsf{x}, \mathsf{w}) \not\in \mathbf{R}_{\mathbf{L}})$ : **return** 1.

  where $Q$ is the set of statement-proof pairs generated by $\mathsf{O}(\mathsf{x})$. **fi**

**Lemma 4.** *If the NIZK argument is updatable simulation sound (in Def.4), and the encryption scheme* $\mathit{\Psi}_{\mathsf{Enc}}$ *is perfect updatable correct (in Def.14) then for two experiments* $\mathsf{EXP}_3^{BB-SE}$ *and* $\mathsf{EXP}_2^{BB-SE}$ *we have* $\Pr[\mathsf{EXP}_3^{BB-SE}] \leq \Pr[\mathsf{EXP}_2^{BB-SE}] + \mathsf{negl}(\lambda)$.

*Proof.* We note that if $(\mathsf{x}, \pi') \not\in Q$, then the tuple $(\mathsf{x}, c, \pi)$ (from $(\mathsf{x}, \pi')$) is a valid pair in the relation $\mathbf{R}_{\mathbf{L}'}$. The updatable simulation soundness property of the NIZK argument $\mathit{\Psi}_{\mathsf{NIZK}}$ impels the non-malleability of proofs, consequently we know that $(\mathsf{x}, \pi') \not\in Q$.

On the other hand, perfect updatable correctness of the cryptosystem $\mathit{\Psi}_{\mathsf{Enc}}$ implies that the decrypted witness $\mathsf{w}$ is unique for all valid ciphertexts, so due to the soundness of the NIZK argument $\mathit{\Psi}_{\mathsf{Enc}}$ the probability that some witness is valid for $\mathbf{L}'$ and $(\mathsf{x}, \mathsf{w}) \not\in \mathbf{R}_{\mathbf{L}}$ is $\mathsf{negl}(\lambda)$, namely $\Pr[\mathsf{EXP}_3^{BB-SE}] \leq 2^{-\lambda}$.

This completes the main proof. □

Note that to bypass the impossibility of achieving Sub-ZK and BB extractability in NIZK arguments, observed by Bellare et al. [BFS16], one-time honest key generation/updating on $\mathsf{pk}_i$ is a crucial requirement in the above theorem. Roughly speaking, if the prover participates in the generating/updating $\mathsf{pk}_i$ once, so due to the updatable key hiding and updatable IND-CPA of the cryptosystem $\mathit{\Psi}_{\mathsf{Enc}}$, even if adversary updates the keys arbitrary times still he/she cannot learn any information about the final secret key $\vec{\mathsf{te}}'_i := \sum_{j=0}^{i} \mathsf{sk}_j$ and consequently from the witness $\mathsf{w}$ used in generating $\pi' := (c, \pi)$.

***Building Updatable Black-Box Knowledge Sound NIZK Arguments with*** **Tiramisu.** The primary goal of Tiramisu is constructing BB-SE NIZK arguments in the updatable CRS model. However, due to some efficiency reasons, in practice one might need to build an Updatable Black-Box Knowledge Sound (U-BB-KS) NIZK argument. In such cases, starting from either an updatable sound NIZK or an U-nBB-KS NIZK (e.g. Groth et al.'s updatable zk-SNARK [GKM+18]), the same language $\mathbf{L}'$ defined in Tiramisu along with our constructed updatable public-key cryptosystem allows one to build an U-BB-KS NIZK argument. Namely, given an updatable cryptosystem $\mathit{\Psi}_{\mathsf{Enc}} := (\mathsf{KG}, \mathsf{KU}, \mathsf{KV}, \mathsf{Enc}, \mathsf{Dec})$ with updatable keys $(\mathsf{pk}_i, \mathsf{sk}_i)$, and an *updatable*

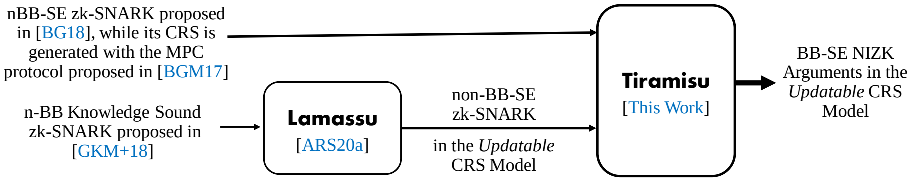
nBB-SE zk-SNARK proposed in [BG18], while its CRS is generated with the MPC protocol proposed in [BGM17]
n-BB Knowledge Sound zk-SNARK proposed in [GKM+18]
Lamassu
[ARS20a]
non-BB-SE zk-SNARK
in the Updatable CRS Model
Tiramisu
[This Work]
BB-SE NIZK Arguments in the Updatable CRS Model

Fig. 4: Instantiating the NIZK argument $\Psi_{\mathsf{NIZK}}$ in Tiramisu .

*sound* NIZK $\Psi_{\mathsf{NIZK}} := (\mathsf{K}_{\vec{\mathsf{crs}}}, \mathsf{CU}, \mathsf{CV}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ for language $\mathbf{L}$ with the corresponding **NP** relation $\mathbf{R_L}$, we define the language $\mathbf{L}'$ for a given random element $r \leftarrow\$ \mathbb{F}_p$, such that $((\mathsf{x}, c, \mathsf{pk}_i), (\mathsf{w}, r)) \in \mathbf{R_{L'}}$ iff:

$$(c = \mathsf{Enc}(\mathsf{pk}_i, \mathsf{w}; r)) \wedge ((\mathsf{x}, \mathsf{w}) \in \mathbf{R_L}).$$

**Corollary 1.** *If the input NIZK argument $\Psi_{\mathsf{NIZK}}$ for $\mathbf{R_L}$ guarantees updatable correctness, updatable soundness and updatable zero-knowledge, and the public-key cryptosystem $\Psi_{\mathsf{Enc}}$ satisfies updatable perfect correctness, updatable key hiding, and updatable IND-CPA, then the NIZK argument for language $\mathbf{L}'$ satisfies updatable correctness, updatable knowledge soundness and updatable zero-knowledge.*

The proof can be done similar to the proof of Theorem 4.

