Lightweight Zero-Knowledge Proofs for Crypto-Computing Protocols

Sven Laur[1] and Bingsheng Zhang[2,3]

[1] University of Tartu, Estonia
swen@math.ut.ee
[2] State University of New York at Buffalo
[3] National and Kapodistrian University of Athens, Greece
bzhang@di.uoa.gr

Abstract Crypto-computing is a set of well-known techniques for computing with encrypted data. The security of the corresponding protocols are usually proven in the semi-honest model. In this work, we propose a new class of zero-knowledge proofs, which are tailored for crypto-computing protocols. First, these proofs directly employ properties of the underlying crypto systems and thus many facts have more concise proofs compared to generic solutions. Second, we show how to achieve universal composability in the trusted set-up model where all zero-knowledge proofs share the same system-wide parameters. Third, we derive a new protocol for multiplicative relations and show how to combine it with several crypto-computing frameworks.

Keywords. Universal composability, conditional disclosure of secrets, zero-knowledge proof, homomorphic encryption scheme, crypto-computing, multi-party computation.

1 Introduction

There are two basic approaches for crypto-computing: garbled circuit evaluation and protocols for computing with ciphertexts. Garbled circuit evaluation is Turing complete [Yao82,BHR12], while the computational expressibility of ciphertext manipulation depends on the underlying cryptosystem [SYY99,IP07,Gen09]. In this work, we consider only the ciphertext manipulation protocols. These protocols often rely on the specific properties of the underlying plaintexts, e.g., they assume that the encrypted inputs are bits. If these conditions are not satisfied, these protocols often break down and even the privacy of inputs is not guaranteed [AIR01,LL07]. Hence, these protocols are often complemented with zero-knowledge proofs to guarantee security.

Conditional disclosure of secrets (CDS) can be used as a lightweight alternative to zero knowledge proofs [GIKM98,AIR01,LL07]. In a nutshell, CDS protocols are used to release secrets only if received ciphertexts satisfy a desired relation. If we use these secrets to encrypt replies in the original protocol, these replies become unreadable when the ciphertexts are malformed. The resulting

protocol is extremely lightweight, as the transformation adds only few extra ciphertexts and some additional crypto-computing steps. On the flip side, CDS transformation ensures only input-privacy.

In this work, we extend this approach to full-fledged zero-knowledge proofs by adding a few short messages. As relatively efficient CDS protocols exist for proving **NP**/**poly** relations between plaintexts [AIR01,LL07], our method can be used to get zero-knowledge proofs for any **NP** language. It is a new and interesting paradigm for constructing zero-knowledge proofs, as resulting CdsZk protocols do not follow standard sigma structure.

In particular, note that parties do not have to agree before the protocol execution whether they aim for input-privacy, honest verifier or full-fledged zero-knowledge. This can be decided dynamically during the protocol with no overhead. For many zero-knowledge techniques, where the full-fledged zero-knowledge is achieved by adding extra messages into the beginning of the protocol, such flexibility is unachievable without additional overhead.

Properties of our zero-knowledge protocols are largely determined by the underlying commitment scheme. Essentially, we must choose between perfect simulation and statistical soundness. For perfect simulatability, the commitment scheme must be equivocal and thus security guarantees hold only against computationally bounded provers. Statistical binding assures unconditional soundness but it also prevents statistical simulatability. In both cases, usage of trusted setup together with dual mode commitments assures universal composability even if the setup is shared between all protocols.

As CDS and CdsZk protocols are mostly applied for protecting crypto-computing protocols against malicious adversaries, we show how the trusted setup can be implemented in the standard or in the common reference string model without losing security guarantees, see Section 6.

As the second major contribution, we describe a CDS protocol for a multiplicative relation with the cost of two additional ciphertexts. This is a major advancement, as all previous CDS protocols for multiplicative relation had a quadratic overhead in communication. This result is important as many crypto-computing protocols can be made secure against active attacks by verifying multiplicative relations. These relations naturally occur in the computation of Beaver tuples and shared message authentication codes [DPSZ12].

A new CDS protocol for multiplicative relation is presented in Section 4 and main results about zero-knowledge proofs are presented in Section 5. Hence, a reader who is familiar with basics of conditional disclosure of secrets can skip Sections 2 and 3. For clarity, all of our results are formalised in the concrete security framework and statements about polynomial model are obtained by considering the asymptotic behaviour w.r.t. the security parameter.

2 Preliminaries

We use boldface letters for vectors and calligraphic letters for sets and algorithms. A shorthand $m \leftarrow \mathcal{M}$ denotes that m is chosen uniformly from a set

$\mathcal{M}$. For algorithms and distributions, the same notation $x \leftarrow \mathcal{A}$ means that the element is sampled according to the (output) distribution. A shorthand $A \equiv B$ denotes that either distributions or elements A and B are identical. All algorithms are assumed to be specified as inputs (*programs*) to a universal Turing machine $\mathcal{U}$. A *t-time* algorithm is an algorithm that is guaranteed to stop in t time steps. In particular note that the program length of a t-time algorithm must be less than t bits.[1] To make definitions more concise, we use stateful adversaries. Whenever an adversary $\mathcal{A}$ is called, it has read-write access to a state variable $\sigma \in \{0,1\}^*$ through which important information can be sent from one stage of the attack to another. In the beginning of the execution σ is empty.

Homomorphic encryption. As all protocols for conditional disclosure of secrets are based on homomorphic encryption schemes, we have to formalise corresponding security notions. A *public key encryption scheme* is specified by a triple of efficient algorithms $(\mathsf{gen}, \mathsf{enc}, \mathsf{dec})$. The probabilistic key generation algorithm gen generates a public key pk and a secret key sk. The deterministic algorithms $\mathsf{enc}_{\mathsf{pk}} : \mathcal{M}_{\mathsf{pk}} \times \mathcal{R}_{\mathsf{pk}} \to \mathcal{C}_{\mathsf{pk}}$ and $\mathsf{dec}_{\mathsf{sk}} : \mathcal{C}_{\mathsf{pk}} \to \mathcal{M}_{\mathsf{pk}}$ are used for encryption and decryption, where the message space $\mathcal{M}_{\mathsf{pk}}$, the randomness space $\mathcal{R}_{\mathsf{pk}}$ and the ciphertext space $\mathcal{C}_{\mathsf{pk}}$ might depend on pk. As usual, we use $\mathsf{enc}_{\mathsf{pk}}(m)$ to denote the distribution of ciphertexts $\mathsf{enc}_{\mathsf{pk}}(m;r)$ for $r \leftarrow \mathcal{R}_{\mathsf{pk}}$.

In this work, we put two important additional restrictions to the encryption scheme. First, encryption scheme must be with *perfect decryption*:

$$\forall(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{gen} \quad \forall m \in \mathcal{M}_{\mathsf{pk}} : \quad \mathsf{dec}_{\mathsf{sk}}(\mathsf{enc}_{\mathsf{pk}}(m)) = m \ .$$

Second, membership for the set $\mathcal{C}_{\mathsf{pk}}$ must be efficiently testable given pk, i.e., everybody should be able to tell whether a message is a valid ciphertext or not.

Some cryptosystems have specific ways to combine ciphertexts, which lead to predictable changes in plaintexts. We say that an encryption scheme is *additively homomorphic* if there exists an efficient binary operation $\cdot$ such that

$$\forall m_1, m_2 \in \mathcal{M}_{\mathsf{pk}} : \quad \mathsf{enc}_{\mathsf{pk}}(m_1) \cdot \mathsf{enc}_{\mathsf{pk}}(m_2) \equiv \mathsf{enc}_{\mathsf{pk}}(m_1 + m_2) \tag{1}$$

where the equivalence means that the corresponding distributions coincide. A cryptosystem is *multiplicatively homomorphic* if there exists an efficient binary operation $\circledast$ such that

$$\forall m_1, m_2 \in \mathcal{M}_{\mathsf{pk}} : \quad \mathsf{enc}_{\mathsf{pk}}(m_1) \circledast \mathsf{enc}_{\mathsf{pk}}(m_2) \equiv \mathsf{enc}_{\mathsf{pk}}(m_1 \cdot m_2) \ . \tag{2}$$

First, these definitions directly imply that if a fixed ciphertext $\mathsf{enc}_{\mathsf{pk}}(m_1)$ is combined with a freshly generated $\mathsf{enc}_{\mathsf{pk}}(m_2)$ then nothing except $m_1 + m_2$ or $m_1 \cdot m_2$ can be deduced from the resulting ciphertext. Secondly, the message space must be cyclic or a direct product of cyclic subgroups. In the following, we refer to these as *simple* and *vectorised* cryptosystems to make a clear distinction.

[1] The direct translation to the polynomial security model is following. The universal Turing machine can be converted to a non-uniform polynomial adversary $\mathcal{A}$ that takes 1^t and the code x as an advice and runs $\mathcal{U}(x)$ to interact with the environment.

An encryption scheme is (t, ε)-*IND-CPA secure*, if for any t-time adversary $\mathcal{A}$, the corresponding distinguishing advantage is bounded:

$$2 \cdot \left| \Pr \left[\begin{array}{l} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{gen}, (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pk}), \\ b \leftarrow \{0,1\}, c \leftarrow \mathsf{enc}_{\mathsf{pk}}(m_b) : \mathcal{A}(c) = b \end{array} \right] - \frac{1}{2} \right| \leq \varepsilon \ .$$

The ElGamal [EG85] and Paillier [Pai99] cryptosystems are the most commonly used cryptosystems that satisfy all these requirements under standard number theoretic assumptions. The ElGamal cryptosystem is multiplicatively homomorphic and the ciphertext space has an efficient membership test if it is built on top of elliptic curve with prime number of elements. The Paillier encryption and its extension Damgård-Jurik [DJ01] cryptosystem are additively homomorphic with ciphertext space $\mathbb{Z}^*_{N^k}$, which is also efficiently testable. Both of these cryptosystems have a cyclic message space.

Commitment schemes. A commitment scheme is specified by triple of efficient probabilistic algorithms $(\mathsf{gen}, \mathsf{com}, \mathsf{open})$. The algorithm gen fixes public parameters ck. The algorithm $\mathsf{com}_{\mathsf{ck}} : \mathcal{M}_{\mathsf{ck}} \to \mathcal{C}_{\mathsf{ck}} \times \mathcal{D}_{\mathsf{ck}}$ maps messages into commitment and decommitment pairs. A commitment is opened by applying an algorithm $\mathsf{open}_{\mathsf{ck}} : \mathcal{C}_{\mathsf{ck}} \times \mathcal{D}_{\mathsf{ck}} \to \mathcal{M}_{\mathsf{ck}} \cup \{\perp\}$ where the symbol $\perp$ indicates that the commitment-decommitment pair is invalid. We assume that

$$\forall \mathsf{ck} \leftarrow \mathsf{gen} \quad \forall m \in \mathcal{M}_{\mathsf{ck}} : \quad \mathsf{open}_{\mathsf{ck}}(\mathsf{com}_{\mathsf{ck}}(m)) = m \ .$$

A commitment scheme is (t, ε)-*hiding*, if for any t-time adversary $\mathcal{A}$, the corresponding distinguishing advantage is bounded by ε:

$$2 \cdot \left| \Pr \left[\begin{array}{l} \mathsf{ck} \leftarrow \mathsf{gen}, (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{ck}), \\ b \leftarrow \{0,1\}, (c, d) \leftarrow \mathsf{com}_{\mathsf{ck}}(m_b) : \mathcal{A}(c) = b \end{array} \right] - \frac{1}{2} \right| \leq \varepsilon \ .$$

A commitment scheme is (t, ε)-*binding*, if for any t-time adversary $\mathcal{A}$, the probability of successful double-openings is bounded by ε:

$$\Pr \left[\begin{array}{l} \mathsf{ck} \leftarrow \mathsf{gen}, (c, d_0, d_1) \leftarrow \mathcal{A}(\mathsf{ck}) : \mathsf{open}_{\mathsf{ck}}(c, d_0) \neq \perp \\ \wedge \, \mathsf{open}_{\mathsf{ck}}(c, d_1) \neq \perp \wedge \mathsf{open}_{\mathsf{ck}}(c, d_0) \neq \mathsf{open}_{\mathsf{ck}}(c, d_1) \end{array} \right] \leq \varepsilon \ .$$

A commitment is ε-*binding* if the bound holds for all adversaries.

