Compressed Σ-protocol theory

Attema, T.

CHAPTER 7

7

Applications of Compressed Σ-Protocols

7.1 Introduction

The primary functionality of compressed Σ-protocols is to prove knowledge of an opening to one or several compact commitments satisfying a linear constraint. In Chapter 4, in order to handle certain *nonlinear* relations, this functionality was enhanced. First, it was shown how to commit to a long vector of multiplication triples $(\alpha_i, \beta_i, \gamma_i = \alpha_i\beta_i)$ and prove that the committed vector satisfies the corresponding multiplicative relation. Second, a proof of partial knowledge technique was presented, allowing a prover to prove knowledge of k-out-of-n homomorphism preimages. The enhancements, required to handle these two nonlinear scenarios, can be viewed as linearization techniques; in both cases the nonlinear relation is reduced to a linear relation amenable for basic compressed Σ-protocols.

In this chapter, we present two applications of the basic compressed Σ-protocols together with these higher level (nonlinear) functionalities. First, in Section 7.2, we show how to prove *arbitrary* constraints, captured by an arithmetic circuit, on committed vectors. More precisely, we show how to prove that a committed vector $\mathbf{x} \in \mathbb{Z}_q^n$ satisfies the constraint $C(\mathbf{x}) = 0$ for some public arithmetic circuit $C\colon \mathbb{Z}_q^n \to \mathbb{Z}_q^s$. Protocols with this functionality are also referred to as *circuit zero-knowledge protocols*. It turns out that, by deploying the linearization techniques of Section 4.2, we only need *black-box* access to the basic functionality for opening linear forms. This explains also why compressed Σ-protocols do not need any *direct* provision to handle nonlinearity. Further, the number of black-box calls to this basic functionality is constant. Therefore, the (poly)logarithmic communication complexity is directly inherited when proving arbitrary constraints on committed vectors. Section 7.2 is based on the article [AC20], co-authored by Ronald Cramer.

Second, in Section 7.3, we combine the proofs of partial knowledge with an appropriate signature scheme to construct a *threshold signature scheme* (TSS). A k-out-of-n TSS is a standard signature scheme, allowing each of the n players to individually sign arbitrary messages m, enriched with a public k-aggregation algorithm. The k-aggregation algorithm takes as input k signatures, issued by any k distinct players, on the same message m and outputs a *threshold signature*. A TSS

is designed such that no adversary holding strictly less than k distinct signatures on a given message m can issue a valid threshold signature on this message. A naive TSS is obtained by exhibiting the k individual signatures directly. However, this approach results in threshold signatures with size linear in the threshold k. The main goal for TSSs is to have *succinct* threshold signatures, i.e., with size sublinear in k and n. The succinct TSS of [Sho00] immediately found an application in reducing the communication complexity of consensus protocols [CKS00; CKS05], this application was revived recently [LM18; AMS19; YMR+19; ADD+19]. The impact of succinctness is significant since, in consensus applications, the threshold k is of the same order of magnitude as n (typically $k = n/2$ or $k = 2n/3$). Although desirable in some applications, it is not required that a threshold signature *hides* the k-subset of signers. We construct a succinct TSS that has this additional security property, i.e., threshold signatures do not reveal any information about the k-subset of players that supplied valid signatures to the aggregation algorithm. Section 7.3 is based on the article [ACR21], co-authored by Ronald Cramer and Matthieu Rambaud.

7.2 Circuit Zero-Knowledge Protocols

First, in Section 7.2.1, we describe the compressed Σ-protocol for basic circuit satisfiability. This protocol allows a prover to commit to an input $\mathbf{x}$ and subsequently prove that the committed input $\mathbf{x}$ satisfies the constraint $C(\mathbf{x}) = 0$ for an arbitrary, but fixed, arithmetic circuit C. In practice, it may happen that the prover is already committed to the secret $\mathbf{x}$ *before* receiving the circuit C. This is referred to as the "commit-and-prove" scenario. In order to deal with this scenario, we need some further utility enhancements. The required enhancements are described in Section 7.2.2. Finally, in Section 7.2.3, we describe a generalization from arithmetic circuits to bilinear group arithmetic circuits.

7.2.1 The Compressed Σ-Protocol for Arithmetic Circuits

Suppose $C\colon \mathbb{Z}_q^n \to \mathbb{Z}_q^s$ is an arithmetic circuit with n inputs, s outputs and m multiplication gates. We only count multiplication gates with *variable* inputs; additions and multiplications by constants are implicitly handled and immaterial to the communication costs. We can easily turn our approach for proving correctness of multiplication triples into a solution for "circuit zero-knowledge," i.e., the prover convinces the verifier it knows an input $\mathbf{x} \in \mathbb{Z}_q^n$ for which the circuit C, without loss of generality, returns 0. We note that [CDP12] also gives a solution for circuit zero-knowledge based on linearizing multiplication triples. But that solution has a communication complexity that is linear in the size of the circuit C. We aim for a (poly)logarithmic communication complexity, so we make some changes.

The protocol goes as follows. The prover first determines the computation graph implied by instantiating the circuit C with its input vector $\mathbf{x}$. In this graph every wire is assigned a value in $\mathbb{Z}_q$. In particular, let $\alpha_1, \ldots, \alpha_m \in \mathbb{Z}_q$ be the left inputs, $\beta_1, \ldots, \beta_m \in \mathbb{Z}_q$ the right inputs and $\gamma_1, \ldots, \gamma_m \in \mathbb{Z}_q$ the outputs of the m multiplication gates in this computation graph. Hence, each $(\alpha_i, \beta_i, \gamma_i) \in \mathbb{Z}_q^3$ is a multiplication triple.

Let us now use the following simple fact about arithmetic circuits. For each i, there are affine forms[1] $u_i, v_i \colon \mathbb{Z}_q^{n+m} \to \mathbb{Z}_q$, depending only on C, such that, for all $\mathbf{x} \in \mathbb{Z}_q^n$, it holds that $\alpha_i = u_i(\mathbf{x}, \gamma_1, \ldots, \gamma_m)$ and $\beta_i = v_i(\mathbf{x}, \gamma_1, \ldots, \gamma_m)$. These forms are uniquely determined by the addition and scalar multiplication gates. In other words, a given vector $(\mathbf{x}, \gamma_1, \ldots, \gamma_m) \in \mathbb{Z}_q^{n+m}$ can be completed to a valid computation graph if and only if

$$u_i(\mathbf{x}, \gamma_1, \ldots, \gamma_m) \cdot v_i(\mathbf{x}, \gamma_1, \ldots, \gamma_m) = \gamma_i \,,$$

for all $1 \leq i \leq m$. Hence, checking whether $(\mathbf{x}, \gamma_1, \ldots, \gamma_m)$ corresponds to a valid computation graph amounts to verifying the multiplication triples defined by the γ_i's and the public linear forms u_i and v_i. This verification can be performed by deploying the arithmetic secret-sharing based linearization technique for multiplication triples of Section 4.2. Further, there are affine forms $w_j \colon \mathbb{Z}_q^{n+m} \to \mathbb{Z}_q$ corresponding to the s output gates of C. Hence, the evaluation $C(\mathbf{x})$ returns 0 if and only if $w_j(\mathbf{x}, \gamma_1, \ldots, \gamma_m) = 0$ for all $1 \leq j \leq s$. Recall that s is the dimension of the codomain of the circuit $C \colon \mathbb{Z}_q^n \to \mathbb{Z}_q^s$.