## 5 Instantiation of Tiramisu

To build an updatable BB-SE NIZK argument with Tiramisu, one requires two primitives. Namely, (1) a key updatable public-key cryptosystem $\Psi_{\mathsf{Enc}}$ that satisfies *perfect updatable correctness*, *updatable key hiding*, and *updatable IND-CPA*, and (2) a NIZK argument $\Psi_{\mathsf{NIZK}}$ with updatable CRS that guarantees *updatable simulation soundness* or *nBB simulation extractability*. Next, we instantiate $\Psi_{\mathsf{Enc}}$ and $\Psi_{\mathsf{NIZK}}$, and obtain two U-BB-SE NIZK arguments. To instantiate $\Psi_{\mathsf{Enc}}$, we use the proposed variation of the El-Gamal cryptosystem in Sec. 3. Whereas to instantiate the $\Psi_{\mathsf{NIZK}}$, one can either use an ad-hoc construction (e.g. [BG18,BKSV20,BPR20] when its CRS is generated with [BGM17], which will have a two-phase updating), or a construction lifted with Lamassu [ARS20a] (e.g. using [GKM$^+$18]). A graphical representation of two approaches that we instantiate the input NIZK argument of Tiramisu is shown in Fig. 4. In a nutshell, considering the above instantiations Tiramisu results in two BB-SE NIZK arguments with updatable parameters.

In BB-SE NIZK arguments built with Tiramisu, the parties have to update the shared parameters individually once and check the validity of the previous updates. This is basically the computational cost that the end-users need to pay to bypass the trust in the standard CRS model. As an important practical optimization, it can be shown that the prover can only update the CRS $\vec{\mathsf{crs}}'_i := (\vec{\mathsf{crs}}_i, \mathsf{pk}_i)$ partially, namely only $\mathsf{pk}_i$. Tab. 2 summarizes the efficiency

Table 2: An efficiency comparison of BB-SE NIZK arguments built with the C∅C∅ and Tiramisu. $n'$: Number of constraints (multiplication gates) used to encode language $\mathbf{L}'$, $|\mathsf{pk}|$: Size of the public key of $\Psi_{\mathsf{Enc}}$, $\lambda$: Security parameter, $E_i$: Exponentiation in $\mathbb{G}_i$, $P$ : Paring operation, $l'$: the size of statement in new language $\mathbf{L}'$, $\mathsf{w}$ : the witness for new relation $\mathbf{R}_{\mathbf{L}'}$.

| | **C∅C∅**<br>with [Gro16] | **Tiramisu**<br>(with [GKM$^+$18,ARS20b]) | **Tiramisu**<br>(with [BGM17,BG18]) |
|---|---|---|---|
| Trusted Setup? | Yes | No | No |
| CRS Size | $\approx 3n'\mathbb{G}_1 + n'\mathbb{G}_2$ | $\approx 30n'^2\mathbb{G}_1 + 9n'^2\mathbb{G}_2$ | $\approx 3n'\mathbb{G}_1 + n'\mathbb{G}_2$ |
| CRS Verifier | — | $\approx 78n'^2 P$ | $14n'P$ (batchable) |
| CRS Updater | — | $\approx 30n'^2 E_1 + 9n'^2 E_2$ | $\approx 6n'E_1 + n'E_2$ |
| Prover | $\approx 4n'E_1 + n'E_2$ | $\approx 4n'E_1 + n'E_2$ | $\approx 4n'E_1 + n'E_2$ |
| Proof Size | $o(\mathsf{w}) + 3\mathbb{G}_1 + 2\mathbb{G}_2 + \lambda$ | $o(\mathsf{w}) + 4\mathbb{G}_1 + 3\mathbb{G}_2$ | $o(\mathsf{w}) + 3\mathbb{G}_1 + 2\mathbb{G}_2$ |
| Verifier | $4P + l'E_1$ | $6P + l'E_1$ | $5P + l'E_1$ |

of two BB-SE NIZK arguments built with Tiramisu and compares them with a construction lifted by the C∅C∅ framework in the standard CRS model. We instantiate C∅C∅ with the state-of-the-art zk-SNARK [Gro16] and instantiate Tiramisu with 1) the lifted version of [GKM$^+$18] with Lamassu [ARS20a], and 2) the construction proposed in [BG18] or in [BPR20] when their CRS is sampled using the protocol proposed in [BGM17].

Both C∅C∅ and Tiramisu constructions result a linear proof in the witness size, but they keep the asymptomatic efficiency of other algorithms in the input NIZK. Consequently, instantiating Tiramisu with a more efficient nBB-SE NIZK argument will result in a more efficient BB-SE NIZK argument. Therefore, as also is shown in Tab. 2, suitable ad-hoc constructions result in more efficient U-BB-SE NIZK arguments. We found constructing more efficient nBB-SE zk-SNARKs as an interesting future research direction. Following, the impossibility result of Gentry and Wichs [GW11], it is undeniable that achieving BB extraction will result in non-succinct proof. Consequently, in all the schemes in Tab. 2, the proof size is dominated with the size of $c$ which is a ciphertext of IND-CPA cryptosystem and is $o(\mathsf{w})$.