A commitment scheme is *perfectly equivocal* if there exists a modified setup procedure gen^* that in addition to ck produces an equivocation key ek, which is later used by additional algorithms $\mathsf{com}^*_{\mathsf{ek}}$ and $\mathsf{equiv}_{\mathsf{ek}}$. The algorithm $\mathsf{com}^*_{\mathsf{ek}}$ returns a fake commitment c_* together with a trapdoor information σ such that the invocation of $\mathsf{equiv}_{\mathsf{ek}}(\sigma, m)$ produces a valid decommitment d_* for m, i.e., $\mathsf{open}_{\mathsf{ck}}(c_*, d_*) = m$. More over, for any message $m \in \mathcal{M}_{\mathsf{ck}}$ the distribution (c_*, d_*) must coincide with the distribution generated by $\mathsf{com}_{\mathsf{ck}}(m)$.

In some proofs, we need commitments that are simultaneously equivocal and ε-binding. To achieve such a chameleon-like behaviour, we can use two commitment schemes, which differ only in the key generation. That is, they use the same

algorithms com and open for committing and decommitting. Now if commitment parameters ck are (t, ε_1)-indistinguishable and the first commitment scheme is perfectly equivocal and the second ε_2-binding, we can switch the key generation algorithms during security analysis and use both properties. This construction is known as a (t, ε_1)-*equivocal and* ε_2-*binding dual-mode commitment.*

Such commitments can be constructed from additively homomorphic encryption, see [GOS06]. Let $e \leftarrow \mathsf{enc}_{\mathsf{pk}}(1)$ together with pk be the commitment key. Then we can commit $m \in \mathcal{M}_{\mathsf{pk}}$ by computing $c \leftarrow e^m \cdot \mathsf{enc}_{\mathsf{pk}}(0; r)$ for $r \leftarrow \mathcal{R}$. To open c, we have to release m and r. This construction is perfectly binding. To assure equivocality, we can set $e \leftarrow \mathsf{enc}_{\mathsf{pk}}(0; r_*)$ for some $r_* \leftarrow \mathcal{R}$. Then any commitment c is an encryption of zero and can be expressed as $e^m \cdot \mathsf{enc}_{\mathsf{pk}}(0; r)$ provided that r_* is known. It is easy to see that (t, ε)-IND-CPA security is sufficient to guarantee hiding and computational indistinguishability of commitment keys. As you never need to decrypt during the equivocation, the construction can be based on lifted ElGamal or Paillier encryption scheme.

Trusted setup model. In this model, a trusted dealer $\mathcal{T}$ computes and privately distributes all public and secret parameters according to some procedure π_{ts}. For instance, $\mathcal{T}$ might set up a public key infrastructure or generate a common reference string. In practical applications, the trusted setup π_{ts} is commonly implemented as a secure two- or multi-party protocol run in *isolation.* Hence, protocols with trusted setup are practical only if many protocols can share the same setup without rapid decrease in security.

3 Conditional Disclosure of Secrets

A *conditional disclosure of secrets* (CDS) is a two-message protocol between a client $\mathcal{P}$ and a server $\mathcal{V}$ where the client learns a secret s specified by the server only if its encrypted inputs $\boldsymbol{x} = (x_1, \ldots, x_n)$ satisfy a public predicate $\phi(\boldsymbol{x})$. The server should learn nothing beyond the vector of encryptions $(q_1, \ldots, q_n)$. We also assume that the client knows the secret key sk, whereas the server knows only the public key pk. These protocols are often used as implicit sub-protocols in more complex crypto-computing protocols, see for example [AIR01,BK04].

The complexity of CDS protocol depends on the predicate. For instance, it is straightforward to construct CDS protocols for all monotone predicates if the input $\boldsymbol{x}$ is a bit vector [AIR01,LL07]. These constructions can be used as a basis for more complex predicates. In particular, note that for any predicate $\phi(\boldsymbol{x})$ there exists a constant depth monotonous predicate $\psi(\boldsymbol{x}, \boldsymbol{w})$ such that

$$\phi(\boldsymbol{x}) = 1 \quad \Leftrightarrow \quad \exists \boldsymbol{w} : \psi(\boldsymbol{x}, \boldsymbol{w}) = 1$$

and $\boldsymbol{w}$ can be efficiently computed from $\boldsymbol{x}$ and ϕ. For the conversion, fix a circuit that computes $\phi(\boldsymbol{x})$. Let $w_1, \ldots, w_k$ denote the output values of all gates in the circuit when the input is $\boldsymbol{x}$. Then you can define a monotonous formula $\psi(\boldsymbol{x}, \boldsymbol{w})$, which states that all gates are correctly evaluated and the output of the circuit is one. Clearly, the circuit complexity of ψ is linear in the circuit complexity

of ϕ. As a result, efficient CDS protocols exist for all predicates provided that the client is willing to encrypt $\boldsymbol{w}$ besides $\boldsymbol{x}$ and the server is willing to combine encryptions. See [AIR01,LL07] for more detailed discussions.

Formal security definition. A CDS protocol is specified by a triple of algorithms $(\mathsf{query}, \mathsf{answer}, \mathsf{recov})$. The client first sends a query $\boldsymbol{q} \leftarrow \mathsf{query}_{\mathsf{pk}}(\boldsymbol{x}, \boldsymbol{w})$ for which the server computes a reply $\boldsymbol{a} \leftarrow \mathsf{answer}_{\mathsf{pk}}(\boldsymbol{q}, s; r)$ for $r \leftarrow \mathcal{R}$. The client can recover the secret by computing $\mathsf{recov}_{\mathsf{sk}}(\boldsymbol{a})$.

Some CDS protocols also specify how the server must combine ciphertexts to get new encryptions with specific properties. We omit this step from the protocol description as it is a separate local post-processing step. For clarity, let $\mathcal{Q}_{\mathrm{inv}}$ denote the *set of all invalid queries*, i.e., queries for which $\psi(\boldsymbol{x}, \boldsymbol{w}) = 0$ or which contain invalid ciphertext or are otherwise malformed.

All standard implementations of CDS protocols are secure in the relaxed model, where the client can be malicious and the server is assumed to be honest but curious. Hence, a security definition should be expressed in terms of simulator constructions. However, as the protocol structure is so simple, we can be more explicit. Namely, a CDS protocol is (ε, t_a)-*client-private*, if for any t_a-time stateful adversary $\mathcal{A}$, the next inequality holds:

$$2 \cdot \left| \Pr \left[\begin{array}{l} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{gen}, (\boldsymbol{x}_0, \boldsymbol{w}_0, \boldsymbol{x}_1, \boldsymbol{w}_1) \leftarrow \mathcal{A}(\mathsf{pk}), \\ i \leftarrow \{0,1\}, \boldsymbol{q} \leftarrow \mathsf{query}_{\mathsf{pk}}(\boldsymbol{x}_i, \boldsymbol{w}_i) : \mathcal{A}(\boldsymbol{q}) = i \end{array} \right] - \frac{1}{2} \right| \leq \varepsilon \ .$$

To simulate a malicious client $\mathcal{P}_*$, the simulator can forward its query $\boldsymbol{q}$ to the trusted third party $\mathcal{T}$ who will test it. If $\boldsymbol{q} \notin \mathcal{Q}_{\mathrm{inv}}$, $\mathcal{T}$ releases the secret s and the simulator can compute $\mathsf{answer}_{\mathsf{pk}}(\boldsymbol{q}, s)$. Otherwise, $\mathcal{T}$ will release nothing and we need an efficient algorithm $\mathsf{answer}^*_{\mathsf{pk}}(\boldsymbol{q})$ for faking replies without knowing the secret. A CDS protocol is ε-*server private* if, for all valid public keys pk and secrets $s \in \mathcal{S}$ and for all invalid queries $\boldsymbol{q} \in \mathcal{Q}_{\mathrm{inv}}$, the statistical distance between the distributions $\mathsf{answer}_{\mathsf{pk}}(\boldsymbol{q}, s)$ and $\mathsf{answer}^*_{\mathsf{pk}}(\boldsymbol{q})$ is at most ε. As the statistical distance of replies is at most ε, the joint output distribution is also at most ε apart and thus the protocol is secure against malicious clients [LL07].

Example protocols. The simplest of CDS protocols is a *disclose-if-one* protocol, where the client $\mathcal{P}$ learns the secret only if the server $\mathcal{V}$ receives encryption of one. For clarity, let us consider cryptosystems with a cyclic plaintext space of size p. A standard way to build such a protocol relies on the fact that

$$\mathsf{enc}_{\mathsf{pk}}(x)^e \circledast \mathsf{enc}_{\mathsf{pk}}(s) \equiv \mathsf{enc}_{\mathsf{pk}}(x^e \cdot s)$$

when the underlying encryption scheme is multiplicatively homomorphic. If p is publicly known prime then e be chosen uniformly from $\mathbb{Z}_p$ and it is easy to prove that $x^e \cdot s$ is uniformly distributed over $\mathcal{M}_{\mathsf{pk}}$ if $x \neq 1$ and $x^e \cdot s = s$ otherwise. The latter forms a core of many CDS constructions [AIR01,BGN05]. For additively homomorphic encryption schemes, we can utilise the equality

$$\mathsf{enc}_{\mathsf{pk}}(x)^e \cdot \mathsf{enc}_{\mathsf{pk}}(s) \equiv \mathsf{enc}_{\mathsf{pk}}(xe + s)$$

to construct a *disclose-if-zero* protocol. As commonly used additively homomorphic encryption schemes have a composite plaintext space $\mathbb{Z}_N$, extra care is required to address cases when x is non-trivial factor of N. In such cases, the noise term xe is uniformly distributed over non-zero additive subgroup $\mathcal{G} \subseteq \mathbb{Z}_N$ and additional randomness is needed to hide the secret. Laur and Lipmaa [LL07] proposed a solution where the secret is first encoded with a randomised substitution cipher encode. Thus, the client learns

$$\mathsf{enc}_{\mathsf{pk}}(x)^e \cdot \mathsf{enc}_{\mathsf{pk}}(\mathsf{encode}(s)) \equiv \mathsf{enc}_{\mathsf{pk}}(xe + \mathsf{encode}(s)) \ .$$

If there exists an efficient function decode such that $\mathsf{decode}(\mathsf{encode}(s)) = s$ for all $s \in \mathcal{S}$, the honest client can still recover the secret. For the security, $\mathsf{encode}(s)$ must contain enough randomness so that an additive noise from a small subgroup can hide the secret when the query is invalid. We say that the encoding is ε-*secure* if for any $s \in \mathcal{S}$ and for any non-zero additive subgroup $\mathcal{G} \subseteq \mathcal{M}_{\mathsf{pk}}$, $\mathsf{encode}(s) + g$ for $g \leftarrow \mathcal{G}$ is statistically ε-close to uniform distribution over $\mathcal{M}_{\mathsf{pk}}$.

Secure encodings. If the message space $\mathcal{M}_{\mathsf{pk}}$ has a prime order, the only non-zero subgroup is $\mathcal{M}_{\mathsf{pk}}$ and thus the identity function can be used as perfectly secure encoding. For composite message spaces $\mathbb{Z}_N$, we can choose t randomly from $\mathbb{Z}_{\lfloor N/2^\ell \rfloor}$ and set $\mathsf{encode}(s) = s + 2^\ell \cdot t$ to encode ℓ-bit secrets. Laur and Lipmaa showed that this encoding is $2^{\ell-1}/\gamma$-secure where γ is the smallest factor of N and there are no alternative encoding functions with significantly longer secrets [LL07]. See App. A for a more detailed discussion about optimality.

This construction can be lifted to the vectorised setting provided that all plaintext components have the same public order n as in [SV11,GHS12,DPSZ12]. However, the information about s must be split between different plaintext components to assure security. For instance, if the plaintext space is $\mathbb{Z}_p \times \mathbb{Z}_p$, the subgroup generated by $(0, 1)$ is non-zero while the first component of $\mathsf{encode}(s) + g$ for $g \leftarrow \mathcal{G}$ comes without a protective noise. Hence, an additive secret sharing of $s = s_1 + s_2$ is needed to assure that the secret is recovered only if ciphertext corresponds to $(0, 0)$. The same technique is applicable for message spaces with composite order when we additionally encode each share s_i. As the noise hides at least one share s_i when $\mathcal{G} \neq \{\mathbf{0}\}$, the secret s becomes irrecoverable.

4 A New CDS Protocol for a Multiplicative Relation

For clarity, we specify the solution for additively homomorphic encryption and then discuss how the same protocol can be modified to work with other types of cryptosystems such as the lifted ElGamal and vectorised cryptosystems.