Altogether, after computing the above computation graph, the circuit satisfiability protocol therefore proceeds as follows. As in Section 4.2, the prover selects a random polynomial $f(X)$ of degree at most m that defines a packed secret sharing of the vector $(\alpha_1, \ldots, \alpha_m)$ of left inputs to the multiplication gates. The prover also selects a random polynomial $g(X)$ of degree at most m that defines a packed secret sharing of the vector $(\beta_1, \ldots, \beta_m)$ of right inputs to the multiplication gates. Finally, the prover computes the product polynomial $h(X) := f(X)g(X)$ of degree at most $2m < q$.

The prover commits to each coordinate of $\mathbf{x}$ and to the *auxiliary data*

$$\mathsf{aux} = (f(0), g(0), h(0), h(1), \ldots, h(2m)) \in \mathbb{Z}_q^{2m+3}$$

in one single compact commitment. The length of the committed vector $\mathbf{y} = (\mathbf{x}, \mathsf{aux})$ thus equals $n + 2m + 3$. Note that the vector $\mathbf{y}$ contains the outputs $\gamma_1 = h(1), \ldots, \gamma_m = h(m)$ of the multiplication gates of C evaluated on $\mathbf{x}$. However, it does not necessarily contain the inputs $\alpha_1, \beta_1, \ldots, \alpha_m, \beta_m$ of these multiplication gates. These inputs are namely affine combinations of the coefficients of $\mathbf{y}$. This explains why it is not necessary to commit explicitly to the α_i's and the β_i's as these are now implicitly committed to via said affine forms evaluated on $\mathbf{y}$. Therefore, since the values $f(0)$ and $g(0)$ are still included in $\mathbf{y}$, the polynomials $f(X)$, $g(X)$ and $h(X)$ are well defined by $\mathbf{y}$, and their evaluations are, by composition of the appropriate maps, also affine evaluations on $\mathbf{y}$. What remains is to check that the polynomial $h(X)$ is indeed the product of $f(X)$ and $g(X)$.

Therefore, with the above observations at hand, the circuit zero-knowledge protocol is reduced to opening the affine forms that, on input $\mathbf{y}$, output $\big(C(\mathbf{x}), f(c), g(c), h(c)\big) \in \mathbb{Z}_q^{s+3}$ for a challenge $c \leftarrow_R \mathbb{Z}_q \setminus \{1, \ldots, m\}$ sampled uniformly at random by the verifier. First, the verifier checks that $h(c) = f(c)g(c)$,

[1] Recall that an affine form $A \colon \mathbb{Z}_q^n \to \mathbb{Z}_q$ is a linear form L plus a constant $a \in \mathbb{Z}_q$. Hence, opening an affine form $A = L + a$ amounts to opening the linear form L and adding the (public) constant a.

which, as in Section 4.2, shows that $h(X) = f(X)g(X)$ holds with high probability. Second, the verifier checks that $C(\mathbf{x}) = 0$, which shows that the circuit is satisfiable and that the prover knows a witness $\mathbf{x}$.

This approach thus reduces the nonlinear circuit satisfiability relation to opening a constant number of affine forms on a compactly committed vector, it is therefore again a *linearization* technique. The compressed Σ-protocol for circuit satisfiability thus consists of two main building blocks: (1) the linearization technique and (2) a compressed Σ-protocol for opening linear forms.

The linearization technique itself can be presented as an interactive proof for the circuit satisfiability relation

$$\mathfrak{R}_{\mathsf{CS}} = \{(C; \mathbf{x}) : C(\mathbf{x}) = 0\}\,.$$

This interactive proof is composable with a basic compressed Σ-protocol for opening linear forms, allowing its linear communication complexity to be reduced. A formal description can be found in Protocol 16. To simplify the exposition we consider an abstract compact vector commitment scheme

$$[\cdot]\colon \bigcup_{\ell\in\mathbb{N}} \mathbb{Z}_q^\ell \to \mathbb{H}$$

that allows a prover to commit to arbitrary length vectors $\mathbf{x} \in \bigcup_{\ell\in\mathbb{N}} \mathbb{Z}_q^\ell$ in a single group element $P \in \mathbb{H}$. In this notation, we leave the commitment randomness implicit, i.e., $[\mathbf{x}]$ denotes a commitment to the vector $\mathbf{x}$.

As a stand-alone building block, the interactive proof described in Protocol 16 might seem pointless, as the prover's final message $(\mathbf{y}, u, v, w)$ contains the witness $\mathbf{x}$. Hence, it is clearly not zero-knowledge and it is less efficient than a trivial interactive proof that simply reveals the witness $\mathbf{x}$. However, the key point is that the long vector $\mathbf{y}$ can be viewed as an interactive proof for opening linear forms. This long vector, dominating the communication costs, can therefore be replaced by a basic compressed Σ-protocol. Thus, Protocol 16 indeed linearizes the nonlinear circuit satisfiability relation, making it amenable for basic compressed Σ-protocols. Below we describe the properties of this composition, but Theorem 7.1 first summarizes the main properties of the stand-alone linearization technique for circuit satisfiability. It shows that this technique is a perfectly complete and $(2m+1)$-out-of-$(q-m)$ special-sound Σ-protocol.

Theorem 7.1 (Linearization for Circuit Satisfiability)**.** *Let $n, m, s \in \mathbb{N}$, $q > 3m$ a prime, $[\cdot]\colon \bigcup_{\ell\in\mathbb{N}} \mathbb{Z}_q^\ell \to \mathbb{H}$ a homomorphic vector commitment scheme and $C\colon \mathbb{Z}_q^n \to \mathbb{Z}_q^s$ an arithmetic circuit with m multiplication gates. The Σ-protocol for relation*

$$\mathfrak{R}_{\mathsf{CS}} = \{(C; \mathbf{x}) : C(\mathbf{x}) = 0\}\,,$$

described in Protocol 16, is perfectly complete, and $(2m+1)$-out-of-$(q-m)$ special-sound, under the assumption that the commitment scheme is binding.

Proof. **Completeness:** This property follows immediately.

Protocol 16 Linearization of the Circuit Satisfiability Relation.

Parameters:	$n, m, s \in \mathbb{N}$, prime $q > 3m$, group $(\mathbb{H}, \cdot)$ with exponent q and homomorphic vector commitment scheme $[\cdot]\colon \bigcup_{\ell \in \mathbb{N}} \mathbb{Z}_q^\ell \to \mathbb{H}$
Public Input:	circuit $C\colon \mathbb{Z}_q^n \to \mathbb{Z}_q^s$ with m multiplication gates
Prover's Private Input:	$\mathbf{x} \in \mathbb{Z}_q^n$ and, for $1 \leq i \leq m$, (α_i, β_i) denote the left and right inputs to the multiplication gates of the circuit C evaluated in $\mathbf{x}$
Prover's Claim:	$C(\mathbf{x}) = 0$