***Using Updatable BB-SE NIZK Arguments in UC-Protocols.*** Following the known result that BB-SE NIZK arguments can realize the ideal NIZK-functionality $\mathcal{F}_{\mathsf{NIZK}}$ [GOS06,Gro06], the UC-protocols like Hawk [KMS$^+$16], Gyges [JKS16], and Ouroboros Crypsinous [KKKZ19], directly use the BB-SE NIZK arguments constructed by the C∅C∅ framework. Consequently, under *a trusted setup* phase, the deployed BB-SE NIZK argument securely composes with other primitives in the main protocol. But, in BB-SE NIZK arguments that are built with Tiramisu, the parties do not need to trust a third party, instead, they need to update the CRS elements and give a proof $\Pi_{\vec{\mathsf{crs}}}$ for the correctness of the initial key generation or updating. But, since the proof $\Pi_{\vec{\mathsf{crs}}}$ relies on a knowledge assumption [Dam91] and nBB extraction, therefore in the lifted NIZK arguments the setup phase can not achieve UC, as the nBB extraction is not allowed in the UC framework. Albeit once the CRS elements are generated and updated by both prover and verifier, rest of the protocol including proof generation and proof verification achieves UC-security. Technically speaking, in comparison with the realization of NIZK-functionality $\mathcal{F}_{\mathsf{NIZK}}$ in the $\mathcal{F}_{\vec{\mathsf{crs}}}$-model which is described in [Gro06], the U-BB-SE NIZK arguments constructed with Tiramisu can realize NIZK-functionality $\mathcal{F}_{\mathsf{NIZK}}$ in a relaxed model which uses non-black-box extractions. More details about such realizations will appear in the full version of the paper. A similar case where a trustless setup phase (called offline phase) uses nBB extractors, while the online phase achieves BB extraction and UC-security is achieved in the known SPDZ MPC protocol [DPSZ12]. However, their construction does not guarantee the updatability of parameters in the offline phase [DPSZ12]. Due to construction of current subversion/updatable NIZK arguments that rely on a knowledge assumption in the setup phase, this looks an avoidable fact that one has to take either 1) a UC-secure setup phase *while trusting* a third party, or 2) a non-UC secure setup phase but *without* a need for a trusted third party. In practice, usually, the public parameters are generated once and used for a long-time, therefore having a non-UC secure setup phase might be a more desired option than having a UC-secure setup but with the need for a trusted party for a *long* time.

## 6 Conclusion

We proposed Tiramisu that allows one to build BB-SE NIZK arguments in the *updatable* CRS model. BB-SE NIZK arguments built with Tiramisu allow the parties (both prover and verifier) to bypass the trust in a third party by one-time participation in the CRS generation/updating. We instantiated Tiramisu in two ways and presented NIZK arguments that achieve U-BB-SE without the need for a trusted third party. We observed that some instantiations results in more efficient U-BB-SE NIZK arguments. Meanwhile, as a building block for Tiramisu , we defined the syntax of public-key cryptosystems with updatable keys and presented a variation of the El-Gamal cryptosystem [ElG84] as an efficient construction.

In practice, by deploying the constructed U-BB-SE NIZK arguments in UC-protocols, such as Hawk [KMS+16], Gyges [JKS16], Ouroboros Crypsinous [KKKZ19], the end-users can bypass the trust in the setup phase and achieve UC security in the proof generation and proof verification. The extra cost that end-users need to pay is a one-time updating the parameters plus checking the others' updates on parameters. Tiramisu comes with efficient algorithms $\mathsf{CU}$ and $\mathsf{CV}$ for parameter updating and verification, respectively. Specifically about UC-secure privacy-preserving smart contracts systems like Hawk [KMS+16], by deploying an U-BB-SE NIZK argument in a two-party smart contract, both parties can avoid trusting a third party if both individually update the public parameters using $\mathsf{CU}$ and also check the other party's update with $\mathsf{CV}$.

We believe our proposed technique to build U-BB-SE NIZK arguments along with the proposed updatable public-key cryptosystem can be of independent interest, particularly in constructing other cryptographic protocols in the updatable CRS model.

**Acknowledgement.** This work was supported in part by Cyber Security Research Flanders with reference number VR20192203, by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001120C0085, and by the Research Council KU Leuven C1 on Security and Privacy for Cyber-Physical Systems and the Internet of Things with contract number C16/15/058.

## References

AB19. Shahla Atapoor and Karim Baghery. Simulation extractability in Groth's zk-SNARK. In Cristina Perez-Sola, Guillermo Navarro-Arribas, Alex Biryukov, and Joaquin Garcia-Alfaro, editors, *Data Privacy Management, Cryptocurrencies and Blockchain Technology - ESORICS 2019 International Workshops, DPM 2019 and CBT 2019, Luxembourg, September 26-27, 2019, Proceedings*, volume 11737 of *Lecture Notes in Computer Science*, pages 336–354. Springer, 2019.

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.

ABL$^+$19. Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, Janno Siim, and Michal Zajac. UC-secure CRS generation for SNARKs. In Johannes Buchmann, Abderrahmane Nitaj, and Tajje eddine Rachidi, editors, *AFRICACRYPT 19*, volume 11627 of *LNCS*, pages 99–117. Springer, Heidelberg, July 2019.

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.

AHI11. Benny Applebaum, Danny Harnik, and Yuval Ishai. Semantic security under related-key attacks and applications. In Bernard Chazelle, editor, *ICS 2011*, pages 45–60. Tsinghua University Press, January 2011.

ARS20a. Behzad Abdolmaleki, Sebastian Ramacher, and Daniel Slamanig. Lift-and-Shift: Obtaining simulation extractable subversion and updatable SNARKs generically. In *ACM SIGSAC Conference on Computer and Communications Security, CCS 2020 (accepted to appear), avalable on: http://eprint.iacr.org/2020/062*, 2020.

ARS20b. Behzad Abdolmaleki, Sebastian Ramacher, and Daniel Slamanig. Lift-and-shift: Obtaining simulation extractable subversion and updatable SNARKs generically. Cryptology ePrint Archive, Report 2020/062, (Accessed 20 May 2020), 2020. http://eprint.iacr.org/2020/062.