Let $\mathsf{enc}_{\mathsf{pk}}(x_1)$, $\mathsf{enc}_{\mathsf{pk}}(x_2)$ and $\mathsf{enc}_{\mathsf{pk}}(x_3)$ be the ciphertexts sent by the client $\mathcal{P}$ and let $s \in \mathcal{S}$ be the secret picked by the server $\mathcal{V}$. Then the client should learn s only if the multiplicative relation $x_1 x_2 = x_3$ holds between plaintexts. Figure 1 depicts the corresponding CdsMul protocol.

Theorem 1. *If the encryption scheme is* (t, ε_1)*-IND-CPA secure and* encode *is* ε_2*-secure, the* CdsMul *protocol is* $(t, 3\varepsilon_1)$*-client and* ε_2*-server private.*

GLOBAL PARAMETERS: Both parties know functions encode and decode for secrets. The client $\mathcal{P}$ has a secret key sk and the server $\mathcal{V}$ has the corresponding public key pk. Let n be a publicly known common multiple of all cyclic subgroup sizes in $\mathcal{M}_{\mathsf{pk}}$.

CLIENT'S INPUT: The client $\mathcal{P}$ has inputs x_1, x_2, x_3 such that $x_1 x_2 = x_3$ over $\mathcal{M}_{\mathsf{pk}}$
SERVER'S SECRET: The server $\mathcal{V}$ wants to release a secret $s \in \mathcal{S}$.

QUERY: The client $\mathcal{P}$ sends $\boldsymbol{q} = (q_1, q_2, q_3)$ to the server $\mathcal{V}$ where $q_i = \mathsf{enc}_{\mathsf{pk}}(x_i)$.
ANSWER: The server $\mathcal{V}$ picks $e_1, e_2 \leftarrow \mathbb{Z}_n$ and and sends back

$$u_1 \leftarrow q_1^{e_1} \cdot \mathsf{enc}_{\mathsf{pk}}(e_2), \;\; u_2 \leftarrow q_3^{e_1} \cdot q_2^{e_2} \cdot \mathsf{enc}_{\mathsf{pk}}(\mathsf{encode}(s)) \;\; .$$

RECOVERY: $\mathcal{P}$ computes $d_i \leftarrow \mathsf{dec}_{\mathsf{sk}}(u_i)$ for $i \in \{1, 2\}$ and uses the known plaintext value $x_2 = \mathsf{dec}_{\mathsf{sk}}(q_2)$ to compute the output $s \leftarrow \mathsf{decode}(d_2 - x_2 d_1)$.

Figure1. CDS protocol CDSMUL for multiplicative relation.

Proof. As the output computed by the client satisfies the following equation

$$\begin{aligned} d_2 - x_2 d_1 &= x_3 e_1 + x_2 e_2 + \mathsf{encode}(s) - x_2(x_1 e_1 + e_2) \\ &= (x_3 - x_1 x_2) e_1 + \mathsf{encode}(s) \;\; , \end{aligned}$$

the client is guaranteed to recover the secret s when $x_1 x_2 = x_3$. If $x_1 x_2 \neq x_3$ the term $(x_3 - x_1 x_2) e_1$ adds additive noise to the payload. Clearly, the term $(x_3 - x_1 x_2) e_1$ belongs to a cyclic additive subgroup $\mathcal{G} = \{(x_3 - x_1 x_2) m : m \in \mathbb{Z}\}$. Since the size of every cyclic subgroup of $\mathcal{M}_{\mathsf{pk}}$ divides n, the term $(x_3 - x_1 x_2) e_1$ must be uniformly distributed over $\mathcal{G}$. Consequently, we have established that $d_2 - x_2 d_1 \equiv \mathsf{encode}(s) + \mathcal{G}$ for $\mathcal{G} \neq \{0\}$. By the assumptions, $\mathcal{G} + \mathsf{encode}(s)$ is ε_2-close to the uniform distribution over $\mathcal{M}_{\mathsf{pk}}$. More importantly, the distribution of $d_2 - x_2 d_1$ is also independent from e_2. As e_2 perfectly masks the term $x_1 e_1$ in d_1, we can simulate the replies u_1 and u_2 by encrypting two random messages. The claim on client-privacy is straightforward. □

Remarks. First, note that the protocol has perfect server-privacy when the message space is cyclic and has a prime order, since the identity function as encode has perfect privacy. If all non-trivial cyclic subgroups have the same prime order, we can use additive secret sharing to make sure that the multiplicative relation holds for every sub-component of a vectorised cryptosystem.

Second, note that the protocol does not work directly with lifted cryptosystems. In such schemes, the new encryption rule $\overline{\mathsf{enc}}_{\mathsf{pk}}(x) = \mathsf{enc}_{\mathsf{pk}}(g^x)$ is defined in terms of an old multiplicatively homomorphic encryption rule $\mathsf{enc}_{\mathsf{pk}}(\cdot)$ and a generator of the plaintext space g. The resulting scheme is additively homomorphic, but discrete logarithm must be taken to complete the decryption. Hence, we cannot blindly follow the reconstruction phase in CDSMUL protocol.

Still, the client can employ the old decryption algorithm to compute g^{d_1} and g^{d_2} from the lifted ciphertexts u_1 and u_2. Since the client knows x_2, he or she

can compute a partial decryption $g^{d_1}(g^{d_2})^{-x_2} = g^{(x_3 - x_1 x_2)e_1} g^{\mathsf{encode}(s)}$. Thus, the honest client learns $g^{\mathsf{encode}(s)}$, which might not be enough to extract s.

For lifted ElGamal with a prime order message space, we can use identity function to encode s. Hence, the secret s can be restored by brute-forcing g^s if the set of secrets is small. Alternatively, both parties can use g^s instead of s to share a randomly distributed value over $\mathcal{M}_{\mathsf{pk}}$. These solutions are not viable for lifted cryptosystems with composite message spaces, such as [BGN05].

5 From CDS protocols to Zero Knowledge

The new zero-knowledge protocol is based on the following observation. The client $\mathcal{P}$ is able to reconstruct the secret s in a CDS protocol only if ciphertexts satisfy a certain relation, i.e., $\boldsymbol{q} \notin \mathcal{Q}_{\mathrm{inv}}$. Hence, if $\mathcal{P}$ sends the secret s back to the honest server $\mathcal{V}$, the server is able to verify whether $\boldsymbol{q} \notin \mathcal{Q}_{\mathrm{inv}}$ or not. This leaks no information to the semihonest server, since $\mathcal{V}$ already knows s. Consequently, we can view this simple modification as an honest verifier zero-knowledge proof where the client $\mathcal{P}$ is a prover and the server $\mathcal{V}$ acts as a verifier.

However, if $\mathcal{V}$ acts maliciously, the secret s recovered by the honest $\mathcal{P}$ might leak additional information. To counter this attack, $\mathcal{V}$ should prove knowledge of s and randomness $r \in \mathcal{R}$ needed to compute the reply $\mathsf{answer}_{\mathsf{pk}}(\boldsymbol{q}, s; r)$ before $\mathcal{P}$ sends the secret. Since neither s nor r are among private inputs, $\mathcal{V}$ can release them so that $\mathcal{P}$ can validate the behaviour of $\mathcal{V}$ by recomputing $\mathsf{answer}_{\mathsf{pk}}(\boldsymbol{q}, s; r)$. As the secret becomes public, $\mathcal{P}$ must commit to its reply before the pair (s, r) is released or otherwise we lose soundness. The corresponding idea gives a rise to zero-knowledge protocol, which is depicted in Figure 2. The secret s is in this protocol is just a temporary challenge for the prover and has no persistent value. Depending on the properties of commitment scheme and the CDS protocol, we get zero-knowledge protocols with different properties.

5.1 Proper Model for Defining Security

By definition a CdsZk protocol verifies a relation between ciphertexts. As such it is fairly useless without another protocol that uses these ciphertexts in its computations to get an output. As these protocols must share a joint state (at minimum the same public key), standard composability results are not applicable [CR03]. A systematic way to prove security of such compound protocols is to split the proof into phases. In the first phase, we show that instances of CdsZk protocols are concurrently composable with other protocols if the shared state is generated by the trusted setup procedure. In the second phase, we replace the trusted setup with a secure multiparty protocol *run in isolation* and use the standard sequential composability result to prove that the setup protocol followed by the compound protocol remains secure.

Finding an efficient protocol to substitute the trusted setup might be technically challenging (see App. B), however, the corresponding security analysis is straightforward, see Section 6. Hence, we discuss here only the first phase.

TRUSTED SETUP FOR ENCRYPTION. Trusted dealer runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{pkc.gen}$ and sends pk as a public key of $\mathcal{P}$ to everyone. The secret key sk is sent securely to $\mathcal{P}$.

TRUSTED SETUP FOR CDSZK PROTOCOLS. Trusted dealer generates commitment parameters $\mathsf{ck} \leftarrow \mathsf{cs.gen}$ and broadcast ck to everyone.

MESSAGE FORMATION. The CDS protocol is chosen according to the predicate ψ. The prover $\mathcal{P}$ sends $\boldsymbol{q} \leftarrow \mathsf{query}_{\mathsf{pk}}(\boldsymbol{x}, \boldsymbol{w})$ to the verifier $\mathcal{V}$.

PROOF PHASE. A statement to be proved is $\boldsymbol{q} \notin \mathcal{Q}_{\mathrm{inv}}$.

1. $\mathcal{V}$ chooses $s \leftarrow \mathcal{S}$ and $r \leftarrow \mathcal{R}$ and sends $\boldsymbol{a} \leftarrow \mathsf{answer}_{\mathsf{pk}}(\boldsymbol{q}, s; r)$ to $\mathcal{P}$.
2. $\mathcal{P}$ recovers $s \leftarrow \mathsf{recov}_{\mathsf{sk}}(\boldsymbol{a})$, computes $(c, d) \leftarrow \mathsf{cs.com}_{\mathsf{ck}}(s)$ and sends c to $\mathcal{V}$.
3. $\mathcal{V}$ reveals (s, r) to $\mathcal{P}$ who aborts if $\boldsymbol{a} \neq \mathsf{answer}_{\mathsf{pk}}(\boldsymbol{q}, s; r)$.
4. $\mathcal{P}$ reveals decommitment d to $\mathcal{V}$ who accepts the proof only if $s = \mathsf{cs.open}_{\mathsf{ck}}(c, d)$.

Figure2. Zero-knowledge proof of correctness CDSZK for encryptions.

First of all, note that a CDSZK protocol might be used to prove statements that depend on common parameters z shared by all participants after the trusted setup. To handle such cases, the language of all valid statements $\mathcal{L}_z = \{x \mid \exists w : \psi_*(z, x, w) = 1\}$ must be specified by an efficiently decidable ternary predicate ψ_*. As usual x is the *protocol input* (statement) and w is the *witness* and z additional auxiliary input. For a CDSZK protocol, $\boldsymbol{q}$ is the protocol input, sk is the witness, and pk is the common parameter z shared by many CDSZK protocols and crypto-computing protocols using CDSZK. The statement $\boldsymbol{q}$ belongs to $\mathcal{L}_{\mathsf{pk}}$ iff $\boldsymbol{q} \notin \mathcal{Q}_{\mathrm{inv}}(\mathsf{pk})$. Indeed, the knowledge of sk allows us to decrypt $\boldsymbol{q}$ to obtain $\boldsymbol{x}$ and then output $\phi(\boldsymbol{x})$ as $\psi_*(\mathsf{pk}, \boldsymbol{q}, \mathsf{sk})$.

A *proof system* is determined by two algorithms $\mathcal{P}$ and $\mathcal{V}$ which specify the actions of the prover and the verifier, respectively. The prover $\mathcal{P}$ takes (x, w) as inputs and the verifier $\mathcal{V}$ takes x as an input. Both algorithms can additionally access parameters distributed to them by the setup π_{ts}. Let $\langle\!\langle \mathcal{P}, \mathcal{V} \rangle\!\rangle(x)$ denote the verifiers output after the interaction with $\mathcal{P}(x, w)$. Then a proof system is *perfectly complete* if all valid statements are provable, i.e., for all statements $x \in \mathcal{L}_z$ and corresponding witnesses w, $\langle\!\langle \mathcal{P}, \mathcal{V} \rangle\!\rangle(x) \equiv 1$ for all runs of $\pi^{\circ}_{\mathsf{ts}}$.

