On the Portability of Generalized Schnorr Proofs

Jan Camenisch[1⋆], Aggelos Kiayias[2⋆⋆], and Moti Yung[3]

[1] IBM Research, Zurich, Switzerland, jca@zurich.ibm.com
[2] Computer Science and Engineering, University of Connecticut
Storrs, CT, USA. aggelos@cse.uconn.edu
[3] Google Inc. and Computer Science, Columbia University
New York, NY, USA. moti@cs.columbia.edu

Abstract. The notion of Zero Knowledge Proofs (of knowledge) [ZKP] is central to cryptography; it provides a set of security properties that proved indispensable in concrete protocol design. These properties are defined for any given input and also for any auxiliary verifier private state, as they are aimed at any use of the protocol as a subroutine in a bigger application. Many times, however, moving the theoretical notion to practical designs has been quite problematic. This is due to the fact that the most efficient protocols fail to provide the above ZKP properties *for all* possible inputs and verifier states. This situation has created various problems to protocol designers who have often either introduced imperfect protocols with mistakes or with lack of security arguments, or they have been forced to use much less efficient protocols in order to achieve the required properties. In this work we address this issue by introducing the notion of "protocol portability," a property that identifies input and verifier state distributions under which a protocol becomes a ZKP when called as a subroutine in a sequential execution of a larger application. We then concentrate on the very efficient and heavily employed "Generalized Schnorr Proofs" (GSP) and identify the portability of such protocols. We also point to previous protocol weaknesses and errors that have been made in numerous applications throughout the years, due to employment of GSP instances while lacking the notion of portability (primarily in the case of unknown order groups). This demonstrates that cryptographic application designers who care about efficiency need to consider our notion carefully. We provide a compact specification language for GSP protocols that protocol designers can employ. Our specification language is consistent with the ad-hoc notation that is currently widely used and it offers automatic derivation of the proof protocol while dictating its portability (i.e., the proper initial state and inputs) and its security guarantees. Finally, as a second alternative to designers wishing to use GSPs, we present a modification of GSP protocols that is unconditionally portable (i.e., ZKP) and is still quite efficient. Our constructions are the first such protocols proven secure in the standard model (as opposed to the random oracle model).

1 Introduction

Motivation. Zero knowledge proofs [28] [ZKP], and zero knowledge proofs and arguments of knowledge in particular, are a central tool in cryptosystem and protocol design.

⋆ This research has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no 216483.
⋆⋆ This research was partly supported by NSF CAREER 0447808, and NSF CNS 0831306.

These tools allow a designer to enforce parties to assure others that they take specified actions consistent with their internal knowledge state [26]. Properties of ZKP are defined over *all inputs* i.e., they provide security and correctness properties independently of input distribution. A shortcoming of ZKP's is that depending on the underlying language it can be hard to come up with efficient protocols. This has lead to the design of specialized protocols for specific language classes that occur often in applications. A celebrated example that has proven to be very useful in the design of efficient cryptographic schemes is known as Generalized Schnorr Proofs (extending the original seminal proof [37] to various algebraic settings like unknown order modular groups that arise in the context of the RSA cryptosystem). These protocols are at the heart of many efficient cryptographic systems and have been employed in a great number of schemes including: anonymous e-cash, anonymous voting, group signatures, distributed signing, distributed decryption, verifiable encryption, fair exchange, ring signatures, and credential systems. These schemes capitalized on the high efficiency of Schnorr's method and constitute, perhaps, the most extensive application of zero knowledge theory to practice so far. Further, a shorthand notation introduced in [14, 15] for GSP has been extensively employed in the past and contributed to the wide employment of these protocols in cryptographic design. This notation suggested using e.g., $\mathsf{PK}(\alpha : y = g^{\alpha})$ to denote a proof of the discrete logarithm $\log_g y$ and it appeared in many works to describe quite complex discrete logarithm based relations, e.g., [3, 7–11, 13, 24, 25, 30–34, 38–43]. What has been often overlooked though is the fact that Generalized Schnorr Proofs are *not* zero-knowledge proofs of knowledge! This is a consequence of the fact that the security properties of such protocols are affected by the input distribution of the involved parties. Interestingly, despite the long line of works in the proper formalization of zero-knowledge proofs, this aspect has been largely overlooked, mainly due to the fact that it is only critical from an application-oriented *efficiency* point of view rather than a theoretical *feasibility* point of view. Let us illustrate the phenomenon with two examples:

Example 1. Consider the language $\mathcal{L} = \{\langle n, g, h, y\rangle \mid \exists s, t : y = g^s h^t \bmod n\} \subseteq \mathcal{L}_{\mathsf{in}} = \mathbb{N}_k^4$ where $\mathbb{N}_k$ is all k-bit numbers and the following variation of the standard Schnorr proof: the prover sends the value $u = g^{s_0} h^{t_0}$ for some random integers s_0, t_0; upon receiving u the verifier responds with some integer c and finally the prover responds with $s_1 = s_0 - c \cdot s$ and $t_1 = t_0 - c \cdot t$ (calculated over the integers). The verifier returns 1 if and only if $u = y^c g^{s_1} h^{t_1} \bmod n$. This protocol has been used numerous times (see e.g., [23, 15, 1]). However the protocol *is not a proof of knowledge*: on the one hand, in the case that the factorization of n is easy, it is feasible to design a knowledge extractor that in expected polynomial time can recover the witness to the statement when interacting with any convincing prover. Nevertheless such extractor can only succeed for certain choices of y as the above protocol can make the verifier accept with high probability even for "malformed" y's that satisfy $y = \zeta g^s h^t$ where ζ is a small order element of $\mathbb{Z}_n^*$. Furthermore, when the factorization of n is difficult, the knowledge extractor cannot even take advantage of Chinese remaindering to process the values submitted by the prover; in such case ensuring the verifier that a convincing prover is indeed in possession of a witness becomes even more elusive. In addition, observe that the zero-knowledge property is affected by the way the protocol is executed,

and in particular the statistical zero-knowledge aspect of the above protocol depends on the relative sizes of s_0, s and t_0, t.

Example 2. Consider the language $\mathcal{L} = \{\langle n, g, y\rangle \mid \exists s, r : y = g^{s^2} h^r\}$. A way for designing an efficient protocol for this language is to have the prover provide a commitment $C = g^s h^{r'}$ and then prove simultaneously the knowledge of the commitment C as well as the commitment C^s using two instances of the protocol in example 1. Clearly, in this case we will have to deal with similar issues as in example 1, but furthermore we will have an additional difficulty to simulate the value C as part of the zero-knowledge simulator. For choices of the values of g, h, n where $\langle h\rangle$ happens to be a subgroup of $\mathbb{Z}_n^*$ different than $\langle g\rangle$ it can be the case that C is not sufficiently hiding its g^s component. For example $\langle h\rangle$ can be the subgroup of quadratic residues in $\mathbb{Z}_n^*$ and g a quadratic non-residue; this choice would be leaking one bit about the committed value s.

The above two cases exemplify the fact that there are many efficient protocols that are not zero-knowledge proofs but they may potentially be used as such as long as they are employed over a suitable input generation. It follows that given the state of the art what is badly missing is a *methodological,* i.e, a formal way to guide cryptographic protocol designers under what conditions (on input and verifier's state) it is safe to deploy these efficient protocols as subroutines in a larger application context. Identifying such safety conditions and attaching them to a protocol is what we call "identifying the protocol's *portability*."