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
$f(X), g(X) \leftarrow_R \mathbb{Z}_q[X]$ s.t. $\deg(f) \leq m \ \wedge \ \deg(g) \leq m$ $\wedge\ f(i) = \alpha_i \ \wedge \ g(i) = \beta_i \ \forall i$ $\wedge\ h(X) = f(X)g(X)$		
$\mathbf{y} = \big(\mathbf{x}, f(0), g(0), h(0), \ldots, h(2m)\big)$ $P = [\mathbf{y}]$		
	$\xrightarrow{P}$	
		$c \leftarrow_R \mathbb{Z}_q \setminus \{1, \ldots, m\}$
	$\xleftarrow{c}$	
$u = f(c)$ $v = g(c)$ $w = h(c)$		
	$\xrightarrow{\mathbf{y}, u, v, w}$	$u \cdot v \stackrel{?}{=} w$
		$[\mathbf{y}] \stackrel{?}{=} P$ $A_C(\mathbf{y}) \stackrel{?}{=} 0\,,\ f_{\mathbf{y}}(c) \stackrel{?}{=} u$ $g_{\mathbf{y}}(c) \stackrel{?}{=} v\,,\ h_{\mathbf{y}}(c) \stackrel{?}{=} w$

Here, $A_C\colon \mathbb{Z}_q^{n+2m+3} \to \mathbb{Z}_q^s$ is the affine mapping that, on input the vector $\mathbf{y}$ containing the secret input $\mathbf{x} \in \mathbb{Z}_q^n$ and the outputs of the multiplication gates of C evaluated in $\mathbf{x}$, outputs $C(\mathbf{x}) \in \mathbb{Z}_q^s$. Further, $f_{\mathbf{y}}(X)$, $g_{\mathbf{y}}(X)$ and $h_{\mathbf{y}}(X)$ are the polynomials defined by the vector $\mathbf{y}$ and the circuit C. They correspond to the polynomials $f(X)$, $g(X)$ and $h(X)$ constructed by an *honest* prover.

Special-Soundness: Let

$$(P, c_0, \mathbf{y}_0, u_0, v_0, w_0)\,, \ \ldots\ , (P, c_{2m}, \mathbf{y}_{2m}, u_{2m}, v_{2m}, w_{2m})$$

be $2m + 1$ accepting transcripts with common first message P and pairwise distinct challenges $c_j \in \mathbb{Z}_q \setminus \{1, \ldots, m\}$. Then, under the assumption that the commitment scheme $[\cdot]$ is binding, it follows that $\mathbf{y}_0 = \cdots = \mathbf{y}_{2m} = \mathbf{y}$,

and we may write $f(X) = f_{\mathbf{y}}(X)$, $g(X) = g_{\mathbf{y}}(X)$ and $h(X) = h_{\mathbf{y}}(X)$ for the three polynomials unique defined by $\mathbf{y}$.

Further, $\mathbf{y}$ corresponds to a wire value assignment of the circuit C. In particular, for $1 \leq i \leq m$, $\alpha_i = f(i)$, $\beta_i = g(i)$ and $\gamma_i = h(i)$ correspond to the values assigned to the left input, the right input and the output of the i-th multiplication gate in C. Moreover, since $A_C(\mathbf{y}) = 0$, the values assigned to the output wires are equal to 0. What remains is to verify that this wire value assignment is *valid*, i.e., for all gates the output should correspond to the appropriate combination of the input values.

The linear relations, defined by the addition and scalar multiplication gates, are automatically satisfied. Therefore, all that needs to be verified are the multiplicative relations $\alpha_i \cdot \beta_i = \gamma_i$. To this end, observe that, for all $0 \leq j \leq 2m$,

$$f(c_j) \cdot g(c_j) = u_j \cdot v_j = w_j = h(c_j)\,.$$

Since the polynomials $f(X)$ and $g(X)$ are of degree at most m and $h(X)$ is of degree at most $2m$, it follows that $f(X) \cdot g(X) = h(X)$. Hence,

$$\alpha_i \cdot \beta_i = f(i) \cdot g(i) = h(i) = \gamma_i$$

for all $1 \leq i \leq m$, which completes the proof.

□

The vector $\mathbf{y}$, sent in the final round of Protocol 16, is a trivial proof of knowledge for opening the $s + 3$ affine forms that return

$$\big(C(\mathbf{x}) = A_C(\mathbf{y}), f_{\mathbf{y}}(c), g_{\mathbf{y}}(c), h_{\mathbf{y}}(c)\big) \in \mathbb{Z}_q^{s+3}$$

on input $\mathbf{y}$. The compressed Σ-protocol for basic circuit satisfiability simply replaces this trivial interactive proof, i.e., the message $\mathbf{y}$, by a compressed Σ-protocol for opening the appropriate linear forms. Recall that the costs of these $s+3$ linear form openings can be amortized (Section 3.4.2), i.e., the amortized communication costs are independent of s.

The exact communication costs depend on the instantiation of the compact commitment scheme. For instance, the discrete logarithm based instantiation has logarithmic communication, while, due to soundness slack, the lattice based instantiation has polylogarithmic communication. For concreteness let us consider the discrete logarithm instantiation of Section 5.2. The following theorem summarizes the main properties of the discrete logarithm based circuit satisfiability protocol Π_{CS}. Note in particular that this compressed Σ-protocol has a communication complexity that is logarithmic in the dimension n of the input vector $\mathbf{x} \in \mathbb{Z}_q^n$ and the number of multiplication gates m. Moreover, since the Pedersen vector commitment scheme is unconditionally hiding, it is special honest-verifier zero-knowledge.

Theorem 7.2 (DL-Based Compressed Σ-Protocol for Circuit Satisfiability)**.** *Let q be a prime and $\mu, n, m, s \in \mathbb{N}$ such that $n + 2m + 4 = 2^\mu$ and $q > 3m$. Further, let* $\text{COM}\colon \mathbb{Z}_q^{n+2m+3} \times \mathbb{Z}_q \to \mathbb{H}$ *be the Pedersen vector commitment scheme.*

Then the compressed Σ*-protocol* Π_{CS} *for relation*

$$\mathfrak{R}_{\mathsf{CS}} = \{(C; \mathbf{x}) : C(\mathbf{x}) = 0\},$$

instantiated with the Pedersen commitment scheme COM*, is perfectly complete, computationally* $(2m+1, s+4, 3, \ldots, 3)$*-out-of-*$(q-m, q, \ldots, q)$ *special-sound, under the discrete logarithm assumption, and special honest-verifier zero-knowledge (SHVZK). Moreover, it has* $2\mu+5$ *communication rounds and the communication costs are:*

- $\mathcal{P} \to \mathcal{V}$*:* 5 *elements of* $\mathbb{Z}_q$ *and* 2μ *elements of* $\mathbb{H}$*;*
- $\mathcal{V} \to \mathcal{P}$*:* $\mu + 2$ *elements of* $\mathbb{Z}_q$*.*

Proof. Completeness and special-soundness follow directly from Lemma 3.1. This lemma describes the properties of the composition of interactive proofs. However, Lemma 3.1 only states that the composition $\Pi_b \diamond \Pi_a$ of interactive proofs is SHVZK if the interactive proof Π_a that is applied *first* is SHVZK.

In our case, the interactive proof that is applied first, the linearization of Protocol 16, is not SHVZK. So we need to use additional properties to prove that the compressed Σ-protocol for circuit satisfiability is SHVZK. It turns out that this property follows from the fact that the Pedersen vector commitment scheme is perfectly hiding and the deployed instantiation of Shamir's packed secret-sharing scheme has 1-privacy.