ARS20c. Behzad Abdolmaleki, Sebastian Ramacher, and Daniel Slamanig. SoK: Lifting transformations for simulation extractable subversion and updatable SNARKs. In *3rd ZKProof Workshop, Home Edition, 2020, avalable on: https://docs.zkproof.org/pages/standards/accepted-workshop3/sok-lifting_transformations_se_snarks.pdf*, 2020.

Bag19a. Karim Baghery. On the efficiency of privacy-preserving smart contract systems. In Johannes Buchmann, Abderrahmane Nitaj, and Tajje eddine Rachidi, editors, *AFRICACRYPT 19*, volume 11627 of *LNCS*, pages 118–136. Springer, Heidelberg, July 2019.

Bag19b. 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.

Bag20. Karim Baghery. Subversion-resistant commitment schemes: Definitions and constructions. In Konstantinos Markantonakis and Marinella Petrocchi, editors, *Security and Trust Management - 16th International Workshop, STM 2020, Guildford, UK, September 17-18, 2020, Proceedings*, volume 12386 of *Lecture Notes in Computer Science*, pages 106–122. Springer, 2020.

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.

BCG$^+$15. Eli Ben-Sasson, Alessandro Chiesa, Matthew Green, Eran Tromer, and Madars Virza. Secure sampling of public parameters for succinct zero knowledge proofs. In *2015 IEEE Symposium on Security and Privacy*, pages 287–304. IEEE Computer Society Press, May 2015.

BCPR14. Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In David B. Shmoys, editor, *46th ACM STOC*, pages 505–514. ACM Press, May / June 2014.

BCTV13. Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive arguments for a von neumann architecture. Cryptology ePrint Archive, Report 2013/879, 2013. `http://eprint.iacr.org/2013/879`.

BFM88. Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 103–112. ACM, 1988.

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.

BG90. Mihir Bellare and Shafi Goldwasser. New paradigms for digital signatures and message authentication based on non-interative zero knowledge proofs. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 194–211. Springer, Heidelberg, August 1990.

BG18. Sean Bowe and Ariel Gabizon. Making groth's zk-SNARK simulation extractable in the random oracle model. Cryptology ePrint Archive, Report 2018/187, 2018. `https://eprint.iacr.org/2018/187`.

BGG19. Sean Bowe, Ariel Gabizon, and Matthew D. Green. A multi-party protocol for constructing the public parameters of the pinocchio zk-SNARK. In Aviv Zohar, Ittay Eyal, Vanessa Teague, Jeremy Clark, Andrea Bracciali, Federico Pintore, and Massimiliano Sala, editors, *FC 2018 Workshops*, volume 10958 of *LNCS*, pages 64–77. Springer, Heidelberg, March 2019.

BGM17. Sean Bowe, Ariel Gabizon, and Ian Miers. Scalable multi-party computation for zk-SNARK parameters in the random beacon model. Cryptology ePrint Archive, Report 2017/1050, 2017. `http://eprint.iacr.org/2017/1050`.

BKSV20. Karim Baghery, Markulf Kohlweiss, Janno Siim, and Mikhail Volkhov. Another look at extraction and randomization of Groth's zk-SNARK. Cryptology ePrint Archive, Report 2020/811, 2020. `https://eprint.iacr.org/2020/811`.

BPR20. Karim Baghery, Zaira Pindado, and Carla Ràfols. Simulation extractable versions of Groth's zk-SNARK revisited. Cryptology ePrint Archive, Report 2020/1306, 2020. `https://eprint.iacr.org/2020/1306`.

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

CFQ19. Matteo Campanelli, Dario Fiore, and Anaïs Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. Cryptology ePrint Archive, Report 2019/142, 2019. `https://eprint.iacr.org/2019/142`.

CHK03. Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 255–271. Springer, Heidelberg, May 2003.

Dam91. Ivan Damgård. Towards Practical Public Key Systems Secure against Chosen Ciphertext Attacks. In Joan Feigenbaum, editor, *CRYPTO 1991*, volume 576 of *LNCS*, pages 445–456, Santa Barbara, California, USA, August 11–15, 1991. Springer, Heidelberg, 1992.

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.

DDO$^+$01. Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, Heidelberg, August 2001.

DGP$^+$19. Vanesa Daza, Alonso González, Zaira Pindado, Carla Ràfols, and Javier Silva. Shorter quadratic QA-NIZK proofs. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part I*, volume 11442 of *LNCS*, pages 314–343. Springer, Heidelberg, April 2019.

DPSZ12. Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 643–662. Springer, Heidelberg, August 2012.

DS16. David Derler and Daniel Slamanig. Key-homomorphic signatures and applications to multiparty signatures. Cryptology ePrint Archive, Report 2016/792, 2016. `http://eprint.iacr.org/2016/792`.

ElG84. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, Heidelberg, August 1984.

FMMO19. Prastudy Fauzi, Sarah Meiklejohn, Rebekah Mercer, and Claudio Orlandi. Quisquis: A new design for anonymous cryptocurrencies. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 649–678. Springer, Heidelberg, December 2019.

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.

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.

GM17. Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 581–612. Springer, Heidelberg, August 2017.

GMR89. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

GOS06. Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 339–358. Springer, Heidelberg, May / June 2006.

Gro06. Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, Heidelberg, December 2006.

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.

Gro16. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.

GS08. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 415–432. Springer, 2008.

GW11. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011.

GWC19. Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. https://eprint.iacr.org/2019/953.

JKS16. Ari Juels, Ahmed E. Kosba, and Elaine Shi. The ring of Gyges: Investigating the future of criminal smart contracts. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 283–295. ACM Press, October 2016.