We say that a protocol is *portable* with safety conditions defined by a class of input generators, for the class over which it retains the properties of zero-knowledge proof of knowledge. The lack of properly identifying this notion has created a number of crucial protocol problems on previously published works. For example, the work of [23] has been cited extensively and its results were used directly to justify the proof of knowledge properties of various proposed schemes. This was done without realizing that some of the security arguments in [23] are incorrect, which was finally noticed (and corrected but without providing a formal protocol framework) by Damgård and Fujisaki [21] five years after the publication of the original paper. Further, in various cases the possibility of a biased input generation and reference string contribution by one of the parties was not considered (either in the model or as an omission or as an oversight) and this led to other works pointing out actual problems in these cases. For example, see the attack of [16] on [1] that illustrates how a malicious key generation leads to a soundness attack in the underlying signing protocol that, in turn, enables a framing attack in the group signature scheme. Another example is the attack of [29] on [5] that takes advantage of a malicious parameter generation to break the zero-knowledge property of the protocol construction. In both cases the required properties can be preserved by ensuring proper parameter generation (as it was argued in [2] and [5] respectively). These previous problem instances highlight the need of having a proper formalism that identifies conditions for porting efficient protocols as zero-knowledge proofs.

Our Contributions.

1. We introduce the notion of *portability* for proofs of knowledge protocols which identifies input and initial constraints under which a protocol can be employed and

have the zero-knowledge proof properties. First, we define the notion of an *input-generator* for a proof protocol and we formalize the properties of soundness and zero-knowledge conditional on a given input generator. The portability of the protocol is defined, in turn, by identifying classes of input generators for which the protocol is sound and zero-knowledge (thus, can be deployed safely). Note that *unconditional portability* characterizes protocols that retain their properties for any input distribution (i.e., this notion coincides with regular zero-knowledge proofs of knowledge).

2. We then identify a large class of input generation and soundness parameters over which Generalized Schnorr Proofs (GSP) are portable. This clarifies the correct way to employ the highly popular protocol description notation introduced in [14, 15] for GSP mentioned above. Based on our results the (frequently lacking and often erroneous) security analysis of all these previous works is streamlined and presented in a unified way. Indeed, the notation $\mathsf{PK}(\alpha, \ldots : y = g^{\alpha}, \ldots)$ was originally suggested for a few specific protocols without clear semantics and syntax for the notation nor with a way to derive a concrete protocol for the notation. Subsequently, the notation was extended by many authors and was also used in different (algebraic) settings thereby opening gaps between statement made in the notation and the security properties offered by the protocol that the authors seemingly had in mind. Sometimes, the notation has also been used with no particular protocol in mind but just to describe any protocol (e.g., a generic zero-knowledge proof protocol) that proves knowledge of a witness to the statement. This leads to our next contribution.
3. We introduce a new notation PKspec for specifying GSP proofs that puts forth the soundness guarantees provided by the protocol specified by it. Our notation can be used as a black-box in protocol design and the respective security proofs. To illustrate our notation, as an example, consider two parties that jointly compute the values U, V, n such that $U, V \in \mathbb{Z}_n^*$ and one of them wishes to demonstrate a certain structural relationship between them. This goal will be specified syntactically in the following way (for example):

$$\begin{aligned}&\mathsf{PKspec}(\alpha_1, \alpha_2 : (V = h^{\alpha_1} g^{\alpha_2} \text{ in } \mathbb{Z}_n^*) \wedge \alpha_1 \in [-\infty \ldots + \infty] \wedge \alpha_2 \in [L \ldots R]) \\ &\rightarrow (\alpha_1, \alpha_2 : (V = \zeta \cdot h^{\alpha_1} g^{\alpha_2} \text{ in } \mathbb{Z}_n^*) \wedge \alpha_1 \in [-\infty \ldots + \infty] \wedge \alpha_2 \in [L' \ldots R'])\end{aligned}$$

   Note that the specification is divided into two parts, the one appearing in the first line is what the protocol designer (ideally) wishes to ensure and the second is what will actually be ensured by the Schnorr protocol (in particular, the values ζ_1, ζ_2 will be selected from some small subgroup and the range $[L', R']$ may be extended compared to $[L, R]$). Based on our work, a protocol designer may write a GSP specification as above and then rely on our analysis for the proof of a security and soundness (which assures portability of the GSP protocol to his/ her specific context).
4. To complete the tool kit for protocol designers, we introduce an efficient extension of GSP protocols that is unconditionally portable. This construction is proven correct and secure in the standard model, whereas the only previously known efficient

protocols — known as the class of Σ^+ protocols [5] — were shown secure in the random oracle idealization.

5. The identification of portability for Generalized Schnorr Proofs facilitates the correct and secure design of efficient protocols. To illustrate the power of our framework in this context we consider two well-known cryptographic constructions from different subareas. We show how the employment of our GSP framework clarifies their design and the assumptions they depend on, and assures their security while coping with previously presented attacks. We first consider the original scalable group signature scheme by Ateniese et al. [1] mentioned earlier. Recently, [16] presented an attack (which is actually based on considering the extended setting of dishonest group manager at the system's setup phase, something not originally anticipated; see [2] for a discussion). Employing the GSP framework, in turn, allows us to clarify the settings where the protocol of [1] is secure and highlights the exact requirements on the joint input to the proof of knowledge. As a side benefit our framework also shows how the scheme can be made more efficient. Next, we consider the efficient divisible e-cash scheme of Chan et al. [17]; the security of this scheme was never analyzed properly (and originally the scheme as published had problems). Employing our GSP framework here, we reveal the exact cryptographic assumptions required for the modified scheme to be secure (something that even the corrected version [18] has been lacking).

   Due to lack of space the above contributions are included in the full version of the paper available in [12].

How to use the results of this paper in cryptographic protocol design. Here we comment briefly on the way our results can be used in cryptographic design. Suppose that in a certain cryptographic system a party is required to execute a proof that involves a series of discrete-log relations expressed in the widely used ad-hoc PK notation. Using Theorem 1 the designer can obtain the corresponding PKspec expression and, by the same theorem also automatically get the GSP protocol implementing the proof. Then the designer examines the input generation that preceeds the protocol which is defined by the system execution until the moment the GSP protocol should be invoked; if the conditions of Theorem 1 are satisfied then the soundness and the zero-knowledge property are implied immediately. If on the other hand, the conditions of Theorem 1 are not met, then the designer may use the unconditionally portable transformations of GSP protocols presented in section 6. For two concrete examples the reader can refer to the full version of the paper [12].

2 Preliminaries

Notations. A function $f : \mathbb{N} \rightarrow \mathbb{R}$ is called negligible if for all $c \in \mathbb{R}$ there exists $\nu_0 \in \mathbb{N}$ so that for all $\nu \geq \nu_0$ it holds that $f(\nu) < \nu^{-c}$. When a random variable x is distributed according to the probability distribution X with support S we will write $\mathbf{Prob}_{x \leftarrow X}[x = s]$ for the probability that x takes the value $s \in S$. Let x, y be two random variables with the same support $S(\nu)$ distributed according to the probability

distributions $X(\nu), Y(\nu)$ where $\nu \in \mathbb{N}$. We say that x, y are statistically indistinguishable if the function $f(\nu) := \frac{1}{2}\sum_{s \in S(\nu)} |\mathbf{Prob}_{x \leftarrow X(\nu)}[x = s] - \mathbf{Prob}_{y \leftarrow Y(\nu)}[y = s]|$ is a negligible function. If $m \in \mathbb{N}$ we will use the notation $[m]$ to denote the set $\{0, \ldots, m-1\}$. In general we will denote by $\mathcal{L}$ some language typically over alphabet $\{0,1\}$ unless otherwise specified. If $\mathcal{L}$ is an NP language, $R_{\mathcal{L}}$ will be the corresponding polynomial-time relation, i.e., $\mathcal{L} = \{\phi \mid \exists w : (\phi, w) \in R_{\mathcal{L}}\}$.