To see this, let us describe the SHVZK simulator. Assume that the circuit C admits an input $\mathbf{x}$ such that $C(\mathbf{x}) = 0$ and let $(\alpha_i, \beta_i, \gamma_i = \alpha_i\beta_i)$ denote the left input, right input and output values of the multiplication gates of C evaluated in $\mathbf{x}$. The simulator then proceeds as follows. First, it samples the challenges $c_1 \leftarrow_R \mathbb{Z}_q \setminus \{1, \ldots, m\}$ and $c_2, \ldots, c_{\mu+2} \leftarrow_R \mathbb{Z}_q$ uniformly at random. Second, it samples a Pedersen commitment $P \leftarrow_R \mathbb{H}$ and field elements $u, v \leftarrow_R \mathbb{Z}_q$ uniformly at random, and sets $w = u \cdot v$.

The first three messages P, c_1 and (u, v, w) of the circuit satisfiability protocol have now been simulated. The remaining messages are sampled by using the SHVZK simulator of the compressed Σ-protocol for opening linear forms. However, this is only possible when the commitment P admits an opening $(\mathbf{y}; \gamma)$ satisfying the appropriate linear relations.

To see that this is the case, note that, since $c_1 \notin \{1, \ldots, m\}$, there exist polynomials[2] $f, g \in \mathbb{Z}_q[X]$ of degree at most m such that $f(c_1) = u$ and $g(c_1) = v$, and $f(i) = \alpha_i$ and $g(i) = \beta_i$ for all $1 \leq i \leq m$. Let $h(X) = f(X) \cdot g(X)$.

Hence, there exists a vector $\mathbf{y} = \big(\mathbf{x}, f(0), g(0), h(0), \ldots, h(2m)\big)$ that satisfies the linear relations $A_C(\mathbf{y}) = 0$, $f_{\mathbf{y}}(c_1) = f(c_1) = u$, $g_{\mathbf{y}}(c_1) = g(c_1) = v$ and $h_{\mathbf{y}}(c_1) = h(c_1) = w$. Moreover, since the Pedersen vector commitment scheme is perfectly hiding, the random commitment P has an opening $(\mathbf{y}; \gamma)$ for some $\gamma \in \mathbb{Z}_q$. Hence, the compressed Σ-protocol for opening linear forms is instantiated with a statement P that admits a witness satisfying the appropriate linear constraints. Therefore, the simulator for our circuit satisfiability protocol can run the SHVZK simulator of the compressed Σ-protocol for opening linear forms to

[2] The existence of these polynomials follows from the 1-privacy of the secret-sharing scheme.

simulate the remaining messages. Again using the hiding property of the Pedersen vector commitment scheme and the 1-privacy of the secret-sharing scheme, it is easily seen that the simulated transcripts follow the same distribution as honestly generated ones. □

The terminology *circuit satisfiability* seems to suggest that we are only considering circuits for which it is hard to compute a satisfying witness $\mathbf{x}$, i.e., an $\mathbf{x}$ with $C(\mathbf{x}) = 0$. However, many practical scenarios consider circuits C for which it is easy to compute an $\mathbf{x}$ such that $C(\mathbf{x}) = 0$. In these scenarios the functionality offered by a circuit zero-knowledge protocol is still nontrivial. Namely, after evaluating the protocol, the prover has not only proven knowledge of a witness $\mathbf{x}$, but is also *committed* to this vector $\mathbf{x}$. Hence, in this case, a prover can show that a committed vector satisfies certain properties captured by the arithmetic circuit. These properties do not need to be captured by arithmetic circuits for which it is hard to compute an input evaluating to 0.

7.2.2 An Extension to *Commit-and-Prove* Protocols

In the previous section we treated the basic circuit satisfiability scenario, where a prover claims to know a satisfiable input $\mathbf{x} \in \mathbb{Z}_q^n$ such that $C(\mathbf{x}) = 0$ for some public arithmetic circuit $C\colon \mathbb{Z}_q^n \to \mathbb{Z}_q^s$. The compressed Σ-protocol Π_{CS} for basic circuit satisfiability requires the prover to commit to the input $\mathbf{x}$ and a vector of auxiliary data aux in a single compact commitment. However, in practice it is likely that the prover is *already* committed to the input $\mathbf{x}$ before the start of the protocol. Consider, for example, the following two extreme cases:

Case 1: The prover is committed to $\mathbf{x}$ in a *single* compact commitment.

Case 2: The prover is committed to the coordinates of $\mathbf{x}$ *individually*, i.e., each coordinate is committed to in a separate 1-dimensional commitment.

Besides these extreme cases one can consider hybrid scenarios in which the secret-vector-of-interest $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_s)$ is dispersed over s compact commitments to vectors $\mathbf{x}_i \in \mathbb{Z}_q^{n_i}$. We will focus on the two extreme cases, but hybrid scenarios can be handled similarly.

An interactive proof that allows a prover to prove statements about secret vectors that it is already committed to is called a *commit-and-prove* protocol. More precisely, given a compact vector commitment scheme $[\cdot]\colon \bigcup_{\ell\in\mathbb{N}} \mathbb{Z}_q^\ell \to \mathbb{H}$ (again leaving the commitment randomness implicit), a commit-and-prove protocol, for the case 1 scenario, is an interactive proof for relation

$$\mathfrak{R}^1_{\mathsf{CS}} = \{(P, C; \mathbf{x}) : [\mathbf{x}] = P \wedge C(\mathbf{x}) = 0\},$$

and a commit-and-prove protocol, for the case 2 scenario, is an interactive proof for relation

$$\mathfrak{R}^2_{\mathsf{CS}} = \{(P_1, \ldots, P_n, C; \mathbf{x}) : [x_i] = P_i \ \forall i \wedge C(\mathbf{x}) = 0\}.$$

In order to deal with these scenarios, we first need to bring about the desired starting point for the circuit satisfiability protocol of Section 7.2.1, i.e., the prover

needs to be committed to all coordinates of the input $\mathbf{x}$ and the required auxiliary information aux in a single compact commitment. Similar to Section 4.2.2, we handle the commit-and-prove scenario by deploying the compactification techniques of Section 3.4.4.

Let us first consider the case 1 commit-and-prove scenario. In this case, the basic circuit satisfiability protocol is adapted as follows.

- In the first round, instead of sending a compact commitment to the $n+2m+3$ dimensional vector $\mathbf{y} = (\mathbf{x}, \mathsf{aux})$, the prover sends a compact commitment Q to $(0, \mathsf{aux}) \in \mathbb{Z}_q^{n+2m+3}$ to the verifier.[3]
- Given the commitment $P = [\mathbf{x}] = [(\mathbf{x}, 0)]$ to the input vector $\mathbf{x}$ and the commitment $Q = [(0, \mathsf{aux})]$ to the auxiliary data, both the prover and verifier can compute a single compact commitment $P \cdot Q = [(\mathbf{x}, \mathsf{aux})]$ to all relevant data. This brings about the desired starting point for the circuit satisfiability protocol of Section 7.2.1.
- What remains is for the prover to show that the commitment Q is of the appropriate form. More precisely, Q should be a commitment to a vector with zeros in its first n coordinates. This boils down to opening the linear forms $L_i(\mathbf{y}) = y_i$, for $1 \leq i \leq n$, that return the first n coordinates of the vector $\mathbf{y}$. As before, the communication complexity of opening n different linear forms on the same commitment can be amortized.