KKKZ19. Thomas Kerber, Aggelos Kiayias, Markulf Kohlweiss, and Vassilis Zikas. Ouroboros crypsinous: Privacy-preserving proof-of-stake. In *2019 IEEE Symposium on Security and Privacy*, pages 157–174. IEEE Computer Society Press, May 2019.

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.

KZM$^{+}$15. Ahmed E. Kosba, Zhichao Zhao, Andrew Miller, Yi Qian, T.-H. Hubert Chan, Charalampos Papamanthou, Rafael Pass, Abhi Shelat, and Elaine

Shi. C∅C∅: A Framework for Building Composable Zero-Knowledge Proofs. Technical Report 2015/1093, IACR, November 10, 2015. http://eprint.iacr.org/2015/1093, last accessed version from 9 Apr 2017.

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, Taormina, Italy, March 18–21, 2012. Springer, Heidelberg.

Lip19. Helger Lipmaa. Simulation-extractable SNARKs revisited. Cryptology ePrint Archive, Report 2019/612, 2019. http://eprint.iacr.org/2019/612.

MBKM19. Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, November 2019.

Nao03. Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 96–109. Springer, Heidelberg, August 2003.

PHGR13. Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013.

TW14. Stefano Tessaro and David A. Wilson. Bounded-collusion identity-based encryption from semantically-secure public-key encryption: Generic constructions with short ciphertexts. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 257–274. Springer, Heidelberg, March 2014.

Woo14. Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32, 2014.

## A C∅C∅: a Framework for Constructing BB-SE NIZK Arguments in the CRS Model

In 2015, Kosba et al. [KZM+15] presented a framework, called C∅C∅, with several constructions which allows to build BB-SE NIZK arguments. Their most powerful construction gets a sound NIZK and lifts to a NIZK argument that satisfies BB-SE (defined in Def. 22), which is shown to be a sufficient requirement for NIZK arguments to achieve UC-security [Gro06]. Here we review their most powerful construction.

**Construction.** Given a sound NIZK, to achieve a UC-secure NIZK, the C∅C∅ framework applies several changes in all setup, proof generation and verification procedures of the input NIZK. Initially, the framework defines a new language $\mathbf{L}'$ based on the language $\mathbf{L}$ in underlying NIZK and some new primitives that are needed for the transformation. Let $(\mathsf{KG}_e, \mathsf{Enc}, \mathsf{Dec})$ be a set of algorithms for a semantically secure encryption scheme, $(\mathsf{KG}_s, \mathsf{Sig}_s, \mathsf{Vfy}_s)$ be a one-time signature scheme and $(\mathsf{Com}, \mathsf{Vfy})$ be a perfectly binding commitment scheme. Given a language $\mathbf{L}$ with the corresponding $\mathbf{NP}$ relation $\mathbf{R_L}$, define a new language $\mathbf{L}'$ such that $((\mathsf{x}, c, \mu, \mathsf{pk}_s, \mathsf{pk}_e, \rho), (r, r_0, \mathsf{w}, s_0)) \in \mathbf{R}_{\mathbf{L}'}$ iff:

$$(c = \mathsf{Enc}(\mathsf{pk}_e, \mathsf{w}; r)) \wedge ((\mathsf{x}, \mathsf{w}) \in \mathbf{R_L} \vee (\mu = f_{s_0}(\mathsf{pk}_s) \wedge \rho = \mathsf{Com}(s_0; r_0))) \,,$$

where $\{f_s : \{0,1\}^* \to \{0,1\}^\lambda\}_{s \in \{0,1\}^\lambda}$ is a pseudo-random function family. Now, a sound NIZK argument system $\Psi_{\mathsf{NIZK}}$ for $\mathcal{R}$ constructed from PPT algorithms $(\mathsf{KG}, \mathsf{P}, \mathsf{V}, \mathsf{Sim}, \mathsf{Ext})$ can be lifted to a BB-SE NIZK $\Psi'_{\mathsf{NIZK}}$ with PPT algorithms $(\mathsf{KG}', \mathsf{P}', \mathsf{V}', \mathsf{Sim}', \mathsf{Ext}')$ as follows.

**CRS and trapdoor generation** $\mathsf{KG}'(\mathbf{R_L}, \xi_{\mathbf{R_L}})$**:** Sample $(\vec{\mathsf{crs}} \parallel \vec{\mathsf{ts}}) \leftarrow \mathsf{KG}(\mathbf{R_{L'}}, \xi_{\mathbf{R_{L'}}})$; $(\mathsf{pk}_e, \mathsf{sk}_e) \leftarrow \mathsf{KG}_e(1^\lambda)$; $s_0, r_0 \leftarrow\$ \{0,1\}^\lambda$; $\rho := \mathsf{Com}(s_0; r_0)$; and output $(\vec{\mathsf{crs}}' \parallel \vec{\mathsf{ts}}' \parallel \vec{\mathsf{te}}') := ((\vec{\mathsf{crs}}, \mathsf{pk}_e, \rho) \parallel (s_0, r_0) \parallel \mathsf{sk}_e)$.

**Prover** $\mathsf{P}'(\mathbf{R_L}, \xi_{\mathbf{R_L}}, \vec{\mathsf{crs}}, \mathsf{x}, \mathsf{w})$**:** Parse $\vec{\mathsf{crs}}' := (\vec{\mathsf{crs}}, \mathsf{pk}_e, \rho)$; Abort if $(\mathsf{x}, \mathsf{w}) \notin \mathbf{R_L}$; $(\mathsf{pk}_s, \mathsf{sk}_s) \leftarrow \mathsf{KG}_s(1^\lambda)$; sample $z_0, z_1, z_2, r_1 \leftarrow\$ \{0,1\}^\lambda$; compute $c = \mathsf{Enc}(\mathsf{pk}_e, \mathsf{w}; r_1)$; generate $\pi \leftarrow \mathsf{P}(\mathbf{R_{L'}}, \xi_{\mathbf{R_{L'}}}, \vec{\mathsf{crs}}, (\mathsf{x}, c, z_0, \mathsf{pk}_s, \mathsf{pk}_e, \rho), (r_1, z_1, w, z_2))$; sign $\sigma \leftarrow \mathsf{Sig}_s(\mathsf{sk}_s, (\mathsf{x}, c, z_0, \pi))$; and output $\pi' := (c, z_0, \pi, \mathsf{pk}_s, \sigma)$.