Interactive Protocols. Let $\Pi = (P, V)$ be a protocol where P, V are probabilistic interactive Turing machines (ITM). The *view* of P in Π is a random variable that contains all messages exchanged with V as well as the contents of all tapes of P. Two protocols $\Pi_1 = (P_1, V_1), \Pi_2 = (P_2, V_2)$ can be concatenated if we execute first (P_1, V_1) and then write the private outputs of P_1, V_1 to the input tapes of P_2, V_2 respectively and start the execution of (P_2, V_2). We allow parties to output a special symbol $\perp$ to signify that they "reject" a certain interaction. In the context of sequentially composed protocols, producing a $\perp$ symbol at some intermediate stage would signify that a party refuses to continue with the execution (and the final output of the party becomes $\perp$ which may interpreted as reject in the context of zero-knowledge proofs). For a given protocol $\Pi = (P, V)$ we will say that the two ITM's V, V' are indistinguishable provided that in the context of the Π interaction it is impossible for any adversarial P to distinguish whether it is communicating with V or V' (the notion is defined similarly for the case of the ITM's P, P').

3 Portability of Zero-Knowledge Proofs

A zero-knowledge proof protocol $\Sigma = (P, V)$ for a language $\mathcal{L}$ enables P to demonstrate to V that a joint input t belongs to an NP language $\mathcal{L}$ provided that the prover possesses a witness w such that $(t, w) \in \mathcal{R}_{\mathcal{L}}$. Soundness and zero-knowledge of such protocols should hold for any input distribution. Here we consider the (non-limiting) case that the prover and the verifier collaboratively construct the input t to the proof protocol by engaging in a protocol Π (dubbed the "input-generator"); at this preamble stage we denote the two parties by $P_{\mathsf{in}}, V_{\mathsf{in}}$ to highlight their relation with the actual prover and verifier. The output of this preamble stage will be the input to the actual prover and verifier.

Definition 1. *Let $\mathcal{L}_{\mathsf{in}} \in \mathsf{BPP}, \mathcal{L} \in \mathsf{NP}$ with $\mathcal{L} \subseteq \mathcal{L}_{\mathsf{in}}$. Consider Π, a two-party protocol $\Pi = \langle P_{\mathsf{in}}, V_{\mathsf{in}} \rangle$ where each party may reject returning $\perp$ while if P_{in} terminates successfully it returns a pair $\langle t, w_P \rangle$ and similarly V_{in} returns $\langle t', w_V \rangle$ where $t, t' \in \mathcal{L}_{\mathsf{in}}$. The protocol Π is called an* input generator *for $\mathcal{L}$, if for all executions that neither party returns $\perp$ it holds that $(t, w_P) \in R_{\mathcal{L}}$ and $t = t'$.*

Next we define statistical zero-knowledge proofs of knowledge over input generators. The definition follows the standard ZK notion with the only difference being that the input instead of being totally adversarial (i.e., universally quantified) is produced by an input generator protocol Π. The parties *are allowed* to be adversarial during this input generation stage. In particular for soundness we allow the prover to bias the input generation and in formalizing soundness the knowledge extractor will be interacting

with the malicious prover in both stages (with rewinding power only during the second stage, i.e., the proof system). Regarding zero-knowledge we condition on all input generation executions that the honest prover agrees to execute the proof system and we require the existence of a simulator that can simulate the view of any malicious verifier. Note further that to support design flexibility we will allow the prover to show that the input belongs to a possibly extended language $\mathcal{L}_{\mathsf{ext}}$.

Definition 2. *The two party protocol $\Sigma = \langle P, V \rangle$ is a zero-knowledge proof of knowledge over the input generator $\Pi = \langle P_{\mathsf{in}}, V_{\mathsf{in}} \rangle$ for $\mathcal{L}$ with knowledge error parameters $(\mathcal{L}_{\mathsf{ext}}, \kappa)$ and zero-knowledge distance ϵ if these properties are satisfied:*
(1) Completeness: it holds that both P_{in} and V_{in} terminate successfully with overwhelming probability and subsequently V accepts the interaction with the prover P with overwhelming probability.
(2) Soundness: For any pair of (P^*_{in}, P^*) we denote by $\pi_{P^*_{\mathsf{in}}, P^*}$ the probability that P^* convinces V on inputs generated by P^*_{in} and V_{in} (where $\pi_{P^*_{\mathsf{in}}, P^*}$ is taken over the entire probability space of $(P^*_{\mathsf{in}}, V_{\mathsf{in}}), (P^*, V)$). We say that Σ is sound over Π, if there is some K_{in}, such that: (i) K_{in} and V_{in} are indistinguishable as ITM's, (ii) for any P^* there is some K for which it holds that for any P^*_{in}: K on input the view of K_{in} and the output of P^*_{in}, it returns w' such that $(\mathsf{t}, w') \in R_{\mathcal{L}^{\mathsf{ext}}}$ where t is the statement that is determined in the input generation stage between P^*_{in} and K_{in} with probability of success at least $\mathsf{c} \cdot \pi_{P^*_{\mathsf{in}}, P^*}$ where $\mathsf{c} \in \mathbb{R}$ while running in time polynomial in $(\pi_{P^*_{\mathsf{in}}, P^*} - \kappa)^{-1}$.
(3) Zero-knowledge: Σ is statistical ZK over Π, if there exists an S_{in}, such that (i) S_{in} and P_{in} are indistinguishable as ITMs, (ii) for any V^*, there is a simulator S, such that for any V^*_{in}: the random variable that equals the view of V^* when interacting with P on input generated by $P_{\mathsf{in}}, V^*_{\mathsf{in}}$ is distinguishable with distance at most ϵ from the random variable that equals the output of S given as input the view of S_{in} and the output of V^*_{in}.

We next introduce the notion of portability of a protocol:

Definition 3. *The two party protocol $\Sigma = \langle P, V \rangle$ is said to be* portable *over the class of input generators $\mathcal{W}$ if for all $\Pi \in \mathcal{W}$ it holds that Σ a zero-knowledge proof of knowledge over Π. If $\mathcal{W}$ contains all possible protocols then the protocol Σ is said to be* unconditionally portable.

Ensuring portability from semi-honest behavior. Suppose that a given protocol happens to be a zero-knowledge proof of knowledge for some input-generator Π as long as the prover and the verifier are semi-honest at the input generation stage. In such an occasion one can generically compile a protocol Σ^* from Π and Σ so that Σ^* becomes a zero-knowledge proof of knowledge over Π using the transformation from semi-honest to malicious behavior put forth in [26] (see also [27], section 7.4). Note that while this is feasible, it is not particularly efficient given that it requires expensive steps such as coin-flipping combined with generic zero-knowledge proofs to ensure that no party is deviating from the input distribution (recall that much of cryptographic protocol design is motivated by avoiding generic inefficient tools). Our results will demonstrate that such generic techniques can be substituted by much more efficient ones for the particular class of protocols we consider (i.e., generalized Schnorr proofs).

Comparison to common-reference-string/bare model ZK. Zero-knowledge proofs are sometimes modeled in the common-reference string model, cf. [20] (or the common random string model, [36]); in this setting there is an explicit separation between the input of parties and the reference string that is assumed to be honestly generated and provided to the parties. A common-reference-string ZK protocol is supposed to satisfy the security properties conditional on the distribution of the reference string that no party can bias. By comparison, in our setting there is no unbiased reference string that is independent of the proof's statement that can be used to assist in the proof of soundness or zero-knowledge. While here we deal mainly with the bare model, it is worth noting that even the availability of a common reference string does not eliminate the issues of context dependent contributed inputs.

Relaxed Knowledge Extraction. In our formulation, the knowledge extractor only ensures that the prover possesses knowledge of a witness showing that t belongs to an extended language $\mathcal{L}_{\mathsf{ext}}$. If $\mathcal{L} = \mathcal{L}_{\mathsf{ext}}$ the soundness definition will ensure that the interactive input belongs to $\mathcal{L}$ (as in the standard definition of ZK), however we will also consider slightly different languages $\mathcal{L}_{\mathsf{ext}}$. The reason for this relaxation is that by extending the language one may obtain more efficient protocols which is our primary concern. Naturally this will allow the prover to convince the verifier to accept despite the fact that the interactive input may be in the "gray area" $\mathcal{L}_{\mathsf{ext}} - \mathcal{L}$. Note that in principle we will always be able to modify the interactive input proof of knowledge so that $\mathcal{L} = \mathcal{L}_{\mathsf{ext}}$ (if one does not mind the additional computation overhead that will be incurred).