The above shows how the case 1 commit-and-prove scenario is reduced to opening $s+3$ linear forms on the commitment $P \cdot Q = [(\mathbf{x}, \mathsf{aux})]$ and opening n linear forms on the commitment $Q = [(0, \mathsf{aux})]$. The naive approach of simply evaluating two (amortized) compressed Σ-protocols increases the communication costs with roughly a factor two, with respect to the basic circuit satisfiability protocol Π_{CS}. However, the factor two loss can be avoided by deploying the (case 1) compactification techniques of Section 3.4.4. Recall that compactification allows a prover to *compactify* relevant data that is dispersed over several commitments, into a single compact commitment. Phrased alternatively, compactification techniques allow a prover to open linear forms evaluated on inputs that are dispersed over several commitments.

The resulting compressed Σ-protocol, denoted by Π^1_{CS}, is an interactive proof for commit-and-prove relation $\mathfrak{R}^1_{\mathsf{CS}}$. Theorem 7.3 summarizes the main properties of the discrete logarithm instantiation of Π^1_{CS}, i.e., the instantiation using the Pedersen vector commitment scheme. The other instantiations of Chapter 5 work similarly, but may result in different communication complexities. Note that the communication costs of $\mathfrak{R}^1_{\mathsf{CS}}$ are roughly the same as those of the compressed Σ-protocol for basic circuit satisfiability.

Theorem 7.3 (DL-Based Case 1 Commit-and-Prove Protocol for Arithmetic Circuits)**.** *Let q be a prime and $\mu, n, m, s \in \mathbb{Z}_q$ such that $n+2m+6 = 2^\mu$ and $q > 3m$. Further, let* $\textsc{com}\colon \left(\bigcup_{\ell \in \mathbb{N}} \mathbb{Z}_q^\ell\right) \times \mathbb{Z}_q \to \mathbb{H}$ *the Pedersen vector commitment scheme and $C\colon \mathbb{Z}_q^n \to \mathbb{Z}_q^s$ an arithmetic circuit with m multiplication gates.*

[3] Here, $(0, \mathsf{aux})$ denotes the vector $\mathsf{aux} \in \mathbb{Z}_q^{2m+3}$ of auxiliary information prepended with n zeros.

Then the discrete logarithm based compressed Σ-protocol Π^1_{CS} *for relation*

$$\mathfrak{R}^1_{\mathsf{CS}} = \{(P, C; \mathbf{x}, \gamma) : \text{COM}(\mathbf{x}; \gamma) = P \wedge C(\mathbf{x}) = 0\}$$

is perfectly complete, computationally $(2m+1, \max(n+1, s+4), 2, 2, 3, \ldots, 3)$*-out-of-*$(q-m, q, q, q-1, q, \ldots, q)$ *special-sound, under the discrete logarithm assumption, and special honest-verifier zero-knowledge (SHVZK). Moreover, it has* $2\mu + 11$ *communication rounds and the communication costs are:*

- $\mathcal{P} \to \mathcal{V}$*:* 11 *elements of* $\mathbb{Z}_q$ *and* $2\mu + 4$ *elements of* $\mathbb{H}$*;*
- $\mathcal{V} \to \mathcal{P}$*:* $\mu + 5$ *elements of* $\mathbb{Z}_q$*.*

The case 2 commit-and-prove scenario is handled similarly. However, instead of the case 1 compactification techniques, we now deploy the case 2 compactification techniques of Section 3.4.4. The resulting protocol, denoted by Π^2_{CS}, is a compressed Σ-protocol for relation $\mathfrak{R}^2_{\mathsf{CS}}$. Theorem 7.4 summarizes the main properties of its discrete logarithm instantiation.

Theorem 7.4 (DL-Based Case 2 Commit-and-Prove Protocol for Arithmetic Circuits)**.** *Let* q *be a prime and* $\mu, n, m, s \in \mathbb{Z}_q$ *such that* $n + 2m + 5 = 2^\mu$ *and* $q > 3m$*. Further, let* COM: $\left(\bigcup_{\ell \in \mathbb{N}} \mathbb{Z}_q^\ell\right) \times \mathbb{Z}_q \to \mathbb{H}$ *be the Pedersen vector commitment scheme and* $C \colon \mathbb{Z}_q^n \to \mathbb{Z}_q^s$ *an arithmetic circuit with* m *multiplication gates.*

Then the discrete logarithm based compressed Σ-protocol Π^1_{CS} *for relation*

$$\mathfrak{R}^1_{\mathsf{CS}} = \{(P_1, \ldots, P_n, C; \mathbf{x}, \gamma_1, \ldots, \gamma_n) : \text{COM}(x_i; \gamma_i) = P_i \ \forall i \wedge C(\mathbf{x}) = 0\}\,,$$

is perfectly complete, computationally $(2m+1, n+1, s+5, 2, 3, \ldots, 3)$*-out-of-*$(q-m, q, \ldots, q)$ *special-sound, under the discrete logarithm assumption, and special honest-verifier zero-knowledge (SHVZK). Moreover, it has* $2\mu + 7$ *communication rounds and the communication costs are:*

- $\mathcal{P} \to \mathcal{V}$*:* 7 *elements of* $\mathbb{Z}_q$ *and* $2\mu + 1$ *elements of* $\mathbb{H}$*;*
- $\mathcal{V} \to \mathcal{P}$*:* $\mu + 3$ *elements of* $\mathbb{Z}_q$*.*

7.2.3 A Generalization to Bilinear Group Arithmetic Circuits

Every computable function with fixed input length can be expressed as an arithmetic circuit. Therefore interactive proofs for arithmetic circuit satisfiability are extremely powerful and widely deployed. In fact, they lead to an obvious, but indirect, approach for arbitrary relations:

1. Construct an arithmetic circuit capturing the relation;
2. Apply an efficient circuit ZK protocol to this arithmetic circuit.

However, for some relations, the associated arithmetic circuits can be large and complex, thereby losing the conceptual simplicity and possibly even the concrete efficiency over a more direct approach.

For instance, Lai et al. [LMR19] consider the *bilinear group arithmetic circuit model*. A bilinear group arithmetic circuit is defined over a bilinear group

$(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{H}, e)$, where $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{H}$ is a bilinear pairing between groups of prime order q. Its wires take values in either $\mathbb{Z}_q$, $\mathbb{G}_1$, $\mathbb{G}_2$ or $\mathbb{H}$, and gates are either group operations, $\mathbb{Z}_q$-scalar multiplications or bilinear pairings. Hence, bilinear group circuits are generalizations of arithmetic circuits. They directly capture relations encountered in, e.g., identity based encryption [SW05] and structure preserving signatures [AFG+16].

Every bilinear group arithmetic circuit can also be expressed as an arithmetic circuit. This requires every group element to be expressed as a vector of $\mathbb{Z}_q$-elements and every gate to be replaced by a $\mathbb{Z}_q$-circuit. For instance, for a highly optimized group of order $q \approx 2^{256}$, evaluating a single group exponentiation requires an arithmetic circuit with approximately 800 multiplication gates [HBH+20]. In the bilinear circuit model, exactly the same operation would only comprise a single gate. Hence, expressing a bilinear group arithmetic circuit as an arithmetic circuit can significantly increase its size. Therefore, avoiding this reduction might significantly reduce the communication costs.