We formalise the security of a zero-knowledge protocol by specifying the ideal functionality. An ideal implementation of a zero knowledge proof is a restricted communication channel $\pi^{\circ}_{\psi_*}$ from the prover $\mathcal{P}$ to the verifier $\mathcal{V}$. The prover $\mathcal{P}$ can provide an input x to the channel, after which the verifier $\mathcal{V}$ obtains[2]

$$\pi^{\circ}_{\psi_*}(x) = \begin{cases} x, & \text{if } \exists w : \psi_*(z, x, w) = 1 \ , \\ \bot, & \text{if } \forall w : \psi_*(z, x, w) = 0 \ , \end{cases} \tag{3}$$

where the set of shared parameters z is determined by the trusted setup $\pi^{\circ}_{\mathsf{ts}}$.

[2] Another alternative is to send (x, b) over the channel where $b = \psi_*(z, x, w)$. This models the setting where the verifier does not discard x but just marks it as invalid.

Let $\mathcal{E}\langle \pi^{\circ}_{\mathsf{ts}}, \pi^{\circ}_{\psi_1}, \ldots, \pi^{\circ}_{\psi_\ell} \rangle$ denote a compound protocol (*computational context*), which internally uses a single instance of $\pi^{\circ}_{\mathsf{ts}}$ and ideal zero-knowledge protocols $\pi^{\circ}_{\psi_1}, \ldots, \pi^{\circ}_{\psi_\ell}$. The compound protocol can model any kind of computational activity that uses the functionality of $\pi^{\circ}_{\psi_1}, \ldots, \pi^{\circ}_{\psi_\ell}$. For instance, it can use zero-knowledge to guarantee correctness of an e-voting protocol.

We say that $\mathcal{E}$ is a t_e-*time computational context* if the maximal amount of computation steps done by $\mathcal{E}$ disregarding invocations of $\pi^{\circ}_{\mathsf{ts}}$ and $\pi^{\circ}_{\psi_i}$ is bounded by t_e. Now let $\mathcal{E}\langle \pi^{*}_{\mathsf{ts}}, \pi_1, \ldots, \pi_\ell \rangle$ be a *hybrid implementation*[3] with augmented trusted setup π^{*}_{ts} and real instantiations of zero knowledge protocols π_{ψ_i}. The augmented setup procedure first runs $\pi^{\circ}_{\mathsf{ts}}$ and then creates some additional parameters (commitment parameters in our case) shared by $\pi_1, \ldots, \pi_\ell$. Now we would like that both compound protocols have comparable security against plausible attacks. The corresponding formalisation is rather technical. Hence, we only highlight the major aspects in the formal definition and refer to the manuscripts [Can01,Lin03] for further details. Still, it is important to emphasise that we consider only security against static corruption but semi-adaptively chosen contexts[4] and we assume that the communication is asynchronous.

In a nutshell, let $\boldsymbol{\xi}$ denote inputs of all parties and let $\boldsymbol{\zeta}^{\circ}$ and $\boldsymbol{\zeta}$ denote joint outputs of all parties in the ideal and the hybrid execution. Let f be a low-degree polynomial. Then a protocol π is $(t_e, t_a, f, t_d, \varepsilon)$-*universally composable* in the shared setup model if for any t_e-time $\mathcal{E}$ and for any t_a-time adversary $\mathcal{A}$ against π there exists $f(t_a)$-time adversary $\mathcal{A}^{\circ}$ against π° such that for any input $\boldsymbol{\xi}$ distributions $\boldsymbol{\zeta}^{\circ}$ and $\boldsymbol{\zeta}$ are (t_d, ε)-indistinguishable. We omit the time bound t_d if $\boldsymbol{\zeta}^{\circ}$ and $\boldsymbol{\zeta}$ are statistically indistinguishable.

The parameter t_e limits the complexity of protocols where we can safely use zero-knowledge protocols. The parameter t_a shows how much computational power the adversary can have before we lose security guarantees. The polynomial f shows how much extra power the adversary gains by participating the hybrid protocol. The slower it grows the better. In the standard asymptotic setting, we view all parameters as functions of a security parameter k and we ask whether for any polynomials $t_e(k)$, $t_a(k)$, $t_d(k)$, the distinguishing advantage $\varepsilon(k)$ decreases faster than a reciprocal of any polynomial.

5.2 Soundness Guarantees for CDSZK Protocols

A proof system π is ε-*sound* if for any prover $\mathcal{P}_*$ and adaptively chosen statement x the deception probability is bounded by ε:

$$\mathsf{Adv}^{\mathrm{snd}}_{\pi}(\mathcal{P}_*) = \Pr\left[x \leftarrow \mathcal{P}_* : \langle\!\langle \mathcal{P}_*, \mathcal{V} \rangle\!\rangle(x) = 1 \wedge x \notin \mathcal{L}_z\right] \leq \varepsilon$$

[3] The name hybrid implementation is used here to distinguish the implementation from the final (real world) setting where π^{*}_{ts} is replaced with a setup protocol π_{ts}.

[4] This setting guarantees security against inside attacks even if the corrupted party can influence what is computed and in which contexts zero-knowledge proofs are executed., i.e., the adversary must decide before the execution which of two parties it corrupts. However, the adversary can choose the *computational context* $\mathcal{E}$ based on the outputs received in π_{ts}. Secondly, we limit the overall amount of computations.

where the probability is taken over all possible runs of the setup procedure π_{ts} and $\mathcal{P}_*$ can choose the statement x based on all public and private parameters received during π_{ts}. An argument system π is (t,ε)-*sound* if for any t-time $\mathcal{P}_*$ the deception probability is bounded: $\mathsf{Adv}^{\mathrm{snd}}_{\pi}(\mathcal{P}_*) \le \varepsilon$.

Theorem 2. *If the commitment scheme is ε_2-binding and CDS protocol is ε_3-server private,* CDSZK *protocol is $\left(\frac{1}{|\mathcal{S}|}+\varepsilon_2+\varepsilon_3\right)$-sound for any $\mathcal{P}_*$. If the commitment scheme is perfectly binding, a stronger claim holds*:

$$\forall \mathsf{sk},\mathsf{ck},\ (\psi,\boldsymbol{q}) \leftarrow \mathcal{P}_*(\mathsf{sk},\mathsf{ck}) : \Pr\left[\langle\!\langle \mathcal{P}_*,\mathcal{V}\rangle\!\rangle(\boldsymbol{q})=1 \wedge \boldsymbol{q}\in\mathcal{Q}_{\mathrm{inv}}\right] \le 1/|\mathcal{S}|+\varepsilon_3 \ .$$

Proof. Consider a modified protocol where $\boldsymbol{a}$ is replaced with $\mathsf{answer}^*_{\mathsf{pk}}(\boldsymbol{q})$ whenever $\boldsymbol{q}\in\mathcal{Q}_{\mathrm{inv}}$. As the CDS protocol is ε_3-server private, the replacement can reduce the deception probability at most by ε_3. Let c denote the the commitment issued in the modified protocol and let $d_*\in\mathcal{D}$ be the first valid decommitment for c according to some fixed ordering over the decommitment space $\mathcal{D}$. Let $\hat{s}=\mathsf{open}_{\mathsf{ck}}(c,d_*)$ be the corresponding opening. Then

$$\Pr\left[\langle\!\langle \mathcal{P}_*,\mathcal{V}\rangle\!\rangle(\boldsymbol{q})=1 \wedge \boldsymbol{q}\in\mathcal{Q}_{\mathrm{inv}}\right] \le \Pr\left[\hat{s}=s \wedge \boldsymbol{q}\in\mathcal{Q}_{\mathrm{inv}}\right]+\varepsilon_2+\varepsilon_3 \ .$$

Indeed, note that $\mathcal{P}_*$ can succeed only if $\mathsf{open}_{\mathsf{ck}}(c,d)=s$ and thus all successful runs with $s\neq\hat{s}$ correspond to valid double openings. As the commitment scheme is statistically binding $\Pr\left[s=\mathsf{open}_{\mathsf{ck}}(c,d)\neq\hat{s}\right]\le\varepsilon_2$. To complete the proof, note that the fake reply $\mathsf{answer}^*_{\mathsf{pk}}(\boldsymbol{q})$ is independent of s (which is uniformly chosen in CDSZK) and thus $\Pr\left[\hat{s}=s\wedge\boldsymbol{q}\in\mathcal{Q}_{\mathrm{inv}}\right]\le\Pr\left[\hat{s}=s|\boldsymbol{q}\in\mathcal{Q}_{\mathrm{inv}}\right]\le 1/|\mathcal{S}|$.

For the strengthened claim, note that $\mathsf{answer}^*_{\mathsf{pk}}$ and $\mathsf{answer}_{\mathsf{pk}}(\boldsymbol{q},s)$ are ε_3-close for any valid pk. Now if the commitment is perfectly binding then $s=\hat{s}$ for any ck and we get the desired bound. □

Surprisingly enough, computational binding alone is not sufficient for soundness against time-bounded provers. The CDSZK protocol is sound only if the underlying CDS protocol is equivocal for incorrect queries. The latter is a non-trivial property to achieve. See App. D for further discussion.

5.3 Universal Composability of CDSZK Protocols

Usually, one gives a simulation construction for a malicious verifier to prove the zero-knowledge property. Here, we aim for more. Namely, we show that any adversary who attacks the CDSZK protocol π_{ψ_*} can be converted to an efficient adversary against the ideal implementation $\pi^{\circ}_{\psi_*}$ even if some side-computations are done in parallel with π_{ψ_*}. For that we modify the generation of parameters for the dual-mode commitments to get an equivocality key into the simulator.

Lemma 1. *If (t,ε_1)-equivocal and ε_2-binding dual-mode commitment is used in the hiding mode, then for any t_a-time malicious verifier $\mathcal{A}$ there exists $\mathcal{O}(t_a)$-simulator $\mathcal{A}^{\circ}$ that achieves perfect simulation for single* CDSZK *protocol.*

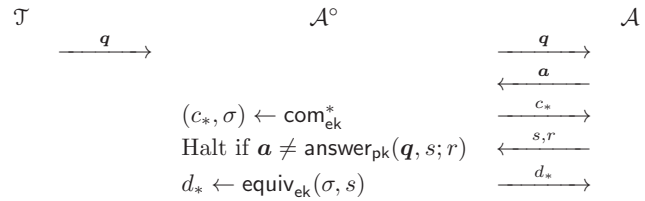

Figure3. Simulation construction for corrupted verifier.

Proof. To convert $\mathcal{A}$ to an equivalent ideal world adversary $\mathcal{A}^\circ$, we need to simulate π^*_{ts} given access to π°_{ts} and π_{ψ_*} given access to $\pi^\circ_{\psi_*}$. The simulation of π^*_{ts} is trivial, since π°_{ts} delivers all messages as π^*_{ts} except for the commitment key ck. For the latter, the simulator $\mathcal{A}^\circ$ computes $(\mathsf{ck}, \mathsf{ek}) \leftarrow \mathsf{gen}^*$ and delivers ck to $\mathcal{A}$ and stores the equivocation key ek for later use.

Figure 3 depicts the simulation construction for π_{ψ_*}. Let (c, d) denote the commitment and decommitment values used in π_{ψ_*} and (c_*, d_*) the corresponding values computed in the simulation. Since the commitment is perfectly equivocal, distributions of c and c_* coincide. Next, note that if $\boldsymbol{a} = \mathsf{answer}_{\mathsf{pk}}(\boldsymbol{q}, s; r)$, then honest prover would have committed s. Consequently, if corrupted verifier $\mathcal{A}$ releases values s, r such that $\boldsymbol{a} = \mathsf{answer}_{\mathsf{pk}}(\boldsymbol{q}, s; r)$, the distribution of (c, d) and (c_*, d_*) coincides, again. Hence, simulation of π_{ψ_*} is perfect. □

Note that the transformation outlined in Lemma 1 can be applied only once to the malicious verifier, as the resulting adversary $\mathcal{A}^\circ$ expects the trusted dealer to engage π°_{ts}. Therefore, we need a separate proof to show that concurrent composition of CdsZk protocols, which share the commitment parameters ck can be replaced with ideal implementations.