Sigma Protocols. Our applications will focus on protocols $\langle P, V \rangle$ that are called Σ-protocols, i.e., a three-move protocol in which the prover goes first, the verifier responds with a random challenge from $\{0,1\}^k$, the prover responds, and finally the verifier either accepts or rejects based on the prover's response. All conversations in a Σ-protocol are of the form $\langle com, c, res \rangle$ (commitment, challenge, response). These protocols typically consider the setting where the verifier is restricted to be "honest" during the interactive proof $\langle P, V \rangle$ when proving the zero-knowledge property. While we will follow this, however, we will still allow the verifier to be totally adversarial in the input building stage. This is justified as the honest verifier setting can be transformed using numerous techniques to the fully adversarial verifier setting (e.g. see [35, 20]) and these techniques readily apply to our setting.

Variations of the definition. In our definition we focused on knowledge extraction following the definition of [6] (note that in our protocols the knowledge error will be $\kappa = 2^{-k}$ where k is a parameter). Moreover we formulated zero-knowledge in the statistical sense. It is easy to reformulate the definition by strengthening zero-knowledge (e.g., perfect zk) or relaxing it (e.g., computational zk). Moreover, soundness can be relaxed to require only language membership from the prover (instead of knowledge extraction), or defined with a specialized knowledge extractor that extracts two accepting conversations with the same first move and then reconstructs the witness. Further, in applications the protocols can be made non-interactive employing the Fiat-Shamir heuristics [22] and then use the forking Lemma [35] for extraction in the random oracle model. These alternative definitions are well understood in the context of building efficient zero-knowledge proofs and can be ported into our setting.

On the input generation stage. In an actual system, the input generator protocol $\langle P_{\mathsf{in}}, V_{\mathsf{in}} \rangle$ may abstract many parties and involve interactions between many participants. From a ZK security point of view, P_{in} will comprise the "prover side" (i.e., the side that is interested in preserving zero-knowledge) and V_{in} will comprise the "verifier side" (i.e., the side of the system that is interested in in preserving soundness). In a multi-party system, we will be interested in primarily two input generators: in the first one, P_{in} will include only the prover and (if it exists) any party the prover trusts while V_{in} will include all other participants. In the second one, V_{in} will include the verifier and (if it exists) any party the verifier trusts, while P_{in} will include all other participants. If a protocol is portable over both of these input generators then it can be safely deployed in the given system.

A central tool in our design is the notion of safeguard groups that we introduce next.

4 Safeguard Groups

A safeguard group is specified by a sampler algorithm S_{sg} that on input 1^ν returns a tuple $\langle \mathbb{G}, g, M, k, \zeta \rangle$; where $\mathbb{G}$ is a description of an Abelian group that contains an implementation of $\mathbb{G}$'s binary operator, inverse computation, the encoding of 1 as well as the description of a polynomial-time group membership test that, given any string, it decides whether it is a proper encoding of a group element; g is a generator of $\mathbb{G}$; M is an approximation of the order of g in $\mathbb{G}$; and k is a security parameter that is related to the length of the order of small-order group elements. Note that we will use the same notation for the description of a group $\mathbb{G}$ and the group itself. Regarding the remaining elements of the tuple we have that $g \in \mathbb{G}, \zeta \subseteq \mathbb{G}, M \in \mathbb{N}$ with further properties to be specified below.

Definition 4. *A safeguard group sampler S_{sg} satisfies the following (where $\langle \mathbb{G}, g, M, k, \zeta \rangle \leftarrow S_{\mathsf{sg}}(1^\nu)$):*

- C1. *The exponent of $\mathbb{G}$ is not divisible by the square of any k-bit integer.*
- C2. *The order m of g in $\mathbb{G}$ has no k-bit integer divisor, and M satisfies that $(M - m)/M = \mathsf{negl}(\nu)$.*
- C3. *ζ contains only a polynomial (in ν) number of elements; they all have a known (say part of the subgroup description) k-bit integer order.*
- C4. **Small-Order Property.** *It is hard to find k-bit order elements of $\mathbb{G}$ outside ζ. Formally, it holds that for all PPT $\mathcal{A}$, $\mathbf{Prob}[(v \notin \zeta) \wedge (v \text{ has } k \text{ bit order}); v \leftarrow \mathcal{A}(1^\nu, \tau); \tau = (\mathbb{G}, g, M, k, \zeta) \leftarrow S_{\mathsf{sg}}(1^\nu)] = \mathsf{negl}(\nu)$.*
- C5. **Strong-Root Property.** *Given $z \in \langle g \rangle$ it is hard to find $e > 1$ and $u \in \mathbb{G}$ such that $u^e = z$. Formally, it holds that for all PPT $\mathcal{A}$, $\mathbf{Prob}[(u^e = z) \wedge (e > 1); \langle u, e \rangle \leftarrow \mathcal{A}(1^\nu, \tau, z); z \leftarrow_R \langle g \rangle; \tau = (\mathbb{G}, g, M, k, \zeta) \leftarrow S_{\mathsf{sg}}(1^\nu)] = \mathsf{negl}(\nu)$.*

We remark that properties C3-C4 are not really essential and can be dropped at the expense of loosing tightness in some of our proof reductions and notational presentation; we opt to enforce them as they make the presentation of the results more succinct and are easily satisfied for the known examples of safeguard groups.

Example 1. A safeguard group distribution can be built as follows: sample n as a safe composite so that $n = pq$, $p = 2p' + 1, q = 2q' + 1$, where p', q' are prime numbers larger than 2^k, set $\mathbb{G} = \mathbb{Z}_n^*$ and let g be a generator of quadratic residues modulo n. Finally set $\zeta = \{1, -1\}$ and $M = \lfloor \frac{n}{4} \rfloor$. Property C1 is immediate as the exponent of $\mathbb{Z}_n^*$ is $2p'q'$. Observe also that the properties C2 and C3 are easily satisfied. Indeed, it is easy to see that M is sufficiently close to $p'q'$. Next observe that a violation of property C4 would mean the recovery of any other element that has a k-bit order outside $\{1, -1\}$; this would violate the factoring assumption (only the four square roots of 1 are k-bit order elements in $\mathbb{Z}_n^*$ based on our selection of n). Property C5 amounts to the Strong-RSA assumption with the target challenge being an arbitrary element of the quadratic residues; this is a variant of the strong RSA problem that has been utilized extensively in previous works (e.g., [19]).

Example 2. A second safeguard group is over the group $\mathbb{G} = \mathbb{Z}_{n^2}^*$ where n is sampled as before, i.e., $n = pq$, $p = 2p' + 1, q = 2q' + 1$, so that g is a generator of the subgroup of square n-th residues; as before we select p', q' larger than 2^k and $\zeta = \{1, -1\}$.

We remark that in both the above examples it is not necessary to select n as a safe composite, i.e., we may allow p' and q' to be composite numbers themselves as long as they have no small divisors (of k-bits). In practical settings where we will employ safeguard groups, the parameter k may be required to be in the range from 80 to 256 bits.