**Verifier** $\mathsf{V}'(\mathbf{R_L}, \xi_{\mathbf{R_L}}, \vec{\mathsf{crs}}', \mathsf{x}, \pi')$**:** Parse $\vec{\mathsf{crs}}' := (\vec{\mathsf{crs}}, \mathsf{pk}_e, \rho)$ and $\pi' := (c, \mu, \pi, \mathsf{pk}_s, \sigma)$; Abort if $\mathsf{Vfy}_s(\mathsf{pk}_s, (\mathsf{x}, c, \mu, \pi), \sigma) = 0$; call $\mathsf{V}(\mathbf{R_{L'}}, \xi_{\mathbf{R_{L'}}}, \vec{\mathsf{crs}}, (\mathsf{x}, c, \mu, \mathsf{pk}_s, \mathsf{pk}_e, \rho), \pi)$ and abort if it outputs 0.

**Simulator** $\mathsf{Sim}'(\mathbf{R_L}, \xi_{\mathbf{R_L}}, \vec{\mathsf{crs}}', \vec{\mathsf{ts}}', \mathsf{x})$**:** Parse $\vec{\mathsf{crs}}' := (\vec{\mathsf{crs}}, \mathsf{pk}_e, \rho)$ and $\vec{\mathsf{ts}}' := (s_0, r_0)$; $(\mathsf{pk}_s, \mathsf{sk}_s) \leftarrow \mathsf{KG}_s(1^\lambda)$; set $\mu = f_{s_0}(\mathsf{pk}_s)$; sample $z_3, r_1 \leftarrow\$ \{0,1\}^\lambda$; compute $c = \mathsf{Enc}(\mathsf{pk}_e, z_3; r_1)$; generate $\pi \leftarrow \mathsf{P}(\mathbf{R_{L'}}, \xi_{\mathbf{R_{L'}}}, \vec{\mathsf{crs}}, (\mathsf{x}, c, \mu, \mathsf{pk}_s, \mathsf{pk}_e, \rho), (r_1, r_0, z_3, s_0))$; sign $\sigma \leftarrow \mathsf{Sig}_s(\mathsf{sk}_s, (\mathsf{x}, c, \mu, \pi))$; and output $\pi' := (c, \mu, \pi, \mathsf{pk}_s, \sigma)$.

**Extractor** $\mathsf{Ext}'(\mathbf{R_L}, \xi_{\mathbf{R_L}}, \vec{\mathsf{crs}}', \vec{\mathsf{te}}', \mathsf{x}, \pi')$**:** Parse $\pi' := (c, \mu, \pi, \mathsf{pk}_s, \sigma)$, $\vec{\mathsf{te}}' := \mathsf{sk}_e$; extract $\mathsf{w} \leftarrow \mathsf{Dec}(\mathsf{sk}_e, c)$; output $\mathsf{w}$.

# B Requirements of NIZKs in the CRS Model

Next, we provide security requirement of standard and subversion-resistant NIZK arguments in the CRS model [Gro16,BFS16,ABLZ17,GM17,KZM$^+$15]. A *zk-SNARK* $\Psi_{\mathsf{NIZK}}$ in the CRS model for $\mathcal{R}$ consists of tuple of PPT algorithms $(\mathsf{K}_{\vec{\mathsf{crs}}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim}, \mathsf{Ext})$, that is expected to satisfy Completeness, ZK and Knowledge soundness defined as bellow,

**Definition 17 (Perfect Completeness [Gro16]).** *A non-interactive argument $\Psi_{\mathsf{NIZK}}$ is* perfectly complete for $\mathcal{R}$*, if for all $\lambda$, all $(\mathbf{R}, \xi_\mathbf{R}) \in \mathrm{im}(\mathcal{R}(1^\lambda))$, and $(\mathsf{x}, \mathsf{w}) \in \mathbf{R}$,*

$$\Pr[\vec{\mathsf{crs}} \leftarrow \mathsf{KG}(\mathbf{R}, \xi_\mathbf{R}) : \mathsf{V}(\mathbf{R}, \xi_\mathbf{R}, \vec{\mathsf{crs}}_\mathsf{V}, \mathsf{x}, \mathsf{P}(\mathbf{R}, \xi_\mathbf{R}, \vec{\mathsf{crs}}_\mathsf{P}, \mathsf{x}, \mathsf{w})) = 1] = 1 \enspace .$$

**Definition 18 (Statistically Zero-Knowledge [Gro16]).** *A non-interactive argument $\Psi_{\mathsf{NIZK}}$ is* statistically ZK for $\mathcal{R}$*, if for all $\lambda$, all $(\mathbf{R}, \xi_\mathbf{R}) \in \mathrm{im}(\mathcal{R}(1^\lambda))$, and for all NUPPT $\mathcal{A}$, $\varepsilon_0^{unb} \approx_\lambda \varepsilon_1^{unb}$, where*

$$\varepsilon_b = \Pr[(\vec{\mathsf{crs}} \parallel \vec{\mathsf{ts}}) \leftarrow \mathsf{KG}(\mathbf{R}, \xi_\mathbf{R}) : \mathcal{A}^{\mathsf{O}_b(\cdot,\cdot)}(\mathbf{R}, \xi_\mathbf{R}, \vec{\mathsf{crs}}) = 1] \enspace .$$