Lemma 2. *If (t, ε_1)-equivocal and ε_2-binding dual-mode commitment is used in the hiding mode, then for any t_a-time malicious verifier $\mathcal{A}$ there exists $\mathcal{O}(t_a)$-simulator $\mathcal{A}^\circ$ that achieves perfect simulation even if many* CdsZk *protocols are concurrently executed.*

Proof. We use the same simulator for the setup phase as in Lemma 1 and share the equivocation key ek for all sub-simulators that simulate π_{ψ_i}. For each protocol, we use the same construction as depicted in Figure 3. As the commitment is still perfectly equivocal, all claims about perfect simulatability of messages of the sub-protocols π_{ψ_i} hold and the claim follows. □

Lemma 2 shows how to handle corrupted verifiers but we also need to handle the case where the prover is malicious. The following result follows directly from soundness guarantees of Theorem 2.

Lemma 3. *If (t, ε_1)-equivocal and ε_2-binding dual-mode commitment is used in the binding mode and the underlying CDS protocols is ε_3-server private, then an unbounded prover can succeed with probability $\frac{1}{|\mathcal{S}|} + \varepsilon_2 + \varepsilon_3$ when $\boldsymbol{q} \in \mathcal{Q}_{\mathrm{inv}}$.*

Note that a corrupted party $\mathcal{P}_1$ can act simultaneously as a prover and verifier. Let pk_1 and pk_2 be the public key of $\mathcal{P}_1$ and $\mathcal{P}_2$. Then $\mathcal{P}_1$ can prove that encryptions under the public key pk_1 are well formed and let $\mathcal{P}_2$ to convince that encryptions under the key pk_2 satisfy certain relations. Hence, the final claim about the security of CdsZk must provide a simultaneous simulation. For brevity, let us prove the claim for two-party setting as the proof can be easily generalised for multi-party setting. Recall that $t_a + t_e$ is the maximal running time of the adversary and the context where the protocols are executed.

Theorem 3. *If (t, ε_1)-equivocal and ε_2-binding dual-mode commitment is used in the hiding mode and all ℓ instances of CDS protocols are ε_3-server private, then the* CdsZk *protocol is $(t_e, t_a, \mathcal{O}(t_a), t_d, \ell(\frac{1}{|\mathcal{S}|} + \varepsilon_2 + \varepsilon_3) + \varepsilon_1)$-universally composable in the shared setup model provided that $t \geq t_a + t_e + \mathcal{O}(\ell)$.*

Proof. Let $\mathcal{E}\langle \pi^{\circ}_{\mathsf{ts}}, \pi^{\circ}_{\psi_1}, \ldots, \pi^{\circ}_{\psi_\ell} \rangle$ denote a compound protocol with ideal implementations and let $\mathcal{E}\langle \pi_{\mathsf{ts}}, \pi_{\psi_1}, \ldots, \pi_{\psi_\ell} \rangle$ be the corresponding hybrid world implementation. Let $\mathcal{P}_1$ denote the corrupted and $\mathcal{P}_2$ the honest party. Let $\mathcal{A}$ denote a malicious adversary that acts in behalf of $\mathcal{P}_1$.

For the proof, we modify the simulator construction as in Lemma 2 so that it can handle protocol instances where $\mathcal{A}$ is a prover. For that $\mathcal{A}^{\circ}$ must forward the query $\boldsymbol{q}$ to trusted third party $\mathcal{T}$ and play the role of honest verifier $\mathcal{V}$. The latter is always possible as $\mathcal{V}$ has no inputs. Note that the simulator will accept some proofs where $\boldsymbol{q} \in \mathcal{Q}_{\mathrm{inv}}$ while the $\mathcal{P}_2$ reading the channel will always reject these proofs. Hence, there will be small discrepancy between ideal and hybrid executions. If the commitment scheme is in the binding mode, such events occur with probability $\varepsilon_s = \frac{1}{|\mathcal{S}|} + \varepsilon_2 + \varepsilon_3$ per each protocol instance due to Lemma 3. If the commitment scheme is in the hiding mode, such events can occur at probability $\ell\varepsilon_s + \varepsilon_1$ if $t_a + t_e + \mathcal{O}(\ell) \leq t$. Otherwise, we can use the hybrid execution model for distinguishing between commitment keys. Consequently, executions in the ideal and hybrid world can diverge with with probability $\ell\varepsilon_s + \varepsilon_1$ and the claim follows. □

Theorem 3 assures that CdsZk protocols are concurrently composable zero-knowledge arguments and not proofs, since unbounded prover can always fool verifiers. If the commitment scheme is run in the binding mode, we get zero-knowledge proofs but lose statistical simulatability.

Theorem 4. *If the commitment scheme is used in the binding mode, the* CdsZk *protocol is $(t_e, t_a, \mathcal{O}(t_a), t_d, \ell(\frac{1}{|\mathcal{S}|} + \varepsilon_2 + \varepsilon_3) + 2\varepsilon_1)$-universally composable in the shared setup model provided that $t \geq t_a + t_e + t_d + \mathcal{O}(\ell)$.*

Proof. Note that t_d algorithm can distinguish hybrid executions with different commitment modes with probability ε_1 if $t \geq t_a + t_e + t_d + \mathcal{O}(\ell)$. Hence, we can use hybrid argument to show that the simulator from Theorem 3 is sufficient. □

Concluding remarks. First, note that the trusted setup for public key infrastructure is essential. Although the commitment key ck can be viewed as common reference string, the CdsZk proofs are not guaranteed to be universally composable in the CRS model. The underlying CDS protocol is required to be server-private only if the public key pk is valid. Hence, a prover with an invalid public key may succeed in deception. There are no easy fix for this problem, as the public keys of most additively homomorphic cryptosystems are not efficiently verifiable. Second, note that CdsZk protocols can be proven secure in the standard model. However, we lose universal composability, see App. C.

5.4 Comparison with Other Zero-Knowledge Protocols

Efficient zero-knowledge protocols can be obtained form challenge-response or sigma protocols. The CdsZk protocol can be viewed as a challenge-response protocol followed by the proof of knowledge to tame cheating verifiers. However, differently from the standard solution [GMR89] where the proof of knowledge is executed before the prover sends the response, the commitment is used to fix the response. As result, the verifier can directly reveal the internal state yielding the challenge, while standard construction needs witness-indistinguishable proofs. Hence, our protocol is clearly superior compared to the standard approach.

More remarkably, our methodology can be used to convert any challenge-response protocol to zero-knowledge proof. By using dual-mode commitments, we can easily get security guarantees analogous to Theorems 3 and 4. If the challenge does not have a nice algebraic form (say we want to establish provers ability to decrypt ciphertexts or invert hash functions) then it is much more efficient to reveal all inputs (to the encryption algorithm or hashing function) directly instead of crafting witness-indistinguishable proofs of knowledge.

There is a strange duality between CDS and sigma protocols. In both cases, protocols for elementary relations can be combined into disjunctive and conjunctive proofs using secret sharing, see [CDS94,LL07]. As there are efficient sigma protocols for affine relations between plaintexts, the overhead of combiner constructions for both types of protocols is comparable.

Table 1 summarises the efficiency of the three most important elementary relations, where E, D, M, X correspond to encryption, decryption, multiplication and exponentiation operations. For the multiplicative relation, we use Damgård-Jurik sigma protocol from [DJ01] detailed in App. E. In all these CDS protocols, the client sends only the ciphertexts to be checked, while prover in a sigma protocols must send extra ciphertexts. Thus, CDS protocols have smaller communication complexity, while the computation complexity is comparable[5].

Dual-mode commitments with the trusted setup can be used to convert sigma protocols into universally composable zero-knowledge proofs. For that the prover must commit the challenge ahead of proof. By using extractable commitments

[5] Exact comparisons are hard to give, since the cost of decryption operations depends on implementation details, which vary from a cryptosystem to a cryptosystem.

Table1. The number of elementary operations done in sigma and CDS protocols.

Relation	Sigma protocol		CDS protocol	
	Prover	Verifier	Client	Server
$x = 0$	1E + 1M	1E + 1M + 1X	1D	1E + 1M + 1X
$ax = b$	1E + 1M	2E + 2M + 2X	1D	2E + 2M + 2X
$x_1x_2 = x_3$	2E + 4M	2E + 3M + 4X	2D + 1M	2E + 3M + 3X

during the simulation, we can extract the challenge and thus create a matching sigma protocol transcript without access to the witness.

The resulting zero-knowledge protocol has higher communication complexity than CdsZk proof and cannot be dynamically reconfigured. Namely, assume that $\mathcal{P}_1$ sends some encryptions to $\mathcal{P}_2$. Then $\mathcal{P}_2$ can add CDS transformation to his reply without prior agreement with $\mathcal{P}_1$ to guarantee input privacy or start honest verifier zero knowledge proof (HVZK). If the prover $\mathcal{P}_1$ feels frightened then it can upgrade HVZK proof to full-fledged zero knowledge by committing the reply. Again, there is no need for an agreement before this step. Such dynamic configurability is not attainable for other types of zero-knowledge proofs.

Zero-knowledge proofs for any NP language. The CdsZk protocol can be directly converted to general purpose zero-knowledge proof for any **NP** statement. Let $\psi(\boldsymbol{x}, \boldsymbol{w})$ be the predicate that checks validity of the witness $\boldsymbol{w}$ and the statement $\boldsymbol{x}$. Now if we encrypt $\boldsymbol{w}$, then we can use CdsZk to prove that we have indeed encrypted a witness to public statement $\boldsymbol{x}$. Since relatively efficient CDS protocols exist for any circuit [AIR01,LL07], the resulting CdsZk protocol is rather efficient. Since CdsZk protocol can be proved secure in the standard model (see App. C), the resulting proof system is no worse than any other general purpose zero-knowledge proof.

6 Securing Crypto-computations Against Active Attacks

Assume that we have a crypto-computing protocol $\mathcal{E}_0$ that is secure in the semi-honest setting and our goal is to protect the protocol against malicious client who has the secret key. Moreover, assume that the only way for the client to cheat is to send invalid ciphertexts to the server. For instance, send an incorrectly encrypted vote in the e-voting protocol. The are wide range of client-server protocols that can be formalised such way, see [AIR01,BK04,LL07].

All such crypto-computing protocols can be split into a computational phase and a trusted setup π^*_{ts} that sets up public and private keys. Let π°_{ts} be the augmented trusted setup, which additionally sets up the commitment parameters shared by all CdsZk protocols. Let π_{ts} be a secure two- or multi-party protocol that implements π°_{ts} in the standard model. Clearly, we can modify the protocol $\mathcal{E}_0\langle\pi^\circ_{\mathsf{ts}}\rangle$ by adding ideal zero-knowledge protocols $\pi^\circ_{\psi_i}$ to guarantee that the client cannot cheat. The resulting protocol can be viewed as a context $\mathcal{E}\langle\pi^\circ_{\mathsf{ts}}, \pi^\circ_{\psi_1}, \ldots, \pi^\circ_{\psi_\ell}\rangle$ for execution ideal zero knowledge protocols.

By substituting the real implementations of CdsZk, we get a hybrid execution model $\mathcal{E}\langle \pi^*_{\mathsf{ts}}, \pi_{\psi_1}, \ldots, \pi_{\psi_\ell} \rangle$. Finally, we can consider the crypto-computing protocol $\mathcal{E}\langle \pi_{\mathsf{ts}}, \pi_{\psi_1}, \ldots, \pi_{\psi_\ell} \rangle$ where first π_{ts} is run isolation to get all setup parameters and then the remaining protocols are concurrently scheduled. To prove the security of the resulting implementation, we have to use hybrid argument several times to show indistinguishability.

As π_{ts} is a secure implementation of π^*_{ts}, the standard sequential composability result guarantees that any efficient adversary $\mathcal{A}$ against $\mathcal{E}\langle \pi_{\mathsf{ts}}, \pi_{\psi_1}, \ldots, \pi_{\psi_\ell} \rangle$ can be converted to an efficient adversary $\mathcal{A}_1$ against $\mathcal{E}\langle \pi^*_{\mathsf{ts}}, \pi_{\psi_1}, \ldots, \pi_{\psi_\ell} \rangle$ so that outputs are computationally indistinguishable. Due to Theorems 3 and 4, we can convert $\mathcal{A}_1$ into an efficient adversary $\mathcal{A}_2$ against $\mathcal{E}\langle \pi^\circ_{\mathsf{ts}}, \pi^\circ_{\psi_1}, \ldots, \pi^\circ_{\psi_\ell} \rangle$ such that outputs remain indistinguishable. As the adversary $\mathcal{A}_2$ cannot successfully cheat and we can efficiently mimic the outputs of $\pi^\circ_{\psi_i}$ by knowing the secret key sk, we can convert $\mathcal{A}_2$ to the adversary $\mathcal{A}_3$ against $\mathcal{E}_0\langle \pi^\circ_{\mathsf{ts}} \rangle$. Since $\mathcal{E}_0\langle \pi^\circ_{\mathsf{ts}} \rangle$ is secure protocol, we can convert $\mathcal{A}_3$ into an efficient adversary against the ideal implementation of crypto-computing protocol.