Properties of Safeguard Groups. In the first lemma below regarding safeguard groups we show that based on the properties of the safeguard group it is hard for an adversary to produce arbitrary powers of a chosen power of a group element. This lemma is an important building block of our general proof protocol. We remark that various restricted special case incarnations of this lemma have appeared in the literature (the most basic of which is referred to as Shamir's trick and corresponds to case (i) in the proof of lemma). These special incarnations are too restricted to be useful in our setting and thus there is need for putting forth the lemma that is formulated as follows:

Lemma 1. *Let $\tau = \langle \mathbb{G}, g, M, k, \zeta \rangle \leftarrow S_{\mathsf{sg}}(1^\nu)$ be a safeguard group distribution. Suppose that $\mathcal{A}$ is a PPT that given τ and a random $z \in \langle g \rangle$ returns $y \in \mathbb{G}$ and $t, m \in \mathbb{Z}$ such that $y^t = z^m$ with $1 \leq \gcd(t, m) < |t|$ and t is a k-bit integer. It holds that the success probability of $\mathcal{A}$ is negligible in ν.*

Our main result regarding safeguard groups is Lemma 3. We show that any adversary that is given any number of bases from the $\langle g \rangle$ subgroup of the safeguard group is incapable of producing an entirely arbitrary discrete-log representation of a power of his choosing within $\mathbb{G}$. Before stating the main lemma, we show an auxiliary lemma.

Lemma 2. *Let A, B be two integers with $A > B$ and $A = \pi B + v$ with $0 \leq v < B$ and let X be a random variable with $X \leftarrow_R [A]$. Let $Y = X \bmod B$. The statistical distance of the distribution of Y and the uniform distribution over $\mathbb{Z}_B$ is at most v/A. Let $Y' = \lfloor X/B \rfloor$. The statistical distance of the uniform distribution over $\{0, \ldots, \pi\}$ and the distribution of Y' is at most $1/(\pi + 1)$.*

Lemma 3. *Let $B_1, \ldots, B_r \leftarrow_R \langle g \rangle$, $\langle \mathbb{G}, g, M, k, \zeta \rangle \leftarrow S_{\mathsf{sg}}(1^\nu)$ be a safeguard group distribution, and let $\mathcal{A}$ be a* PPT *that on input $\mathbb{G}, g, M, k, \zeta, B_1, \ldots, B_r$ it outputs integers $e_1, \ldots, e_r, t$ and $y \in \mathbb{G}$ such that with probability α: $|t| > 1$ and $\prod_{i=1}^r B_i^{e_i} = y^t$ where t is a k-bit number and $\exists i : t \not| e_i$. Then the Strong-Root property is violated with probability at least $\alpha/(2r+1) - \eta$ where η is a function negligible in ν.*

5 The Portability of Generalized Schnorr Proofs

In this section we discuss the portability of Generalized Schnorr Proofs. In particular we will identify a wide class of input generators so that under the right conditions these protocols are portable.

GSP-specs. A generalized Schnorr proof (GSP) operates on a statement $\mathtt{t}$ that involves a number of groups and group elements ("bases") with public and secret exponents. To any such statement $\mathtt{t}$ we will associate the following:

i. A set of symbolic variables denoted by $\mathcal{X} = \{\alpha_1, \ldots, \alpha_r\}$ with $|\mathcal{X}| = r$.
ii. A sequence of group descriptions $\mathbb{G}_1, \ldots, \mathbb{G}_z$ as well as the descriptions of z subgroups $\zeta_1, \ldots, \zeta_z$ of $\mathbb{G}_1, \ldots, \mathbb{G}_z$ respectively, so that the exponent of each ζ_i is (at most) a k-bit integer. The description of the subgroup ζ_i will be typically given as a list of elements (i.e., these subgroups are small). It may be the case that $\zeta_i = \{1\}$.
iii. The group elements $A_{i,j} \in \mathbb{G}_i$ for $j = 0, \ldots, r$ where $A_{i,j}$ will be the base for the variable α_j in group $\mathbb{G}_i$.
iv. The range limits $L_j, R_j, L_j^{\mathsf{ext}}, R_j^{\mathsf{ext}} \in \mathbb{Z} \cup \{-\infty, \infty\}$ such that $L_j < R_j$, and $L_j^{\mathsf{ext}} \leq L_j, R_j \leq R_j^{\mathsf{ext}}$ for $j = 1, \ldots, r$.

Next we give an explicit syntax notation and semantics for specifying the language $\mathcal{L}$ that the prover wishes to convince the verifier the statement $\mathtt{t}$ belongs to. We define two languages $\mathcal{L}$ and $\mathcal{L}^{\mathsf{ext}}$:

$$\mathcal{L} = \Big\{ t \in \mathcal{L}_{\mathsf{in}} \mid \exists\, x_i \in \mathbb{Z} \,:\, \bigwedge_{i=1}^{z} \Big(\prod_{j=0}^{r} A_{i,j}^{x_j} = A_{i,0} \Big) \wedge \bigwedge_{j=1}^{r} \Big(x_j \in [L_j, R_j] \Big) \Big\}$$

$$\mathcal{L}_{\mathsf{ext}} = \Big\{ t \in \mathcal{L}_{\mathsf{in}} \mid \exists\, x_i \in \mathbb{Z} \,:\, \bigwedge_{i=1}^{z} \Big(\prod_{j=0}^{r} A_{i,j}^{x_j} = \zeta_i \cdot A_{i,0} \Big) \wedge \bigwedge_{j=1}^{r} \Big(x_j \in [L_j^{\mathsf{ext}}, R_j^{\mathsf{ext}}] \Big) \Big\}$$

We will use the following syntax to refer to a proof of knowledge for the language $\mathcal{L}$ whose soundness is only ensured in the extended language $\mathcal{L}_{\mathsf{ext}}$; we call this notation a GSP-spec τ.

$$\mathsf{PKspec}\Big(\mathcal{X} \,:\, \prod_{j=1}^{r} A_{1,j}^{\alpha_j} = A_{1,0} (\text{in } \mathbb{G}_1) \ldots \wedge\ \alpha_1 \in [L_1, R_1] \wedge \ldots \wedge \alpha_r \in [L_r, R_r] \Big)$$

$$\rightarrow \Big(\mathcal{X} \,:\, \prod_{j=1}^{r} A_{1,j}^{\alpha_j} = \zeta_1 \cdot A_{1,0} (\text{in } \mathbb{G}_1) \ldots \wedge \alpha_1 \in [L_1^{\mathsf{ext}}, R_1^{\mathsf{ext}}] \wedge \ldots \wedge \alpha_r \in [L_r^{\mathsf{ext}}, R_r^{\mathsf{ext}}] \Big)$$

Note that left-hand side of the above notation (i.e., the first line) is the statement of the proof whereas the right-hand side (namely, the second line) is the actual (extended) statement that will be guaranteed to hold (recall Definition 2). Note that in the extended statement the ranges $[L_j, R_j]$ will be extended to $[L_j^{\mathsf{ext}}, R_j^{\mathsf{ext}}]$ and the unit element of the group is extended to be any element in the (small) subgroup ζ_i for the i-th equation.

The specification allows for a wide range of proofs including polynomial relations among the secret and inequality statements of secrets. We refer to in the full version of the paper [12] for a discussion on what is covered by this specification and how it can be extended, in particular to include also $\vee$-connectives or tighter ranges.

GSP input generators. A GSP input generator $\Pi = \langle P_{\mathsf{in}}, V_{\mathsf{in}} \rangle$ that is consistent with a GSP-spec τ is a two party protocol that determines the parameters: z (the number of groups), r (the number of symbolic variables), k (a parameter related to group selection and the soundness property) and whose public output $\mathtt{t}$ includes the description of all groups, bases and ranges of the GSP-spec as described in the items (i)-(iv) above.

The Generalized Schnorr Protocol $\Sigma_\tau^{\mathsf{GSP}}$. For any GSP-spec τ one can design a Sigma protocol based on Schnorr's proof by introducing appropriate range checking and compensating for the fact that groups of unknown order are used with computations over the integers.