Lai et al. [LMR19] generalize the Bulletproof framework for arithmetic circuits to handle bilinear group arithmetic circuits directly. Also compressed Σ-protocols admit a straightforward adaption for this more general model. To see this, we merely require two observations. First, the pairing-based commitment scheme of Section 5.3 allows a prover to commit to mixed vectors $\mathbf{x} \in \mathbb{Z}_q^{n_0} \times \mathbb{G}_1^{n_1} \times \mathbb{G}_2^{n_2} \times \mathbb{H}^{n_T}$. This commitment scheme is homomorphic and the size of a commitment is constant in $n_0 + n_1 + n_2$ and linear in n_T. Second, the gates in a bilinear group arithmetic circuit are either affine or bilinear. The affine gates are handled directly by our compressed Σ-protocols. Moreover, as before, the bilinear gates can be linearized via an arithmetic secret-sharing scheme. Altogether we obtain a compressed Σ-protocol for relations captured by bilinear arithmetic circuits. For more details we refer the reader to [ACR21].

7.3 Threshold Signature Scheme

In this section, as a second application of compressed Σ-protocols, we construct a transparent k-out-of-n threshold signature scheme (TSS) with threshold signatures that are $\mathcal{O}(\lambda \log n)$ bits, where λ is the security parameter. Recall that a TSS enables any set of at least k players, in a group of n, to issue a "threshold" signature on a message m, but no subset of less than k players is able to issue one. A TSS is called *transparent* if it does not require a trusted setup phase, i.e., all public parameters are random coins. Given recent advances in efficient circuit zero-knowledge, an obvious TSS construction defines a threshold signature as a proof of knowledge attesting the knowledge of k-out-of-n signatures. With the appropriate circuit zero-knowledge protocol this would immediately result in a transparent TSS with sublinear size threshold signatures. However, this approach requires an inefficient reduction from the corresponding threshold signature relation to a relation defined over an arithmetic circuit. More precisely, the arithmetic circuits capturing these relations are typically large.

For this reason, we follow a more *direct* approach avoiding this inefficient reduction. Namely, we append the BLS signature scheme [BLS01; BLS04] with a

k-aggregation algorithm. The BLS signature scheme is defined over a bilinear group. In particular, the BLS verification algorithm checks a linear constraint defined over a bilinear group. This naturally fits with the compressed Σ-protocols for opening homomorphisms. To derive the required threshold functionality, we use the proof of partial knowledge techniques from Section 4.3. The compressed Σ-protocols are interactive. To obtain a signature scheme, they can be made non-interactive by applying the Fiat-Shamir transformation [FS86].

The non-interactive proofs contain precisely the messages sent from the prover to the verifier in the interactive proof. Hence, the logarithmic TSS size is inherited from the logarithmic communication complexity of the compressed Σ-protocol.

The k-aggregation algorithm can be evaluated by any party with input at least k valid signatures from distinct signers. Besides the signatures, the k-aggregation algorithm only takes public input values. Moreover, the threshold k can be chosen at aggregation time independent of the set-up phase. By contrast, Shoup's construction [Sho00] requires a different trusted setup phase for every threshold k. Since the compressed Σ-protocol is special honest-verifier zero-knowledge, an additional property of our TSS is that a threshold signature hides the k-subset of signers S. Further, the TSS does not require a trusted setup and is therefore transparent. More precisely, the players can generate their own public-private key-pairs and the Σ-protocol only requires an unstructured public random string defined by the public parameters of the commitment scheme.

We deviate slightly from the standard TSS definitions. Therefore, in Section 7.3.1, we first formalize our security model before, in Section 7.3.2, we present our construction.

7.3.1 Definition and Security Model

We deviate from standard TSS definitions and aim for a strictly stronger functionality. In standard TSS definitions [Sho00; Bol03], a non-transparent mechanism (e.g., a trusted dealer or a multiparty computation protocol) generates a single public key and n private keys that are distributed amongst the n players. The private keys allow individual players to generate *partial* signatures on messages m. There is a public algorithm to aggregate k partial signatures into a threshold signature. The threshold signature can be verified with the public key generated by the trusted dealer.

By contrast, we define a TSS as an *extension* of a digital signature scheme. The fundamental strengthening of the definitions of [Sho00; Bol03] and related works, is that the public and private keys are generated by the players locally. Public keys are published on a *bulletin board* and thereby publicly tied to the player's identities. Since this setup does not require a trusted dealer (or another non-transparent mechanism for generating keys), it is said to be *transparent*. The players can individually sign messages by using their private keys. The aggregation algorithm now takes as input k signatures, instead of partial signatures, to generate a threshold signature. For simplicity we assume the threshold k to be fixed. We will explain later why our construction (trivially) satisfies some stronger properties.

Let us first give a definition for the basic building block of our TSS.

Definition 7.1 (Digital Signature)**.** A digital signature scheme consists of three

algorithms:

- KEYGEN is a randomized key generation algorithm that outputs a public-private key-pair $(\mathsf{pk}, \mathsf{sk})$;
- SIGN is a (possibly randomized) signing algorithm that, on input a message $m \in \{0,1\}^*$ and a secret key sk, outputs a signature $\sigma = \text{SIGN}(\mathsf{sk}, m)$;
- VERIFY is a deterministic verification algorithm that, on input a public key pk, a message m and a signature σ, outputs either accept or reject.

A signature scheme is *correct* if $\text{VERIFY}(\mathsf{pk}, m, \text{SIGN}(\mathsf{sk}, m)) = \mathsf{accept}$ with probability 1 for all key-pairs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \text{KEYGEN}$ and messages $m \in \{0,1\}^*$. If $\text{VERIFY}(\mathsf{pk}, m, \sigma) = \mathsf{accept}$, we say that σ is a *valid* signature on message m. Moreover, an adversary that does not know the secret key sk should not be able to forge a valid signature. This security property is formally captured in the widely accepted definition *Existential Unforgeability under Chosen-Message Attacks* (EUF-CMA) [Bol03]. We assume digital signature schemes to be correct and EUF-CMA by definition.

Definition 7.2 (Threshold Signature)**.** A k-out-of-n threshold signature scheme (TSS) is a digital signature scheme (KEYGEN, SIGN, VERIFY) appended with two algorithms:

- k-AGGREGATE is a (possibly randomized) aggregation algorithm that, on input n public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$, k signatures $(\sigma_i)_{i \in S}$ for a k-subset $S \subseteq \{1, \ldots, n\}$ and a message $m \in \{0,1\}^*$, outputs a threshold signature Σ;
- k-VERIFY is a deterministic verification algorithm that, on input n public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$, a message m and a threshold signature Σ, outputs either accept or reject;

Let $S \subseteq \{1, \ldots, n\}$ be some k-subset of indices and let $(\sigma)_{i \in S}$ be signatures, such that $\text{VERIFY}(\mathsf{pk}_i, m, \sigma_i) = \mathsf{accept}$, for all $i \in S$, and for some message $m \in \{0,1\}^*$. Then a TSS is *correct* if for all $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$, m, S and $(\sigma)_{i \in S}$,

$$k\text{-VERIFY}\big(\mathsf{pk}_1, \ldots, \mathsf{pk}_n, m, k\text{-AGGREGATE}(m, (\sigma_i)_{i \in S})\big) = \mathsf{accept}\,,$$

with probability 1. If $k\text{-VERIFY}(\mathsf{pk}_1, \ldots, \mathsf{pk}_n, m, \Sigma) = \mathsf{accept}$, we say that Σ is a valid threshold signature. Moreover, an adversary with at most $k-1$ valid signatures on a message m should not be able to construct a valid threshold signature. This *unforgeability* property can be formalized by the following security game. Consider an adversary that is allowed to choose a subset of $k-1$ indices $\mathcal{I} \subset \{1, \ldots, n\}$ and impose the values of the keys pk_i in this subset. Assume that all remaining keys pk_i were generated honestly from KEYGEN and therefore correspond to secret keys sk_i. The adversary is allowed to query polynomially many signatures $\sigma_i' = \text{SIGN}(sk_i, m')$ for arbitrary messages m'. The TSS is said to be *unforgeable* if the adversary is incapable of producing a valid k-out-of-n threshold signature on some message m that has not been queried.