To summarise, we can concurrently schedule CdsZk protocols together with a crypto-computing protocol provided that honest parties do no other computations during the setup phase π_{ts} which generates global parameters.

7 Conclusions

In brief, we showed how to build universally composable zero-knowledge protocols from dual-mode commitments and homomorphic encryption. As the standard dual-mode commitments are also based on homomorphic encryption, our construction can be based solely on homomorphic encryption.

We acknowledge that the CdsZk protocol is not round-optimal, as there are theoretical constructions for zero-knowledge in three rounds. However, these are not as efficient. Similarly, our protocols are not the best in terms of online communication, as there are zero-knowledge protocols with constant communication in the trusted setup model, e.g. [Gro10]. However, these protocols use stronger security assumptions (bilinear parings combined with unfalsifiable assumptions) and the size of public parameters is really large.

In terms of practical performance our protocols are comparable to the zero-knowledge protocols based on sigma protocols where the security against malicious verifiers is achieved by committing the challenge at the beginning of the protocol. These protocols are more rigid against dynamic change of security levels. The latter is a drawback in the covert model where parties randomly alter security levels during the computations to discourage cheating.

Protocols CdsMul and CdsZkMul have also many direct applications. First, crypto-computing protocols often use oblivious polynomial evaluation, in which a valid query consists of encryptions of $x, x^2, \ldots, x^k$. Second, multiplicative relations naturally occur in crypto-computing systems, see App F and G.

Finally, conditional disclosure of secrets (CDS) is often used for going beyond linearity in crypto-computing. For example, it is possible to securely evaluate

greater than predicate using CDS protocols [BK04,LL07]. As somewhat homomorphic encryption (SHE) remains additively homomorphic when we reach the multiplication limit, we can combine CDS with SHE to extend the set of functions computable with SHE without extending its multiplicative depth.

References

[AF90] Martín Abadi and Joan Feigenbaum. Secure circuit evaluation. *J. Cryptology*, 2(1):1–12, 1990.

[AIR01] Bill Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In *Proc. of EUROCRYPT 2001, volume 2045 of LNCS*, pages 119–135. Springer-Verlag, 2001.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF Formulas on Ciphertexts. In *Proc. of TCC 2005*, volume 3378 of *LNCS*, pages 325–342. Springer, 2005.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *Proc. of ITCS 2012*, pages 309–325, 2012.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *Proc. of ACM CCS*, pages 784–796. ACM, 2012.

[BK04] Ian F. Blake and Vladimir Kolesnikov. Strong conditional oblivious transfer and computing on intervals. In *Proc. of ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 515–529. Springer, 2004.

[Can01] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. of FOCS 2001*, pages 136–145. IEEE 2001.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Proc. of CRYPTO 1994*, volume 839 of *LNCS*, pages 174–187. Springer, 1994.

[CIK+01] Ran Canetti, Yuval Ishai, Ravi Kumar, Michael K. Reiter, Ronitt Rubinfeld, and Rebecca N. Wright. Selective private function evaluation with applications to private statistics. In *Proc. of PODC 2001*, pages 293–304. ACM, 2001.

[CR03] Ran Canetti and Tal Rabin. Universal composition with joint state. In *Proc. of CRYPTO 2003*, volume 2729 of *LNCS*, pages 265–281. Springer, 2003.

[DGKN09] Ivan Damgård, Martin Geisler, Mikkel Krøigaard, and Jesper Buus Nielsen. Asynchronous multiparty computation: Theory and implementation. In *Proc. of PKC 2009*, volume 5443 of *LNCS*, pages 160–179. Springer, 2009.

[DJ01] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *In Proc. of PKC 2001*, volume 1992 of *L*NCS, 119–136. Springer, 2001.

[DO10] Ivan Damgård and Claudio Orlandi. Multiparty computation for dishonest majority: From passive to active security at low cost. In *Proc. of CRYPTO 2010*, volume 6223 of *LNCS*, pages 558–576. Springer, 2010.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Proc. of CRYPTO 2012*, volume 7417 of *LNCS*, pages 643–662. Springer, 2012.

[EG85] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proc. of CRYPTO 1984*, volume 196 of *LNCS*, pages 10–18. Springer, 1985.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proc. of STOC 2009*, pages 169–178. ACM, 2009.
[GHS12] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In *Proc. of EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 465–482. Springer, 2012.
[GIKM98] Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. In *Proc. of STOC 1998*, pages 151–160, 1998. ACM.
[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.
[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.
[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive Zaps and New Techniques for NIZK. In *Proc. of CRYPTO 2006*, volume 4117 of *LNCS*, pages 97–111. Springer, 2006.
[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *Proc. of ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, 2010.
[IP07] Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. In *Proc. of TCC 2007*, volume 4392 of *LNCS*, pages 575–594. Springer, 2007.
[Lin03] Yehuda Lindell. General composition and universal composability in secure multi-party computation. In *Proc. of FOCS 2003*, pages 394–403, 2003.
[LL07] Sven Laur and Helger Lipmaa. A new protocol for conditional disclosure of secrets and its applications. In *Proc. of ACNS 2007*, volume 4521 of *LNCS*, pages 207–225. Springer, 2007.
[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proc. of EUROCRYPT 1999*, volume 1716 of LNCS, pages 223–238. Springer, 1999.
[SV11] Nigel P. Smart and Frederik Vercauteren. Fully Homomorphic SIMD Operations. *IACR Cryptology ePrint Archive*, 2011:133, 2011.
[SYY99] Tomas Sander, Adam L. Young, and Moti Yung. Non-Interactive Cryptocomputing For NC^1. In *Proc. of FOCS 1999*, pages 554–567. IEEE Computer Society, 1999.
[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations. In *Proc. of FOCS 1982*, pages 160–164. IEEE Computer Society, 1982.

A Maximal Capacity of Secure Encoding Functions

Theorem 5. *Assume that the message space of a cryptosystem is a cyclic group of size N. Then we can encode at most $\frac{\gamma}{1-\varepsilon}$ different secrets with ε-secure encoding where γ is smallest factor of N.*

Proof. W.l.o.g. we can assume that the message space is $\mathbb{Z}_N$, as we are not interested in the computational efficiency in this proof. Let $\phi = N/\gamma$ denote the biggest non-trivial factor of N. Let $\mathcal{E}_s$ denote the set of all possible encodings of $s \in \mathcal{S}$ and let $\mathcal{Z}_s = \{e \mod \gamma : e \in \mathcal{E}_s\}$ denote of residues of $\mathcal{E}_s$ modulo γ.

First, lets consider the case where $|\mathcal{E}_s| = |\mathcal{Z}_s|$ for all $s \in \mathcal{S}$, i.e., for each $z \in \mathcal{Z}_s$ there exist only one $e \in \mathcal{E}_s$ such that $e \equiv z \pmod{\phi}$. Let us arrange all

elements of $\mathbb{Z}_N$ into a $\gamma \times \phi$ rectangular grid so that element e is in the column $e \mod \phi$ and each row is reserved for separate set $\mathcal{E}_s$. Clearly, we can fit up to γ sets $\mathcal{E}_s$ in such way but there could be enough blank cells to fit more sets.

For the proof, we bound the number of extra sets we can squeeze into these blanks. Let $p_z = \Pr[\mathsf{encode}(s) \equiv z \pmod{\phi}]$ for some fixed s. Then ε-indistinguishability of $\mathsf{encode}(s) + \phi\mathbb{Z}_N$ and $\mathsf{encode}(t) + \phi\mathbb{Z}_N$ implies that $\mathsf{encode}(s) \mod \phi$ and $\mathsf{encode}(t) \mod \phi$ are also ε-indistinguishable. Thus, the total probability of free cells in a single row $\alpha_s = \sum_{z \notin \mathcal{Z}_s} p_z \leq \varepsilon$ and the probability of covered cells $\beta_s = \sum_{e \in \mathcal{E}_s} p_{e \mod \phi} \geq 1 - \varepsilon$ for any additional set. Thus, the number of additional sets is bounded from above by $\gamma\varepsilon/(1-\varepsilon)$, as the total sum of probabilities captured by free sells is at most $\gamma\varepsilon$. The claim follows.

In general, some sets $\mathcal{E}_s$ may contain pairs such that $e_1 \not\equiv e_2 \pmod{N}$ but $e_1 \equiv e_2 \pmod{\phi}$. However, we can get rid of these pairs by defining a new encoding $\mathsf{encode}^*(\cdot)$ which transfers the occurrence probability of e_2 to e_1 so that the distribution of residues modulo ϕ does not change. Hence, the ε-indistinguishability of $\mathsf{encode}^*(s) \mod \phi$ and $\mathsf{encode}^*(t) \mod \phi$ still holds and any new encoding sets $\mathcal{E}_s^*$ is contained in $\mathcal{E}_s$. Now it is straightforward to use the same argument as before to get the same bound on the number of disjoint encoding sets $\mathcal{E}_s^*$ and thus also the size of $\mathcal{S}$. □

B Multi-party Protocol for Trusted Setup

Setting up public and secret key pairs for the cryptosystem is easy. Parties can generate keys by themselves an then use standard techniques to prove the knowledge of their secret key. Setting up the commitment key is bit more difficult, as we must make sure that nobody can learn the trapdoor.

If commitments based on additively homomorphic encryption as in [GOS06], we can use protocols for generating public key such that both parties share the secret key. After that, parties can separately generate two encryptions of zeroes and combine them into the commitment key $e \leftarrow \mathsf{enc}_{\mathsf{pk}}(1; r_0) \cdot \mathsf{enc}_{\mathsf{pk}}(0) \cdot \mathsf{enc}(0)$ for a publicly fixed r_0. To avoid cheating both parities must prove that they indeed submitted an encryption of zero using standard zero-knowledge proofs.

Thus, the final issue is the construction of a valid public key. For the Paillier cryptosystem, the generation of the RSA moduli is a well-researched problem with some good solutions. For the lifted ElGamal, the problem is even simpler as the public key is just a random group element. Hence, it is sufficient if both parties generate their own random elements, prove that they know the exponent, and finally use secure multiplication to obtain the public key as a multiple.

C Composability in the Standard Model

The proof of Theorem 3 hinges on the fact that simulation construction depicted in Figure. 3 works without rewinding. Let us now consider the standard model setting, where each party $\mathcal{P}_i$ generates a commitment key of a perfectly binding commitment, which is shared through all sessions of CDSZK proof where $\mathcal{P}_i$ acts

as a prover. In this setting, we must extract the randomness of the verifier by rewinding. If the CDSZK protocol is executed in isolation, i.e., all side computations are suspended, we can use standard knowledge-extraction techniques. For clarity, we omit all terms in time-bounds that are constant w.r.t. varying t.