The protocol is based on two parameters k, l for free variables $\alpha_1, \ldots, \alpha_r$ such that α_j takes values in the range $[L_j, R_j]$. Below we set $m_j = R_j - L_j$. Suppose the prover is in possession of the witnesses $x_1, \ldots, x_r$; the prover selects first the random values $t_j \in_R [-2^{k+l} m_j, 2^{k+l} m_j]$ and computes the values $B_i = \prod_{j=1}^{r} A_{i,j}^{t_j}$. The prover terminates the first stage of computation by transmitting $B_1, \ldots, B_z$. The verifier selects $c \in_R \{0,1\}^k$ and responds by sending c to the prover. The prover, in response computes the integers $s_j = t_j - c \cdot (x_j - L_j)$ and sends them to the verifier. The verifier returns 1 if and only if for all $j \in \{1, \ldots, r\}$ it holds that $s_j \in [-2^{k+l} m_j - (2^k - 1) m_j, 2^{k+l} m_j]$ as well as for all $i \in \{1, \ldots, z\}$ it holds that $B_i \in \mathbb{G}_i$ and $\prod_{j=1}^{r} A_{i,j}^{s_j} =_{\mathbb{G}_i} B_i (A_{i,0}^{-1} \cdot \prod_{j=1}^{r} A_{i,j}^{L_j})^c$.

Portability of $\Sigma_\tau^{\mathsf{GSP}}$. We will next identify a class of input generators Π for a given GSP-spec τ over which $\Sigma_\tau^{\mathsf{GSP}}$ is portable as a zero-knowledge proof of knowledge. Recall that Π defines the respective inputs $(\mathtt{t}, w)$ for the prover and $\mathtt{t}$ for the verifier. We first describe the setting where some special care needs to be paid when arguing the security of $\Sigma_\tau^{\mathsf{GSP}}$. These settings involve variables that are "unsafe":

Definition 5. *(**Unsafe Variables**) For a GSP-spec τ, a symbolic variable $\alpha_j \in \mathcal{X}$ is called unsafe if it satisfies at least one of the following three conditions: (1) it is involved in an equation over a group $\mathbb{G}_i$ over a base element that is of unknown order to the verifier (i.e., the order of the base is not included in the group's description); (2) the range $[L_j, R_j]$ is non-trivial (i.e., it is not the range $(-\infty, +\infty)$); (3) the variable appears across various bases that have known but different order.*

The presence of unsafe variables may introduce problems in the knowledge extraction argument and make the protocol fail the soundness property. Still, unsafe variables can be tolerated provided they appear in conjunction to safeguard groups (cf. Definition 4). The following definition defines input-generators that are suitable for the $\Sigma_\tau^{\mathsf{ext}}$

protocol in the presence of unsafe variables. In a nutshell it says that for a GSP-input generator protocol Π, a certain group will be called a safeguard group *for* Π as long as there exists a simulator that playing the role of the verifier, it can "plug-in" a safeguard group generated by S_{sg} in black-box fashion in the interaction with P_{in} without P_{in} noticing, even if P_{in} is acting adversarially.

Definition 6. *For any* GSP*-input-generator protocol* $\Pi = \langle P_{\mathsf{in}}, V_{\mathsf{in}} \rangle$*, a group* $\mathbb{G}_i$ *and the bases* $A_{i,j_1}, \ldots, A_{i,j_v} \in \mathbb{G}_i$ *will be called respectively a* **safeguard group for** Π *and its* **safeguard bases** *there exists a polynomial-time simulator* S_V *s.t. for any adversarial party* P^*_{in} *in the protocol* Π*,* S_V *receives as input* $\langle \mathbb{G}, g, M, k, \zeta, g_1, \ldots, g_v \rangle$ *where* $\langle \mathbb{G}, g, M, k, \zeta \rangle \leftarrow S_{\mathsf{sg}}(1^\nu)$ *and* $g_\ell = g^{s_\ell}$ *with* $s_\ell \xleftarrow{\$} [M]$*, and satisfies the property that the input* t *produced by the interaction of* P^*_{in} *and* S_V *contains a group* $\mathbb{G}_i$ *and bases* $A_{i,j_1}, \ldots, A_{i,j_v}$ *that satisfy* $\mathbb{G}_i = \mathbb{G}$ *and* $A_{i,j_1} = g_1, \ldots, A_{i,j_v} = g_v$ *and the view of* P^*_{in} *when interacting with* V_{in} *is indistinguishable from the view of* P^*_{in} *when interacting with* S_V*.*

An equation $\prod_{j=1}^{r} A_{i,j}^{\alpha_j} = A_{i,0}$ over a safeguard group for Π will be called a "safeguarding equation." Armed with the above we next identify a class of input generators for which the generalized Schnorr proof $\Sigma_\tau^{\mathsf{GSP}}$ is portable.

Theorem 1. *(Portability of Generalized Schnorr Proofs) Let* τ *be a* GSP*-spec. The protocol* $\Sigma_\tau^{\mathsf{GSP}}$ *is portable for honest verifiers, for all input generators* Π *consistent with* τ *provided that (I) the generated input* $t \in \mathcal{L}_{\mathsf{in}}$ *has no unsafe variable, or (II) the five following conditions hold: (i) Each unsafe variable appears at least once as an exponent over a safeguard base. (ii) There is an ordering* $i_1, \ldots, i_z$ *of all the equations so that (1)* i_1 *is a safeguarding equation with all its free variables over safeguard bases, and (2) in safeguarding equation* i_w *for* $w > 1$ *it holds that all free variables of equation* i_w *appear over safeguard bases or have appeared at least once in a previous safeguarding equation. (iii) If* $\mathbb{G}_i$ *is a safeguard group then it has description* $\langle \mathbb{G}_i, g_i, M_i, k, \zeta_i \rangle$ *(i.e., all safeguard groups share the same* k*). (iv)* $L_j^{\mathsf{ext}} = L_j - 2^{k+l+2}(R_j - L_j)$ *and* $R_j^{\mathsf{ext}} = R_j + 2^{k+l+2}(R_j - L_j)$*. (v) The knowledge error* κ *is* $\mathsf{c} \cdot (2^{-k} + r \cdot \mathsf{Adv}_{\mathsf{root}})$ *for a suitable* $\mathsf{c} \in \mathbb{R}$ *and the zero-knowledge distance is* $\epsilon = r \cdot 2^{-l}$*.*

Example. Suppose that V_{in} selects an RSA-modulus n which is a multiple of two safe primes, a quadratic residue base $g \in \mathbb{Z}_n^*$ as well as $h \xleftarrow{\$} \langle g \rangle$. V_{in} transmits n, g, h to P_{in}. In turn, P_{in} sends $y = g^u h^v \bmod n$ where $u \xleftarrow{\$} [\lceil \frac{n}{4} \rceil]$ and $v \in [2^e]$ for some $e \in \mathbb{N}$. The input[4] t generated by $P_{\mathsf{in}}, V_{\mathsf{in}}$ in this case is the vector $\langle n, g, h, y \rangle$. Suppose now that the prover P wishes to demonstrate to the verifier V that she knows u, v in their respective ranges such that $y = g^u h^v \bmod n$. It is easy to see that $\mathbb{Z}_n^*$ can play the role of a safeguard group for the input generator described above with $\zeta = \{-1, +1\}$ and that the conditions of Theorem 1 are satisfied, thus the protocol $\Sigma_\tau^{\mathsf{GSP}}$ can be used to ensure to V that $y = \pm g^u h^v \bmod n$ and $u \in [-E_u, \lceil \frac{n}{4} \rceil + E_u], v \in [-E_v, 2^e + E_v]$ where $E_u = 2^{k+l+2} \cdot \lceil \frac{n}{4} \rceil, E_v = 2^{k+l+2+e}$.

[4] In this simple example, it could be that y leaks some information about u, v to V_{in} (which recall it may be an entity that includes more parties beyond the verifier); this does not affect the zero-knowledge property over this input generator which — as it is the case with regular ZK proofs — is concerned only with information leaks during the P, V interaction.