7.3.2 The Threshold Signature Scheme

We follow a non-standard, but conceptually simple, approach for constructing a threshold signature scheme. The starting point of our TSS is a digital signature scheme (KEYGEN, SIGN, VERIFY) and the k-aggregation algorithm k-AGGREGATE simply produces a proof of knowledge of k valid signatures on a message m, i.e., a proof of knowledge for the following relation:

$$\begin{aligned}\mathfrak{R}_T = \big\{ \, &(\mathsf{pk}_1, \ldots, \mathsf{pk}_n, m; S, (\sigma_i)_{i \in S}) : \\ &|S| = k, \ \text{VERIFY}(\mathsf{pk}_i, m, \sigma_i) = \mathsf{accept} \ \forall i \in S \big\} \, .\end{aligned} \tag{7.1}$$

The obvious approach is to capture this relation by an arithmetic circuit, i.e., reduce it to a number of constraints defined over $\mathbb{Z}_q$, and apply a communication-efficient proof of knowledge for arithmetic circuit relations in a black-box manner. A significant drawback of this *indirect* approach is that it relies on an inefficient reduction to arithmetic circuit relations. For this reason, we follow a *direct* approach avoiding these inefficient reductions.

We instantiate our TSS with the BLS signature scheme [BLS01; BLS04] defined over a bilinear group $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{H}, e, G, H)$. Recall that we write the group operations in $\mathbb{G}_1$ and $\mathbb{G}_2$ additively and the group operations in $\mathbb{H}$ multiplicatively. Let us now briefly recall the BLS signature scheme, instantiated in our n-player setting. All players i, $1 \leq i \leq n$, generate their own private key $u_i \in \mathbb{Z}_q$, and publish the associated public key $P_i = u_i \cdot H \in \mathbb{G}_2$. To sign a message $m \in \{0,1\}^*$, player i computes signature $\sigma_i = u_i \cdot \mathcal{H}(m) \in \mathbb{G}_1$, where $\mathcal{H} \colon \{0,1\}^* \to \mathbb{G}_1$ is some (public) collision resistant hash function. The public verification algorithm accepts a signature σ_i if

$$e(\sigma_i, H) = e(\mathcal{H}(m), P_i) \, . \tag{7.2}$$

By the bilinearity of e, all honestly generated signatures are accepted. The unforgeability follows from the so called co-CDH* assumption [BLS04].

Remark 7.1. The BLS signature scheme was originally instantiated such that $\mathbb{G}_1 = \mathbb{G}_2$, i.e., both input coordinates of the pairing e are elements of the same group. However, the authors already showed that the scheme can be instantiated in a more general setting, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are possibly different. But still, their security proof, showing that unforgeability follows from the *Computational co-Diffie-Hellman* (co-CDH) assumption, requires the existence of an efficiently computable isomorphism $\psi \colon \mathbb{G}_2 \to \mathbb{G}_1$. As discussed in Section 2.6, the existence of such an isomorphism contradicts the SXDH assumption; more precisely, the DDH assumption in $\mathbb{G}_2$ cannot hold if there exists an efficiently computable isomorphism $\psi \colon \mathbb{G}_2 \to \mathbb{G}_1$. Now recall that the binding properties of the commitment schemes of Definition 5.2 and Definition 5.3 are derived from the DDH assumption in $\mathbb{G}_2$ and the SXDH assumption, respectively. Hence, at first glance BLS signatures and these pairing-based commitments appear incompatible, i.e., they seem to require different bilinear groups. Fortunately, Boneh, Lynn and Shacham already commented on the necessity of the isomorphism ψ in the journal version of their work [BLS04]. They mention that, by relying on a slightly different complexity assumption referred to as the co-CDH* assumption [SV07], the BLS signature

scheme can also be instantiated in bilinear groups $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H)$ without efficiently computable isomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$, i.e., bilinear groups of Type III [GPS08]. This shows that, under the co-CDH* assumption, we can safely instantiate the BLS signature scheme and the pairing-based commitment scheme in the same bilinear group. A more detailed analysis of certain pairing-based signature schemes, instantiated with Type III bilinear groups, is provided in [CHK+10]. In particular, they show that the co-DHP and co-DHP* assumptions are equivalent if the generators are suitably chosen and conclude that existing evidence suggests that Type III pairings offer at least as much security as Type II pairings when used to implement the BLS signature scheme.

In order to commit to mixed vectors with coefficients in both $\mathbb{Z}_q$ and $\mathbb{G}_1$, we will use the extended Pedersen vector commitment scheme of Definition 5.2:

$$\text{COM}\colon \mathbb{Z}_q^{n_0} \times \mathbb{G}_1^{n_1} \times \mathbb{Z}_q \to \mathbb{H}, \quad (\mathbf{x}, \mathbf{y}; \gamma) \mapsto \mathbf{h}^{\mathbf{x}} \cdot e(\mathbf{y}, \mathbf{g}) \cdot h^{\gamma}\,,$$

where $\mathbf{h}^{\mathbf{x}} := \prod_{i=1}^{n_0} h_i^{x_i}$ and $e(\mathbf{y}, \mathbf{g}) := \prod_{i=1}^{n_1} e(y_i, g_i)$. This commitment scheme is binding under the DDH assumption in $\mathbb{G}_1$. We do not need to be able to commit to $\mathbb{G}_2$- and $\mathbb{H}$-coefficients.

Instantiating relation $\mathfrak{R}_T$ with the BLS signature scheme therefore results in the following relation:

$$\mathfrak{R}_{TSS} = \{(P_1, \ldots, P_n, m; S, (\sigma_i)_{i\in S}) : |S| = k,\ e(\sigma_i, H) = e(\mathcal{H}(m), P_i)\ \forall i \in S\}\,.$$

The k-AGGREGATE algorithm simply computes a proof of knowledge for relation $\mathfrak{R}_{TSS}$. The main challenge is that the prover only knows k-out-of-n signatures. To handle this problem the k-out-of-n case is reduced to the n-out-of-n case by deploying the linear secret sharing based proofs of partial knowledge technique from Section 4.3. In fact, this technique allows us to reduce the nonlinear relation $\mathfrak{R}_{TSS}$ to a linear relation defined over the bilinear group $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{H}, e, G, H)$.