*Here, the oracle $\mathsf{O}_0(\mathsf{x}, \mathsf{w})$ returns $\perp$ (reject) if $(\mathsf{x}, \mathsf{w}) \notin \mathbf{R}$, and otherwise it returns $\mathsf{P}(\mathbf{R}, \xi_\mathbf{R}, \vec{\mathsf{crs}}_\mathsf{P}, \mathsf{x}, \mathsf{w})$. Similarly, $\mathsf{O}_1(\mathsf{x}, \mathsf{w})$ returns $\perp$ (reject) if $(\mathsf{x}, \mathsf{w}) \notin \mathbf{R}$, and otherwise it returns $\mathsf{Sim}(\mathbf{R}, \xi_\mathbf{R}, \vec{\mathsf{crs}}, \mathsf{x}, \vec{\mathsf{ts}})$. $\Psi_{\mathsf{NIZK}}$ is* perfect ZK for $\mathcal{R}$ *if one requires that $\varepsilon_0 = \varepsilon_1$.*

Intuitively, a non-interactive argument $\Psi_{\mathsf{NIZK}}$ is zero-knowledge if it does not leak extra information besides the truth of the statement.

**Definition 19 (Computational Knowledge-Soundness [Gro16]).** *A non-interactive argument $\Psi_{\mathsf{NIZK}}$ is computationally (adaptively)* knowledge-sound for *$\mathcal{R}$, if for every NUPPT $\mathcal{A}$, there exists a NUPPT extractor $\mathsf{Ext}_{\mathcal{A}}$, s.t. for all $\lambda$,*

$$\Pr\left[\begin{array}{l}(\mathbf{R}, \xi_{\mathbf{R}}) \leftarrow \mathcal{R}(1^\lambda), (\vec{\mathsf{crs}} \,\|\, \vec{\mathsf{ts}}) \leftarrow \mathsf{KG}(\mathbf{R}, \xi_{\mathbf{R}}), \\ r \leftarrow_r \mathsf{RND}(\mathcal{A}), ((\mathsf{x}, \pi) \,\|\, \mathsf{w}) \leftarrow (\mathcal{A} \,\|\, \mathsf{Ext}_{\mathcal{A}})(\mathbf{R}, \xi_{\mathbf{R}}, \vec{\mathsf{crs}}; r) : \\ (\mathsf{x}, \mathsf{w}) \not\in \mathbf{R} \wedge \mathsf{V}(\mathbf{R}, \xi_{\mathbf{R}}, \vec{\mathsf{crs}}_{\mathsf{V}}, \mathsf{x}, \pi) = 1\end{array}\right] \approx_\lambda 0 \enspace .$$

Here, $\xi_{\mathbf{R}}$ can be seen as a common auxiliary input to $\mathcal{A}$ and $\mathsf{Ext}_{\mathcal{A}}$ that is generated by using a benign [BCPR14] relation generator; A knowledge-sound argument system is called an *argument of knowledge.*

Besides the mentioned properties defined in Def. 17-19, a zk-SNARK has *succinctness* property, meaning that the proof size is $\mathsf{poly}(\lambda)$ and the verifier's computation is $\mathsf{poly}(\lambda)$ and the size of the instance.

Next, we recall some stronger notions of NIZK arguments that usually are needed in cases that one requires to achieve stronger security guarantees in the NIZK argument.

**Definition 20 (Simulation Soundness [Gro06]).** *A non-interactive argument $\Psi_{\mathsf{NIZK}}$ is* simulation sound for *$\mathcal{R}$ if for all NUPPT $\mathcal{A}$, and all $\lambda$,*

$$\Pr\left[\begin{array}{l}(\mathbf{R}, \xi_{\mathbf{R}}) \leftarrow \mathcal{R}(1^\lambda), (\vec{\mathsf{crs}} \,\|\, \vec{\mathsf{ts}}) \leftarrow \mathsf{KG}(\mathbf{R}, \xi_{\mathbf{R}}), (\mathsf{x}, \pi) \leftarrow \mathcal{A}^{\mathsf{O}(.)}(\mathbf{R}, \xi_{\mathbf{R}}, \vec{\mathsf{crs}}) : \\ (\mathsf{x}, \pi) \not\in Q \wedge \mathsf{x} \not\in \mathbf{L} \wedge \mathsf{V}(\mathbf{R}, \xi_{\mathbf{R}}, \vec{\mathsf{crs}}_{\mathsf{V}}, \mathsf{x}, \pi) = 1\end{array}\right] \approx_\lambda 0 \enspace .$$

Here, $Q$ is the set of simulated statement-proof pairs generated by adversary's queries to $\mathsf{O}$, that returns simulated proofs.

**Definition 21 (Non-Black-Box Simulation Extractability [GM17]).** *A non-interactive argument $\Psi_{\mathsf{NIZK}}$ is* non-black-box simulation-extractable for *$\mathcal{R}$, if for any NUPPT $\mathcal{A}$, there exists a NUPPT extractor $\mathsf{Ext}_{\mathcal{A}}$ s.t. for all $\lambda$,*

$$\Pr\left[\begin{array}{l}(\mathbf{R}, \xi_{\mathbf{R}}) \leftarrow \mathcal{R}(1^\lambda), (\vec{\mathsf{crs}} \,\|\, \vec{\mathsf{ts}}) \leftarrow \mathsf{KG}(\mathbf{R}, \xi_{\mathbf{R}}), \\ r \leftarrow_r \mathsf{RND}(\mathcal{A}), ((\mathsf{x}, \pi) \,\|\, \mathsf{w}) \leftarrow (\mathcal{A}^{\mathsf{O}(.)} \,\|\, \mathsf{Ext}_{\mathcal{A}})(\mathbf{R}, \xi_{\mathbf{R}}, \vec{\mathsf{crs}}; r) : \\ (\mathsf{x}, \pi) \not\in Q \wedge (\mathsf{x}, \mathsf{w}) \not\in \mathbf{R} \wedge \mathsf{V}(\mathbf{R}, \xi_{\mathbf{R}}, \vec{\mathsf{crs}}_{\mathsf{V}}, \mathsf{x}, \pi) = 1\end{array}\right] \approx_\lambda 0 \enspace .$$

Here, $Q$ is the set of simulated statement-proof pairs generated by adversary's queries to $\mathsf{O}$ that returns simulated proofs.