6 Unconditionally Portable Protocols for GSP-specs

Theorem 1 of the previous section describes a class of input-generators for which the generalized Schnorr proof protocol can be used in a safe way. Nevertheless, it may be very well the case that we would like to use a proof for a GSP-spec outside this class of input generators. In the remaining of the section we describe an efficient protocol enhancement to the basic generalized Schnorr protocol that is unconditionally portable.

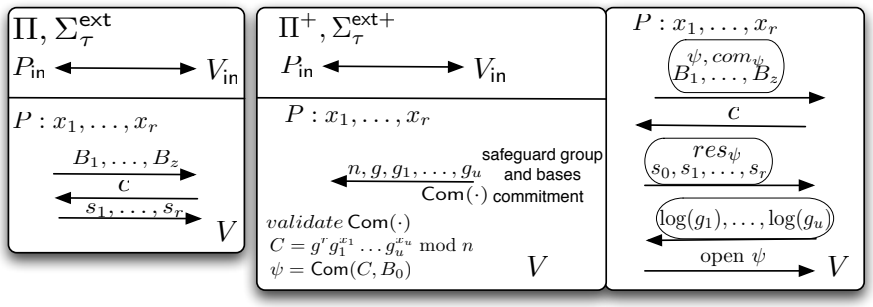

Fig. 1. Illustration of the transformation of $\Sigma_\tau^{\mathsf{ext}}$ over input generator Π to the $\Sigma_\tau^{\mathsf{ext}+}$.

The $\Sigma_\tau^{\mathsf{ext},+}$ protocol. Consider any input generator Π for which Theorem 1 does not apply, i.e., $(\Pi, \Sigma_\tau^{\mathsf{ext}})$ is not a zero-knowledge proof over Π. We next show one modification of $\Sigma_\tau^{\mathsf{ext}}$ into a protocol $\Sigma_\tau^{\mathsf{ext}+}$ so that $\Sigma_\tau^{\mathsf{ext}+}$ is a protocol that is universally portable as a zero-knowledge proof.

The protocol $\Sigma_\tau^{\mathsf{ext}+}$ operates as follows: The verifier first selects a safeguard group $\langle \mathbb{Z}_n^*, g, M = \lfloor n/4 \rfloor, k, \mathbb{V} = \{-1, 1\} \rangle$ where $\langle g \rangle = QR(n)$ together with a number of safeguard bases $g_1, \ldots, g_u \in \langle g \rangle$ where u is the number of variables that are unsafe. We will denote the discrete-logarithm values of g_ℓ base g as ρ_ℓ. The verifier also selects a prime P such that $(P-1)/2$ is also prime and satisfies $(P-1)/2 > n$ as well as two elements of order $(P-1)/2$ in $\mathbb{Z}_P^*$ denoted by G, H where H is randomly selected from $\langle G \rangle$. When these elements are received the prover will check that $P, (P-1)/2 \in \mathsf{Prime}$, $(P-1)/2 > n$ and that $G, H \in QR(P)$ (i.e., that $H \in \langle G \rangle$). We denote $\mathsf{Adv}_{\mathtt{DLOG}}$ an upper bound on the probability that any polynomial-time bounded algorithm has in returning $\log_G(H)$ given G, H, P. Next, the prover computes a commitment of the form $C = g^r g_1^{x_1} \ldots g_u^{x_u} (\mathrm{mod} n)$ (which is an extended Pedersen commitment over the safeguard group); note that $r \stackrel{\mathsf{c}}{\leftarrow} [2^{l+3} M]$ where l is the security parameter related to the zero-knowledge distance and $x_1, \ldots, x_u$ are the witnesses of P. Intuitively, what will happen next can be interpreted as follows: the prover and the verifier will include in the GSP-spec τ the safeguarding equation $(C = g^r g_1^{x_1} \ldots g_u^{x_u} (\text{ in } \mathbb{Z}_n^*))$ as one of the equations that are needed to be shown (we call the extended GSP-spec τ^+) but the prover will not reveal C. This is because the parameters of the safeguard group were

selected by the verifier and thus the prover is at risk of revealing some information about the witnesses.

Instead, the (P, V) protocol interaction for τ^+ will be modified as follows: the prover P will make a commitment ψ_1 to the value C denoted by $\psi_1 = G^{r^*} H^C \bmod P$. Similarly, the prover P will not submit the value B_0 (that corresponds to the commitment equation $(C = g^r g_1^{x_1} \dots g_u^{x_u}(\text{ in } \mathbb{Z}_n^*))$); instead it will submit a commitment $\psi_2 = G^{r_0^*} H^{B_0} \bmod P$. We call $\psi = (\psi_1, \psi_2)$. Next, the prover P will need to show that ψ is well-formed; this is easy as ψ_1, ψ_2 are Pedersen commitments, so it suffices to prove knowledge of r^* and C in ψ_1 and prove knowledge of r_0^* and B_0 in ψ_2. We denote the Σ proof for the ψ commitment as com_ψ, c, res_ψ. These additional proofs can be composed in parallel AND composition with the GSP protocol $\Sigma_\tau^{\mathsf{GSP}}$ and do not incur any additional round complexity. After the verifier receives all values and accepts the proofs (except for the equation over the safeguard group), it submits to the prover the values $\rho_1, \dots, \rho_u$ who in turn checks whether $g_\ell = g^{\rho_\ell}$. In this case, the prover opens the commitments ψ_1, ψ_2, and now the verifier is able to complete the verification as described in the $\Sigma_\tau^{\mathsf{ext}}$ protocol. We illustrate the transformation in figure 1.

Remark. This transformation generalizes and improves the setting of the Σ^+ proof method introduced in [5]; it obviates the need of random oracles (their soundness argument was in the random oracle model). We note that if the number of rounds is at premium then it is possible to reduce them to 3 by giving up on other aspects of the protocol in terms of security or efficiency. Specifically, one can either have the verifier demonstrate to the prover that the safeguard group is properly selected in an "offline" stage (that will not be counting towards the rounds of the actual protocol) or assuming the existence of an auxiliary input that is honestly distributed (an approach shown in [4]).

We next prove our protocol secure for a type of "partly honest" verifiers that may operate maliciously in the safeguard group selection (i.e., the first move of the $\Sigma_\tau^{\mathsf{ext}+}$ protocol) but still select the challenge honestly (in the third move of the protocol). We choose to do this for ease of presentation as there are standard techniques that can be applied to port the protocol to the entirely malicious verifier setting (much like how an honest verifier zero-knowledge protocol can be ported to the zero-knowledge setting).

Theorem 2. *For any* GSP*-spec τ and any consistent input generator Π, the protocol $\Sigma_\tau^{\mathsf{ext},+}$ is an (unconditionally portable) zero-knowledge proof of knowledge over Π against partly honest verifiers for the same $L_\ell^{\mathsf{ext}}, R_\ell^{\mathsf{ext}}$ parameters as Theorem 1, knowledge error $\kappa = \mathsf{c}(2^{-k} + \mathsf{Adv}_{\mathtt{DLOG}} + r \cdot \mathsf{Adv}_{\mathsf{root}})$ for some $\mathsf{c} \in \mathbb{R}$ and zero-knowledge distance $(r+1)2^{-l}$.*

7 Demonstrative Applications and Extensions

In the full version of the paper available in [12] we provide two demonstrative applications of our framework as well as a number of possible extensions to it. The full version also includes proofs for all the statements in this version.

Acknowledgements

The authors thank Endre Bangerter for helpful discussions on the subject.