Lemma 4. *Assume that a commitment scheme is (κ, t)-hiding and the* CDSZK *protocol is executed in isolation. Then there exists a $\mathcal{O}\big(\frac{t \log(1/\delta)}{\varepsilon - \kappa}\big)$-time knowledge extractor which fails with probability at most δ provided that a t-time verifier $\mathcal{V}_*$ passes the check $\boldsymbol{a} = \mathsf{answer}_{\mathsf{pk}}(\boldsymbol{q}, s; r)$ with probability at least ε.*

Proof (Sketch). The knowledge extraction strategy is straightforward. Simulate the provers first reply by sending a commitment to zero. Since the overall running time of the malicious verifier $\mathcal{V}^*$ is below t the probability that $\mathcal{V}_*$ releases the correct randomness (s, r) can decrease by κ. Markov inequality assures that the probability that $\mathcal{V}_*$ has not released correct randomness (s, r) after $\frac{2}{\varepsilon - \kappa}$ rewindings is at most $\frac{1}{2}$. Thus, after $\log(1/\delta) \cdot \frac{2}{\varepsilon - \kappa}$ rewindings the failure probability must be below δ. □

Corollary 1. CDSZK *protocols are secure in the standalone model in the sense that, for any desired failure probability ε, we can construct a simulator that runs in time $\mathcal{O}\big(\frac{t \log(1/\varepsilon)}{\varepsilon - \kappa}\big)$ and fails the simulation with probability ε.*

Proof (Sketch). For the proof, we modify the simulator constructor in in Figure. 3 by replacing the equivocal commitment with a knowledge extractor from Lemma 4. That is, the simulator first rewinds $\mathcal{V}_*$ to get a correct randomness (s, r) and in case of success commits to s as expected. In case of a knowledge-extraction failure, the simulator commits zero and hopes for the best. Let $\varepsilon(\boldsymbol{a})$ denote the probability that $\mathcal{V}_*$ gives a correct randomness after the commitment from honest prover $\mathcal{P}$. Then by doing $\mathcal{O}\big(\frac{\log(1/\varepsilon)}{\varepsilon - \kappa}\big)$ rewindings we can assure that (s, r) is recovered with probability ε whenever $\varepsilon(\boldsymbol{a}) \geq \varepsilon$. The cases where $\varepsilon(\boldsymbol{a}) < \varepsilon$ do not matter, since for these cases the simulator has to open the commitment with probability at most ε and thus rarely fails. □

Clarification. Security guarantees given by Corollary 1 are comparable with other proofs relying on black-box knowledge extraction techniques. In the framework of asymptotic security, this result guarantees security in the weak polynomial security model. That is, we can choose any positive power $c > 0$ such that the simulation failure decreases asymptotically $\mathcal{O}(k^{-c})$ w.r.t. to the security parameter k, while the simulator construction is still polynomial. However, we cannot find a single polynomial-time simulator such that simulation failure is asymptotically negligible, i.e., $\mathcal{O}(k^{-\omega(1)})$. To hide this fact, one often considers asymptotic of expected running time instead strict running-time, as the latter formally allows to achieve negligible simulation error.

Corollary 2. CDSZK *protocols are sequentially self-composable in the standard model provided that validity of public keys can be assured.*

Proof (Sketch). The proof is based on the observation that we can simulate each protocol instance similarly to Corollary 1 and all these simulators are sequentially executed. As consequence, it is straightforward to show that failures sum up linearly. As a small but important detail, note that if a malicious prover manages to use invalid public key, then the server-privacy is not guaranteed in the initial CDS phase of the proof and we cannot establish soundness guarantees. Therefore, we need public verifiability of public keys or parties must execute a dedicated zero-knowledge protocol for proving correctness of public keys before executing any of CdsZk protocols. □

Security of other composition types. The simulation construction sketched through Lemma 4 and Corollary 1 can be generalised for more complex settings. More precisely, the knowledge-extraction is guaranteed to work as long as we can efficiently simulate the actions of honest prover until the verifier releases the randomness. In particular, parallel execution of many CdsZk protocols is secure, as we can simulate commitments to all secrets at the same time.

However, not all all self-compositions lead to an efficient simulation. As a concrete example, consider a case where a CdsZk instance is embedded between the commitment and release message of another protocol instance. As the messages of the innermost protocol are not simulatable without rewinding, one must first extract s from it and then span outwards. If such an embedding is done for all ℓ protocols, we quickly reach exponential number of rewindings. Hence, only *well-aligned* concurrent compositions, where no answer-commit-release-decommit modules are in-lined between a commit-release module, are efficiently simulatable without dual commitments. This can be enforced by the prover, if it delays decommitments in case of potential conflicts.

The same restriction holds for other composition types, as well. As long as we can efficiently simulate all messages of a prover sent after the commitment and before the randomness release, the CdsZk protocol remains secure in this context. For instance, note that the asymmetric crypto-computing framework described in App. F has an interesting property that the server is prover in all CdsZk proofs and sends only encryptions to the client. As the client does not know the secret key, these encryptions can be simulated by encryption zeroes. Hence, CdsZk protocols can be arbitrarily scheduled as long as they remain well-aligned w.r.t. each other. The same claim holds also for the symmetric crypto-computing framework described in App. G.

D Equivocal Conditional Disclosure of Secrets

Computational soundness of the the CdsZk protocol does not follow from computational binding without extra assumptions on the CDS protocol. We say that a CDS protocol is (t_{eq}, ε)-*equivocal* if there exists a t_{eq}-time equivocation algorithm $\mathsf{equiv}_{\mathsf{sk}}$. More precisely, let r be the randomness space for the protocol. Then for every invalid query $\boldsymbol{q} \in \mathcal{Q}_{\mathrm{inv}}$, $s_0, s_1 \in \mathcal{S}$ and $r_0 \in \mathcal{R}$, we should get $r_1 \leftarrow \mathsf{equiv}_{\mathsf{sk}}(\boldsymbol{q}, s_0, s_1, r_0)$ such that $\mathsf{answer}_{\mathsf{pk}}(\boldsymbol{q}, s_0; r_0) = \mathsf{answer}_{\mathsf{pk}}(\boldsymbol{q}, s_1; r_1)$. The

algorithm might fail for some inputs, however, the distributions r_0 and r_1 must be statistically ε-close for any $\boldsymbol{q} \in \mathcal{Q}_{\mathrm{inv}}$ and $s_0, s_1 \in \mathcal{S}$.

Lemma 5. *Assume that the CDS protocol is (t_{eq}, ε)-equivocal. Let $\boldsymbol{a}_0$ denote a correct reply $\mathsf{answer}_{\mathsf{pk}}(\boldsymbol{q}, s_0, r_0)$ for $s_0 \leftarrow \mathcal{S}$ and randomness $r_0 \leftarrow \mathcal{R}$. Let r_1 denote the equivocation value $\mathsf{equiv}_{\mathsf{sk}}(\boldsymbol{q}, s_0, s_1, r_0)$ for $s_1 \leftarrow \mathcal{S}$. Then distributions $(\boldsymbol{a}_0, s_0, r_0)$ and $(\boldsymbol{a}_0, s_1, r_1)$ are ε-close for any $\boldsymbol{q} \in \mathcal{Q}_{\mathrm{inv}}$.*

Proof. Fix an arbitrary $\boldsymbol{q} \in \mathcal{Q}_{\mathrm{inv}}$. By definition the distributions of r_0 and r_1 are statistically ε-close for any values of s_0 and s_1. Consequently, the statistical distance between the distributions (s_0, r_0) and (s_1, r_1) is bounded by ε, since s_0 and s_1 are identically distributed. Let $\boldsymbol{a}_1 = \mathsf{answer}_{\mathsf{pk}}(\boldsymbol{q}, s_1, r_1)$. Then distributions $(\boldsymbol{a}_0, s_0, r_0)$ and $(\boldsymbol{a}_1, s_1, r_1)$ also ε-close, as the first element can be computed from the other two. The definition of equivocality also implies that $\boldsymbol{a}_0 \neq \boldsymbol{a}_1$ only if $r_1 = \perp$. Since triples with $r_1 = \perp$ are clearly distinguishable from elements of $(\boldsymbol{a}_0, s_0, r_0)$, the statistical distance does not change if we consider the distribution $(\boldsymbol{a}_0, s_1, r_1)$ instead of $(\boldsymbol{a}_1, s_1, r_1)$. Hence, we have proved that the distributions $(\boldsymbol{a}_0, s_0, r_0)$ and $(\boldsymbol{a}_0, s_1, r_1)$ are ε-close. □

Hence, we can consistently lie about underlying secret value whenever the prover submits $\boldsymbol{q} \in \mathcal{Q}_{\mathrm{inv}}$. The latter provides an efficient way to get double openings and allows us to reduce soundness to computational binding.

Theorem 6. *If the commitment scheme is $(2t, \varepsilon_1)$-binding and CDS protocol is (t_{eq}, ε_2)-equivocal, then for any $(t - t_{eq})$-time prover $\mathcal{P}_*$:*

$$\Pr\left[\langle\!\langle \mathcal{P}_*, \mathcal{V} \rangle\!\rangle(\boldsymbol{c}) = 1 \wedge \boldsymbol{q} \in \mathcal{Q}_{\mathrm{inv}}\right] \leq \frac{1}{|\mathcal{S}|} + \sqrt{\varepsilon_1} + \varepsilon_2 \ .$$

Proof. For the sake of contradiction, assume that some adversarial algorithm $\mathcal{P}_*$ violates the claims. Then we can consider the following adversary $\mathcal{A}$ with hardwired $(\mathsf{sk}, \mathsf{pk})$ against the binding property.

1. Given ck as input, send ck, pk to $\mathcal{P}_*$ to simulate the trusted setup.
2. Run the message formation phase. Decrypt $\boldsymbol{q}$ and halt if $\boldsymbol{q} \notin \mathcal{Q}_{\mathrm{inv}}$.
3. Generate a reply $\boldsymbol{a} \leftarrow \mathsf{cds.answer}_{\mathsf{pk}}(\boldsymbol{q}, s_0; r_0)$ for $s_0 \leftarrow \mathcal{S}$ and $r_0 \leftarrow \mathcal{R}$.
4. Store the reply c. Release (s_0, r_0) to $\mathcal{P}_*$ and store the decommitment as d_0.
5. Obtain alternative opening $r_1 \leftarrow \mathsf{cds.equiv}_{\mathsf{sk}}(\boldsymbol{q}, s_0, s_1, r_0)$ for $s_1 \leftarrow \mathcal{S}$.
6. Rewind $\mathcal{P}_*$, release (s_1, r_1) instead and store the decommitment as d_1.
7. Output (c, d_0, d_1) as a double opening.

Let $\alpha_i(\mathsf{ck}, \boldsymbol{q}) = \Pr\left[\mathsf{open}_{\mathsf{pk}}(c, d_i) = s_i | \mathsf{ck}, \boldsymbol{q}\right]$ for $i \in \{0, 1\}$ denote the success probability of $\mathcal{P}_*$ in Steps 4 and 6. Since the CDS protocol is (t_{eq}, ε_2)-equivocal, Lemma 5 assures that $|\alpha_0(\mathsf{ck}, \boldsymbol{q}) - \alpha_1(\mathsf{ck}, \boldsymbol{q})| \leq \varepsilon_2$ for any $\boldsymbol{q} \in \mathcal{Q}_{\mathrm{inv}}$ and for any $\mathsf{ck} \leftarrow \mathsf{cs.gen}$. Now, let Dopen denote the double-opening event $\perp \neq \mathsf{open}_{\mathsf{ck}}(c, d_1) \neq \mathsf{open}_{\mathsf{ck}}(c, d_2) \neq \perp$ and $\kappa = \frac{1}{|\mathcal{S}|}$. Then

$$\begin{aligned} \Pr\left[\mathsf{Dopen} | \mathsf{ck}, \boldsymbol{q}\right] &\geq \alpha_0(\mathsf{ck}, \boldsymbol{q})(\alpha_1(\mathsf{ck}, \boldsymbol{q}) - \kappa) \\ &\geq \alpha_0(\mathsf{ck}, \boldsymbol{q})^2 - (\varepsilon_2 + \kappa) \cdot \alpha_0(\mathsf{ck}, \boldsymbol{q}) \ . \end{aligned}$$

As square is a convex-cup function, Jensen's inequality assures

$$\begin{aligned}\Pr\left[\mathsf{Dopen}\right] &\geq \sum_{\mathsf{ck},\boldsymbol{q}} \Pr\left[\mathsf{ck} \wedge \boldsymbol{q} \in \mathcal{Q}_{\mathrm{inv}}\right] \cdot \left(\alpha_0(\mathsf{ck},\boldsymbol{q})^2 - (\varepsilon_2+\kappa)\cdot\alpha_0(\mathsf{ck},\boldsymbol{q})\right) \\ &\geq \alpha^2 - (\varepsilon_2+\kappa)\cdot\alpha\end{aligned}$$

where $\alpha = \Pr\left[\langle\!\langle \mathcal{P}_* \mathcal{V} \rangle\!\rangle(\boldsymbol{c}) = 1 \wedge \boldsymbol{q} \in \mathcal{Q}_{\mathrm{inv}}\right]$ is average of $\alpha_0(\mathsf{ck},\boldsymbol{q})$. Under the assumptions of the theorem, $\mathcal{A}$ is at most $2t$-time adversary and thus

$$\alpha_0^2 - (\varepsilon_2+\kappa)\cdot\alpha_0 \leq \varepsilon_1 \ ,$$

which itself implies $\alpha_0 \leq \varepsilon_2 + \kappa + \sqrt{\varepsilon_1}$. The claim is proved. □

*****Remarks. First, equivocality is essential if we want to reduce soundness to computational binding. Indeed, note that Fig. 3 describes an adversary $\mathcal{A}$, which bypasses the check by equivocating the commitment to the revealed secret. As $\mathcal{A}$ releases decommitment only if the CDS reply $\boldsymbol{a}$ is correctly opened, we must equivocate $\boldsymbol{a}$ to get a double opening from $\mathcal{A}$. Thus, no black-box reductions can exist without employing equivocality of the CDS protocol.