Let us recall the proof of partial knowledge technique in the context of the threshold signature relation $\mathfrak{R}_{TSS}$. First, the k-aggregator defines

$$p(X) = 1 + \sum_{j=1}^{n-k} a_j X^j \in \mathbb{Z}_q[X]$$

to be the unique polynomial of degree at most $n-k$ with $p(i) = 0$ for all $i \in \{1, \ldots, n\}\backslash S$. This polynomial defines an $(n-k+1)$-out-of-n secret sharing of 1, with shares $s_i = 0$ for all $i \notin S$. Then, the k-aggregator lets $\widetilde{\sigma}_i = p(i)\sigma_i$, where $\widetilde{\sigma}_i$ is understood to be equal to 0 for $i \notin S$, i.e., the secret sharing defined by $p(X)$ *eliminates* the signatures $(\sigma_i)_{i\notin S}$ that the k-aggregator does not know. Subsequently, the k-aggregator commits to the mixed vector

$$\mathbf{x} = (a_1, \ldots, a_{n-k}, \widetilde{\sigma}_1, \ldots, \widetilde{\sigma}_n) \in \mathbb{Z}_q^{n-k} \times \mathbb{G}_1^n\,.$$

Note that the committed vector $\mathbf{x}$ satisfies

$$f_i(\mathbf{x}) = f_i(a_1, \ldots, a_{n-k}, \widetilde{\sigma}_1, \ldots \widetilde{\sigma}_n) = e(\mathcal{H}(m), P_i)$$

for all $1 \leq i \leq n$, where

$$f_i \colon \mathbb{Z}_q^{n-k} \times \mathbb{G}_1^n \to \mathbb{H}, \quad \mathbf{x} \mapsto e(\widetilde{\sigma}_i, H) - \sum_{j=1}^{n-k} a_j i^j e(\mathcal{H}(m), P_i)\,. \tag{7.3}$$

Hence, by proving that the committed vector satisfies these relations, it follows that the k-aggregator knows a non-zero polynomial $p(X)$ of degree at most $n-k$ and group elements $\widetilde{\sigma}_1, \ldots \widetilde{\sigma}_n \in \mathbb{G}_1$ such that $e(\widetilde{\sigma}_i, H) = p(i)e(\mathcal{H}(m), P_i)$ for all $1 \leq i \leq n$. Therefore, the k-aggregator must know valid signatures for all indices i with $p(i) \neq 0$, and since $p(X)$ is non-zero and of degree at most $n-k$, at least k of its evaluations are non-zero. Because the mappings f_i are homomorphisms, the required proof of knowledge follows by applying the appropriate compressed Σ-protocol. As before, amortization can be applied to open all n homomorphisms $f_1, \ldots, f_n$ for essentially the price of one. Further, the protocol is made non-interactive by applying the Fiat-Shamir transformation. Altogether, the threshold signature contains a commitment $P \in \mathbb{H}$ to the mixed vector $\mathbf{x}$ together with a non-interactive proof of knowledge π of an opening of P that satisfies the aforementioned linear constraints. The k-AGGREGATE algorithm is summarized in Algorithm 17. The associated k-verification algorithm k-VERIFY simply runs the verifier of the compressed Σ-protocol. Correctness of the resulting threshold signature follows immediately from the completeness of the compressed Σ-protocol, and unforgeability follows from its (knowledge) soundness. The properties of the TSS are summarized in Theorem 7.5. Note that our TSS has some additional properties not required by the definition of Section 7.3.1. For instance, since the interactive proof of knowledge is special honest-verifier zero-knowledge, our threshold signatures hide the k-subset S of signers.

Theorem 7.5 (Threshold Signature Scheme)**.** *The k-out-of-n threshold signature scheme defined by the BLS signatures scheme [BLS01; BLS04], appended with the k-aggregation algorithm described in Algorithm 17, is correct and unforgeable. Moreover:*

- *a threshold signature contains exactly $4\lceil \log_2(n) \rceil + 3$ elements of $\mathbb{H}$, 1 element of $\mathbb{G}_1$ and 1 element of $\mathbb{Z}_q$;*
- *a threshold signature is zero-knowledge on the identities of the k-signers;*
- *the threshold k can be chosen at aggregation time;*
- *the threshold signature scheme resists against an adaptive adversary which, can replace the public keys of corrupted players.*

Proof. **Correctness** This immediately follows from the completeness of compressed Σ-protocol Σ_{comp}.

Unforgeability The proof is similar to the proof of Theorem 4.1, describing the properties of the proof of partial knowledge protocol. From special-soundness of the compressed Σ-protocol, it follows that there exists an efficient extractor $\mathcal{E}$ that outputs a vector $\mathbf{x}' = (\mathbf{a}', \tau_1, \ldots, \tau_n) \in \mathbb{Z}_q^{n-k} \times \mathbb{G}_1^n$ such that

Algorithm 17 Algorithm k-AGGREGATE.

PARAMETERS:	$k, n \in \mathbb{N}$, prime q, hash function $\mathcal{H}\colon \{0,1\}^* \to \mathbb{G}_1$ and bilinear group $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{H}, e, G, H)$
PUBLIC INPUT:	Public keys $P_1, \ldots, P_n \in \mathbb{G}_2$ and message $m \in \{0,1\}^*$
PRIVATE INPUT :	Subset $S \subseteq \{1, \ldots, n\}$ and signatures $\sigma_i \in \mathbb{G}_1$ $\forall i \in S$
OUTPUT :	TSS $\Sigma = (\pi, P) \in \mathbb{Z}_q \times \mathbb{G}_1 \times \mathbb{H}^{4\lceil \log_2(n) \rceil + 3} \cup \{\perp\}$

1. If $\exists i \in S$ such that $e(\sigma_i, H) \neq e(\mathcal{H}(m), P_i)$, output $\perp$ and abort.
2. Compute the unique polynomial $p(X) = 1 + \sum_{i=1}^{n-k} a_j X^j \in \mathbb{Z}_q[X]$ of degree at most $n-k$ such that $p(i) = 0$ for all $i \in \{1, \ldots, n\} \backslash S$.
3. Compute $\widetilde{\sigma}_i := p(i)\sigma_i$ for all $i \in S$ and set $\widetilde{\sigma}_i = 0$ for all $i \notin S$.
4. Let $\mathbf{x} = (a_1, \ldots, a_{n-k}, \widetilde{\sigma}_1, \ldots, \widetilde{\sigma}_n) \in \mathbb{Z}_q^{n-k} \times \mathbb{G}_1^n$ and compute commitment $P = \text{COM}(\mathbf{x}; \gamma) \in \mathbb{H}$ for $\gamma \in \mathbb{Z}_q$ sampled uniformly at random.
5. Run the non-interactive variant of compressed Σ-protocol Σ_{comp} to produce a proof π attesting that the committed vector $\mathbf{x}$ satisfies

$$f_i(\mathbf{x}) = f_i(a_1, \ldots, a_{n-k}, \widetilde{\sigma}_1, \ldots \widetilde{\sigma}_n) = e(\mathcal{H}(m), P_i)$$

   for all $1 \leq i \leq n$, where f_i are the homomorphisms defined in Equation (7.3).
6. Output commitment P and the non-interactive proof π.

$f_i(\mathbf{x}) = e(\mathcal{H}(m), P_i)$ for all $1 \leq i \leq n$, where f_i is as in Equation (7.3). Let us denote $p'(X) = 1 + \sum_{i=1}^{n-k} a'_j X^k \in \mathbb{Z}_q[X]$, then $S' = \{i : p'(i) \neq 0\}$ has cardinality at least k. Moreover, it is easily seen that $p'(i)^{-1} S_i$ is a valid BLS signature on message m associated to public key P_i. Hence, an adversary capable of forging a threshold signature is also capable of computing k distinct valid signatures on m. Since the adversary is capable of corrupting at most $k-1$ players, this contradicts the unforgeability of the BLS signature scheme.

The remaining properties are trivially verified. □