It is worth to mention that *non-black-box simulation extractability* implies *knowledge soundness* (given in Def. 19), as the earlier is a strong notion of the later which additionally the adversary is allowed to send query to the proof simulation oracle. Similarly, one can observe that *nBB simulation extractability* implies *simulation soundness* (given in Def. 20) [Gro06].

**Definition 22 (Black-Box Simulation Extractability [KZM+15]).** *A non-interactive argument $\Psi_{\mathsf{NIZK}}$ is* black-box simulation-extractable for $\mathcal{R}$ *if there exists a black-box extractor* $\mathsf{Ext}$ *that for all NUPPT $\mathcal{A}$, and all $\lambda$,*

$$\Pr\left[\begin{array}{l}(\mathbf{R},\xi_{\mathbf{R}}) \leftarrow \mathcal{R}(1^\lambda), (\vec{\mathsf{crs}} \parallel \vec{\mathsf{ts}} \parallel \vec{\mathsf{te}}) \leftarrow \mathsf{KG}(\mathbf{R},\xi_{\mathbf{R}}),\\ (\mathsf{x},\pi) \leftarrow \mathcal{A}^{\mathsf{O}(\cdot)}(\mathbf{R},\xi_{\mathbf{R}},\vec{\mathsf{crs}}), \mathsf{w} \leftarrow \mathsf{Ext}(\mathbf{R},\xi_{\mathbf{R}},\vec{\mathsf{crs}},\vec{\mathsf{te}},\mathsf{x},\pi) :\\ (\mathsf{x},\pi) \notin Q \wedge (\mathsf{x},\mathsf{w}) \notin \mathbf{R} \wedge \mathsf{V}(\mathbf{R},\xi_{\mathbf{R}},\vec{\mathsf{crs}}_{\mathsf{V}},\mathsf{x},\pi) = 1\end{array}\right] \approx_\lambda 0 \enspace .$$

Similarly, $Q$ is the set of simulated statement-proof pairs, and $\vec{\mathsf{te}}$ is the extraction trapdoor. A keynote about Def. 22 is that the extraction procedure is BB and unlike the nBB case, the extractor $\mathsf{Ext}$ works for all adversaries.

A subversion-resistant *zk-SNARK* $\Psi_{\mathsf{NIZK}}$ in the CRS model for $\mathcal{R}$ consists of tuple of PPT algorithms $(\mathsf{K}_{\vec{\mathsf{crs}}}, \mathsf{CV}, \mathsf{P}, \mathsf{V}, \mathsf{Sim}, \mathsf{Ext})$, that beyond Completeness, ZK and Knowledge soundness it is expected to achieve Subversion ZK which is defined as follows,

**Definition 23 (Statistically Subversion Zero-Knowledge [ABLZ17]).** *A non-interactive argument $\Psi$ is* statistically subversion ZK for $\mathcal{R}$*, if for any NUPPT subvertor* $\mathsf{Sub}$ *there exists a NUPPT extractor* $\mathsf{Ext}_{\mathsf{Sub}}$*, such that for all $\lambda$, all $(\mathbf{R},\xi) \in \mathrm{im}(\mathcal{R}(1^\lambda))$, and for all NUPPT $\mathcal{A}$, $\varepsilon_0 \approx_\lambda \varepsilon_1$, where*

$$\Pr\left[\begin{array}{l}r \leftarrow_r \mathsf{RND}(\mathsf{Sub}), (\vec{\mathsf{crs}}, \xi_{\mathsf{Sub}} \parallel \vec{\mathsf{ts}}) \leftarrow (\mathsf{Sub} \parallel \mathsf{Ext}_{\mathsf{Sub}})(\mathbf{R},\xi;r) :\\ \mathsf{CV}(\mathbf{R},\xi,\vec{\mathsf{crs}}) = 1 \wedge \mathcal{A}^{\mathsf{O}_b(\cdot,\cdot)}(\mathbf{R},\xi,\vec{\mathsf{crs}},\vec{\mathsf{ts}},\xi_{\mathsf{Sub}}) = 1\end{array}\right] \enspace .$$

*Here, $\xi_{\mathsf{Sub}}$ is auxiliary information generated by subvertor* $\mathsf{Sub}$*, and the oracle $\mathsf{O}_0(\mathsf{x},\mathsf{w})$ returns $\perp$ (reject) if $(\mathsf{x},\mathsf{w}) \notin \mathbf{R}$, and otherwise it returns* $\mathsf{P}(\mathbf{R},\xi,\vec{\mathsf{crs}}_{\mathsf{P}},\mathsf{x},\mathsf{w})$*. Similarly, $\mathsf{O}_1(\mathsf{x},\mathsf{w})$ returns $\perp$ (reject) if $(\mathsf{x},\mathsf{w}) \notin \mathbf{R}$, and otherwise it returns* $\mathsf{Sim}(\mathbf{R},\xi,\vec{\mathsf{crs}},\vec{\mathsf{ts}},\mathsf{x})$*. $\Psi$ is* perfectly subversion ZK for $\mathcal{R}$ *if one requires that $\varepsilon_0 = \varepsilon_1$.*