Second, note that standard CDS protocols are not equivocal when we use ElGamal and Paillier cryptosystems, since one cannot efficiently extract randomness of a ciphertext even if the secret key is known.

Fortunately, Goldwasser-Micali cryptosystem [GM84] is a suitable cryptosystem for CDS that facilitates randomness extraction. For brevity, we provide a solution for equivocal disclose-if-zero protocol, as the remaining CDS constructions can be build according to the description given in [AIR01,LL07].

Recall that Goldwasser-Micali cryptosystem is additively homomorphic over $\mathbb{Z}_2$. The public key is a random quadratic non-residue y over $\mathbb{Z}_N$ and zero is encrypted as a random quadratic residue and one is encrypted as a random quadratic non-residue. Consequently, a valid reply to quadratic non-residue $\boldsymbol{q}$ is a random ciphertext $\boldsymbol{a}$ with two explanations $\boldsymbol{a} = \boldsymbol{q}^{e_0} r_0^2$ and $\boldsymbol{a} = \boldsymbol{q}^{e_1} y r_1^2$ where e_0 and e_1 are uniquely determined by the quadratic residuosity of $\boldsymbol{a}$. For the equivocation, we must find $r_0 = \sqrt{\boldsymbol{a}\boldsymbol{q}^{-e_0}}$ and $r_1 = \sqrt{\boldsymbol{a}\boldsymbol{q}^{-e_1}y^{-1}}$. As the knowledge of secret key provides an efficient way to find square roots, the corresponding equivocation algorithm is also efficient.

E Sigma Protocols for Standard Relations

Zero-testing protocol depicted in Figure 4 is the basic protocol for proving relations about additively homomorphic encryptions. Together with disjunctive and conjunctive proof constructions it is sufficient to construct a sigma protocol for any efficiently computable relation. Hence, it is a direct counterpart for disclose-if-zero CDS protocol. Testing affine relation can reduced to the zero-testing provided that all multipliers are public. Damgård-Jurik protocol for multiplicative relations [DJ01] that is comparable to CdsMul is depicted in Figure 5.

TRUSTED SETUP FOR ENCRYPTION. Trusted dealer runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{pkc.gen}$ and sends pk as a public key of $\mathcal{P}$ to everyone. The secret key sk is sent securely to $\mathcal{P}$.

PROVERS INPUT. The prover $\mathcal{P}$ knows r such that $c = \mathsf{enc}_{\mathsf{pk}}(0; r)$.

1. The prover $\mathcal{P}$ sends $c_1 \leftarrow \mathsf{enc}(0, r_1)$ for $r_1 \leftarrow \mathcal{R}_{\mathsf{pk}}$ to $\mathcal{V}$.
2. The verifier $\mathcal{V}$ pics $e \leftarrow \mathcal{M}_{\mathsf{pk}}$ and sends e to $\mathcal{V}$.
3. The prover $\mathcal{P}$ computes $r_2 \leftarrow r - er_1$ to $\mathcal{V}$ who verifies that $c_1^e \cdot \mathsf{enc}_{\mathsf{pk}}(0, r_2) = c_0$.

Figure4. Sigma protocol for zero testing.

F Crypto-computing in Asymmetric Setting

Let us consider a setting where the client knows the public key pk of the server and the server helps him or her to perform various operations on the ciphertexts. As the cryptosystem is additively homomorphic, the client can compute encryptions of affine functions without the help from the server. Hence, the client needs help only for computing $\mathsf{enc}_{\mathsf{pk}}(x_1x_2)$ from $\mathsf{enc}_{\mathsf{pk}}(x_1)$ and $\mathsf{enc}_{\mathsf{pk}}(x_2)$. In the semihonest model, the client can use blinding to shift the task back to the server [AF90,CIK$^+$01]. Namely, the client can generate two random masks $r_1, r_2 \leftarrow \mathcal{M}_{\mathsf{pk}}$ and send $\mathsf{enc}_{\mathsf{pk}}(x_1 + r_1)$ and $\mathsf{enc}_{\mathsf{pk}}(x_2 + r_2)$ to the server. Due to the homomorphic properties (see Eq. (1)), the client can compute $\mathsf{enc}_{\mathsf{pk}}(x_1x_2)$ from the servers reply $\mathsf{enc}_{\mathsf{pk}}((x_1 + r_1)(x_2 + r_2))$ as $\mathsf{enc}_{\mathsf{pk}}((x_1 + r_1)(x_2 + r_2))\mathsf{enc}_{\mathsf{pk}}(x_2 + r_2)^{-r_1}\mathsf{enc}_{\mathsf{pk}}(x_1)^{-r_2}$.

In the malicious model, the server must additionally prove that $\mathsf{enc}_{\mathsf{pk}}(x_1+r_1)$, $\mathsf{enc}_{\mathsf{pk}}(x_2+r_2)$, $\mathsf{enc}_{\mathsf{pk}}((x_1+r_1)(x_2+r_2))$ are in multiplicative relation. By using out new CDSMUL together with the CDSZK construction, we obtain security against

TRUSTED SETUP FOR ENCRYPTION. Trusted dealer runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{pkc.gen}$ and sends pk as a public key of $\mathcal{P}$ to everyone. The secret key sk is sent securely to $\mathcal{P}$.

PROVERS INPUT.
The prover $\mathcal{P}$ knows r_1, r_2, r_3 and x_1, x_2, x_3 such that $x_1x_2 = x_3$ and $c_i = \mathsf{enc}_{\mathsf{pk}}(x_i; r)$.

1. The prover $\mathcal{P}$ generates $m_1^* \leftarrow \mathcal{M}_{\mathsf{pk}}$, $r_1^*, r_2^* \leftarrow \mathcal{R}_{\mathsf{pk}}$.
   The prover $\mathcal{P}$ sends $c_1^* \leftarrow \mathsf{enc}_{\mathsf{pk}}(m_1^*, r_1^*)$ and $c_2^* \leftarrow \mathsf{enc}_{\mathsf{pk}}(m_1^*x_2, r_2^*)$ to $\mathcal{V}$.
2. The verifier $\mathcal{V}$ pics $e \leftarrow \mathcal{M}_{\mathsf{pk}}$ and sends e to $\mathcal{V}$.
3. The prover $\mathcal{P}$ sends back

$$\bar{m}_1 \leftarrow ex_1 + m_1^*, \quad \bar{r}_1 \leftarrow er_1 + r_1^*, \quad \bar{r}_2 \leftarrow \bar{m}_1 r_2 - (er_3 + r_2^*) \ .$$

   The verifier $\mathcal{V}$ verifies that $c_1^e c_1^* = \mathsf{enc}_{\mathsf{pk}}(\bar{m}_1, \bar{r}_1)$ and $c_2^{\bar{m}_1} \cdot (c_3^e \cdot c_2^*)^{-1} = \mathsf{enc}_{\mathsf{pk}}(0, \bar{r}_2)$.

Figure5. Sigma protocol for multiplicative relation.

malicious servers with a protocol that has roughly tripled communication and computation. The exact overhead depends on the cryptosystem.

Since addition and multiplication is enough to represent any function, the client can compute an encryption of any function. Hence, we are left with the problem how the server can decrypt the final output without learning the outcome. In the semi-honest model, the client can mask the final outcome by sending $\mathsf{enc}_{\mathsf{pk}}(y+r)$ for decryption. In the malicious model, the server can lie about decryption and must add correctness proof. There are many efficient protocols for specific cryptosystems. However, we can use also CdsZk protocol, since after obtaining $x \leftarrow \mathsf{dec}_{\mathsf{sk}}(c)$, the client can compute $\mathsf{enc}_{\mathsf{pk}}(-x)c$ which is $\mathsf{enc}_{\mathsf{pk}}(0)$ only if x was correctly computed.

G Crypto-computing in Symmetric Setting

Let us consider a symmetric setting where each party has a secret keys and the opponent knows their public key. As a party can decrypt only ciphertexts corresponding to his or her secret key, we can combine additive secret sharing with encryption to get verifiable xor-secret sharing. More formally, let pk_1 and pk_2 be the public keys of $\mathcal{P}_1$ and $\mathcal{P}_2$. Then a shared bit $[\![x]\!]$ consists of private shares $x_1 \oplus x_2 = x$ and their public commitments $\mathsf{enc}_{\mathsf{pk}_1}(x_1)$ and $\mathsf{enc}_{\mathsf{pk}_2}(x_2)$. To get a Turing complete share-computing system, we must implement secure addition and multiplication over $\mathbb{Z}_2$.

First, we describe how $\mathcal{P}_1$ can shares the secret x. The protocol for $\mathcal{P}_2$ is symmetric. For sharing, $\mathcal{P}_1$ chooses $x_2 \leftarrow \mathbb{Z}_2$, sets $x_1 \leftarrow x \oplus x_2$ and publishes x_2 and $\mathsf{enc}_{\mathsf{pk}_1}(x_1)$. Next, both parties jointly compute $\mathsf{enc}_{\mathsf{pk}_2}(x_2)$ to avoid cheating. To complete the sharing, $\mathcal{P}_1$ must show that $\mathsf{enc}_{\mathsf{pk}_1}(x_1)$ either encryption of zero or one. The latter can be with corresponding CdsZk proof, which reply $\boldsymbol{a}$ consist of two encryptions. See [LL07] for further details.

It is easy to compute shares $[\![x \oplus y]\!]$ securely from $[\![x]\!]$ and $[\![y]\!]$, since parties can locally compute $z_i = x_i \oplus y_i$ and publish $\mathsf{enc}_{\mathsf{pk}_1}(z_1)$ and $\mathsf{enc}_{\mathsf{pk}_2}(z_2)$. To prove the correctness of its computations, $\mathcal{P}_i$ must show that encryptions $\mathsf{enc}_{\mathsf{pk}_i}(x_i), \mathsf{enc}_{\mathsf{pk}_i}(y_i)$ and $\mathsf{enc}_{\mathsf{pk}_i}(z_i)$ satisfy the relation $2x_iy_i = x_i + y_i - z_i$. As the encryption for the right hand side can be computed by the opponent, the proof can be done with a single invocation of CdsZkMul protocol.

For the multiplication protocol note that $xy = x_1y_1 \oplus x_1y_2 \oplus x_2y_1 \oplus x_2y_2$ where all multiplications are done over integers. As terms x_iy_i are locally computable, it is straightforward to create a valid sharings $[\![x_1y_1]\!]$ and $[\![x_2y_2]\!]$. Let $z_1 \oplus z_2 = x_iy_i$. Then $\mathcal{P}_i$ must prove the relation $x_iy_i = z_1 + z_2 - 2z_1z_2$. Again, we need only a single CdsZkMul protocol, since the opponent can compute the the right hand side. Xor-sharing for the terms x_iy_j requires cooperation. For brevity, we consider the term x_1y_2 as the treatment of the second term is symmetrical. In this case, $\mathcal{P}_2$ can generate $z_2 \leftarrow \mathbb{Z}_2$ and then send

$$\mathsf{enc}_{\mathsf{pk}_1}(x_1)^{y_2}\mathsf{enc}_{\mathsf{pk}_1}(z_2)\mathsf{enc}_{\mathsf{pk}_1}(x_1)^{-2y_2z_2} \equiv \mathsf{enc}_{\mathsf{pk}_i}(x_1y_2 \oplus z_2) \equiv \mathsf{enc}_{\mathsf{pk}_1}(z_1)$$

to $\mathcal{P}_1$ who decrypts it to get z_1. Finally, $\mathcal{P}_2$ publishes $\mathsf{enc}_{\mathsf{pk}_2}(z_2)$.

To assure correctness, parties must prove that $x_1 y_2 = z_1 + z_2 - 2z_1 z_2$. Although, the latter is multiplicative relation, we cannot use CDSZK protocols, as published encryptions $\mathsf{enc}_{\mathsf{pk}_1}(x_1), \mathsf{enc}_{\mathsf{pk}_1}(z_1)$ and $\mathsf{enc}_{\mathsf{pk}_2}(y_2), \mathsf{enc}_{\mathsf{pk}_2}(z_2)$ are under different keys. Thus, we must use standard zero-knowledge proofs to complete the multiplication protocol.