References

1. Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO ' 2000*, volume 1880 of *Lecture Notes in Computer Science*. International Association for Cryptologic Research, Springer, 2000.
2. Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. Remarks on "analysis of one popular group signature scheme" in asiacrypt 2006. Cryptology ePrint Archive, Report 2006/464, 2006. `http://eprint.iacr.org/`.
3. Giuseppe Ateniese, Dawn Xiaodong Song, and Gene Tsudik. Quasi-efficient revocation in group signatures. In Matt Blaze, editor, *Financial Cryptography*, volume 2357 of *Lecture Notes in Computer Science*, pages 183–197. Springer, 2002.
4. Endre Bangerter. *On Efficient Zero-Knowledge Proofs of Knowledge*. PhD thesis, Ruhr U. Bochum, 2005.
5. Endre Bangerter, Jan Camenisch, and Ueli M. Maurer. Efficient proofs of knowledge of discrete logarithms and representations in groups with hidden order. In *Public Key Cryptography*, pages 154–171, 2005. Corrected full version: `http://www.zurich.ibm.com/~jca/papers/bacama05.pdf`.
6. Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420. Springer, 1992.
7. Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In Bart Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444. Springer Verlag, 2000.
8. Emmanuel Bresson and Jacques Stern. Efficient revocation in group signatures. In Kwangjo Kim, editor, *Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 190–206. Springer, 2001.
9. Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *Proc. 11th ACM Conference on Computer and Communications Security*, pages 225–234. acm press, 2004.
10. Laurent Bussard, Refik Molva, and Yves Roudier. History-based signature or how to trust anonymous documents. In Christian D. Jensen, Stefan Poslad, and Theodosis Dimitrakos, editors, *iTrust*, volume 2995 of *Lecture Notes in Computer Science*, pages 78–92. Springer, 2004.
11. Laurent Bussard, Yves Roudier, and Refik Molva. Untraceable secret credentials: Trust establishment with privacy. In *PerCom Workshops*, pages 122–126. IEEE Computer Society, 2004.
12. Jan Camenisch, Aggelos Kiayias, and Moti Yung. On the portability of generalized schnorr proofs. Technical report, Cryptology ePrint Archive, 2009.
13. Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 126–144, 2003.
14. Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups. *Lecture Notes in Computer Science*, 1294:410–424, 1997.

15. Jan Leonhard Camenisch. *Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem*. PhD thesis, ETH Zürich, 1998. Diss. ETH No. 12520, Hartung Gorre Verlag, Konstanz.
16. Zhengjun Cao. Analysis of one popular group signature scheme. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology - ASIACRYPT 2006, 12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, December 3-7, 2006, Proceedings*, volume 4284 of *Lecture Notes in Computer Science*, pages 460–466. Springer, 2006.
17. Agnes Hui Chan, Yair Frankel, and Yiannis Tsiounis. Easy come - easy go divisible cash. In Kaisa Nyberg, editor, *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998*, volume 1403 of *Lecture Notes in Computer Science*, pages 561–575. Springer, 1998.
18. Agnes Hui Chan, Yair Frankel, and Yiannis Tsiounis. Easy come - easy go divisible cash. GTE Technical Report, `http://www.ccs.neu.edu/home/yiannis/pubs.html`, 1998.
19. Ronald Cramer and Victor Shoup. Signature schemes based on the strong rsa assumption. *ACM Trans. Inf. Syst. Secur.*, 3(3):161–185, 2000.
20. Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *EUROCRYPT*, pages 418–430, 2000.
21. Ivan Damgard and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *ASIACRYPT: Advances in Cryptology – ASIACRYPT: International Conference on the Theory and Application of Cryptology*, Lecture Notes in Computer Science, pages 125–142. Springer-Verlag, 2002.
22. Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Proceedings of CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer Verlag, 1986.
23. Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In Burton S. Kaliski, Jr., editor, *Advances in Cryptology – CRYPTO ' 97*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 1997.
24. Jun Furukawa and Shoko Yonezawa. Group signatures with separate and distributed authorities. In Carlo Blundo and Stelvio Cimato, editors, *SCN*, volume 3352 of *Lecture Notes in Computer Science*, pages 77–90. Springer, 2004.
25. Matthieu Gaud and Jacques Traoré. On the anonymity of fair offline e-cash systems. In Rebecca N. Wright, editor, *Financial Cryptography*, volume 2742 of *Lecture Notes in Computer Science*, pages 34–50. Springer, 2003.
26. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *STOC '87: Proceedings of the nineteenth annual ACM conference on Theory of computing*, pages 218–229, New York, NY, USA, 1987. ACM Press.
27. Oded Goldreich. *The Foundations of Cryptography, Vol. 2*. Cambridge University Press, 1999.
28. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, February 1989.
29. Sébastien Kunz-Jacques, Gwenaëlle Martinet, Guillaume Poupard, and Jacques Stern. Cryptanalysis of an efficient proof of knowledge of discrete logarithm. In *PKC '06, New York*, 2006.
30. Tri Van Le, Khanh Quoc Nguyen, and Vijay Varadharajan. How to prove that a committed number is prime. In Kwok-Yan Lam, Eiji Okamoto, and Chaoping Xing, editors, *ASIACRYPT*, volume 1716 of *Lecture Notes in Computer Science*, pages 208–218. Springer, 1999.

31. Anna Lysyanskaya and Zulfikar Ramzan. Group blind digital signatures: A scalable solution to electronic cash. In Rafael Hirschfeld, editor, *Financial Cryptography*, volume 1465 of *Lecture Notes in Computer Science*, pages 184–197. Springer, 1998.
32. Philip D. MacKenzie and Michael K. Reiter. Two-party generation of dsa signatures. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 137–154. Springer, 2001.
33. Einar Mykletun, Maithili Narasimha, and Gene Tsudik. Signature bouquets: Immutability for aggregated/condensed signatures. In Pierangela Samarati, Peter Y. A. Ryan, Dieter Gollmann, and Refik Molva, editors, *ESORICS*, volume 3193 of *Lecture Notes in Computer Science*, pages 160–176. Springer, 2004.
34. Toru Nakanishi, Mitsuaki Shiota, and Yuji Sugiyama. An efficient online electronic cash with unlinkable exact payments. In Kan Zhang and Yuliang Zheng, editors, *ISC*, volume 3225 of *Lecture Notes in Computer Science*, pages 367–378. Springer, 2004.
35. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, March 2000.
36. Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge proof systems. In Carl Pomerance, editor, *CRYPTO*, volume 293 of *Lecture Notes in Computer Science*, pages 52–72. Springer, 1987.
37. C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.
38. Dawn Xiaodong Song. Practical forward secure group signature schemes. In *Proc. 8th ACM Conference on Computer and Communications Security*, pages 225–234. ACM press, November 2001.
39. Willy Susilo and Yi Mu. On the security of nominative signatures. In Colin Boyd and Juan Manuel González Nieto, editors, *ACISP*, volume 3574 of *Lecture Notes in Computer Science*, pages 329–335. Springer, 2005.
40. Chunming Tang, Zhuojun Liu, and Mingsheng Wang. A verifiable secret sharing scheme with statistical zero-knowledge. Cryptology ePrint Archive, Report 2003/222, 2003. `http://eprint.iacr.org/`.
41. Patrick P. Tsang and Victor K. Wei. Short linkable ring signatures for e-voting, e-cash and attestation. In Robert H. Deng, Feng Bao, HweeHwa Pang, and Jianying Zhou, editors, *ISPEC*, volume 3439 of *Lecture Notes in Computer Science*, pages 48–60. Springer, 2005.
42. Patrick P. Tsang, Victor K. Wei, Tony K. Chan, Man Ho Au, Joseph K. Liu, and Duncan S. Wong. Separable linkable threshold ring signatures. In Anne Canteaut and Kapalee Viswanathan, editors, *INDOCRYPT*, volume 3348 of *Lecture Notes in Computer Science*, pages 384–398. Springer, 2004.
43. Victor K. Wei. Tracing-by-linking group signatures. In Jianying Zhou, Javier Lopez, Robert H. Deng, and Feng Bao, editors, *ISC*, volume 3650 of *Lecture Notes in Computer Science*, pages 149–163. Springer, 2005.