8-2022

Resumable zero-knowledge for circuits from symmetric key primitives

Handong ZHANG

Puwen WEI

Haiyang XUE
Singapore Management University, haiyangxue@smu.edu.sg

Yi DENG

Jinsong LI

See next page for additional authors

Citation

ZHANG, Handong; WEI, Puwen; XUE, Haiyang; DENG, Yi; LI, Jinsong; WANG, Wei; and LIU, Guoxiao. Resumable zero-knowledge for circuits from symmetric key primitives. (2022). *Proceedings of the 27th Australasian Conference, ACISP 2022 Wollongong, Australia, November 28-30*. 375-398.

Available at: https://ink.library.smu.edu.sg/sis_research/9202

Author

Handong ZHANG, Puwen WEI, Haiyang XUE, Yi DENG, Jinsong LI, Wei WANG, and Guoxiao LIU

Resumable Zero-Knowledge for Circuits from Symmetric Key Primitives

Handong Zhang[1,2], Puwen Wei[1,2(✉)], Haiyang Xue[3], Yi Deng[4,5], Jinsong Li[1,2], Wei Wang[1,2], and Guoxiao Liu[1,2]

[1] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China
{hdzhang,jsli,liuguoxiao}@mail.sdu.edu.cn, {pwei,weiwangsdu}@sdu.edu.cn
[2] School of Cyber Science and Technology, Shandong University, Qingdao, China
[3] The University of Hong Kong, Pokfulam, Hong Kong, China
haiyangxc@gmail.com
[4] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
deng@iie.ac.cn
[5] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

Abstract. Consider the scenario that the prover and the verifier perform the zero-knowledge (ZK) proof protocol for the same statement multiple times sequentially, where each proof is modeled as a session. We focus on the problem of how to resume a ZK proof efficiently in such scenario. We introduce a new primitive called *resumable honest verifier zero-knowledge proof of knowledge* (resumable HVZKPoK) and propose a general construction of the resumable HVZKPoK for circuits based on the "MPC-in-the-head" paradigm, where the complexity of the resumed session is less than that of the original ZK proofs. To ensure the knowledge soundness for the resumed session, we identify a property called extractable decomposition. Interestingly, most block ciphers satisfy this property and the cost of resuming session can be reduced dramatically when the underlying circuits are implemented with block ciphers. As a direct application of our resumable HVZKPoK, we construct a post quantum secure stateful signature scheme, which makes Picnic3 suitable for blockchain protocol. Using the same parameter setting of Picnic3, the sign/verify time of our subsequent signatures can be reduced to 3.1%/3.3% of Picnic3 and the corresponding signature size can be reduced to 36%. Moreover, by applying a parallel version of our method to the well known Cramer, Damgård and Schoenmakers (CDS) transformation, we get a compressed one-out-of-N proof for circuits, which can be further used to construct a ring signature from symmetric key primitives only. When the ring size is less than 2^4, the size of our ring signature scheme is only about 1/3 of Katz et al.'s construction.

Keywords: Resumable · Honest verifier zero-knowledge · MPC-in-the-head · Stateful signature · Ring signature · Blockchain

K. Nguyen et al. (Eds.): ACISP 2022, LNCS 13494, pp. 375–398, 2022.
https://doi.org/10.1007/978-3-031-22301-3_19

1 Introduction

Zero-knowledge (ZK) proofs [29,30] and their non-interactive form (NIZK) [12], which allow a prover to convince a verifier of a certain statement without revealing any additional information, are among the most fundamental and important cryptographic primitives. It is known that there exists ZK proof [29] for any NP language, while the resulting construction is rather inefficient. A lot of works have been done to propose efficient (NI)ZK proofs for arbitrary circuits or specific algebraic computation, e.g., zk-SNARKs [9,25], which have short proof for a statement. Some works focus on the efficient composition of ZK proofs for several statements [1,20,24]. Other works such as [14,34,40] investigate the batch ZK proofs, which enable many instances of the same relation to be proved and verified simultaneously. Amongst most of those constructions, the randomness and the related transcripts are "recycled" or compressed in *one* execution of the resulting ZK protocol in order to reduce the cost of computation or communication.

Notice that one common case of ZK proof, however, is proving the same statement repeatedly multiple times. For example, a user may be required to provide digital signatures on different messages periodically, where each signature can be thought of as one execution of the NIZK proof of knowledge of the signing key [7]. One typical application is validating the authenticity of firmware updates for IoT devices. The manufacturer periodically offers firmware updates and the corresponding signatures, and the IoT devices verify these signatures in order to ensure the authenticity of the updates. Another direct application of ZK is the identification protocol, which could be used by a company to determine the identity of a user each time he tries to access company resources.

Hence, it is worth considering the efficiency of ZK protocols in a scenario where the prover and the verifier need to run the ZK proof of a statement many times (sequentially). In practice, the state information derived from previous sessions could be reused in the following sessions to achieve significant savings in processing load and bandwidth, e.g., session resumption of TLS 1.3 [41]. It is desired that the ZK protocol for the subsequent sessions be much more efficient than that of the original one. Therefore, a natural question is that

How can we resume a session of ZK proofs efficiently?

An intuitive way is to reuse the information of previous ZK sessions (of the same statement). In fact, similar issues have been considered in the research of NIZK. A series of works explored how to reuse the common reference string (CRS) of NIZK for multiple theorems and multiple provers [12,22,33], which implies the case of CRS reuse in multiple sessions (or executions). On the interactive ZK protocols, how to securely reuse previous transcripts among different sessions is, however, more subtle and tends to result in a breach of security. For instance, the witness in many Σ protocols can be extracted when the same commitments are reused in different sessions (with different challenges).

In another recent line of works, researchers have shown how to use secure multiparty computation (MPC) to obtain (NI)ZK proofs, and further quantum-resistant signatures. Ishai et al. [35] showed how to use the so-called "MPC-in-the-head" approach to obtain public-coin ZK proofs. Their scheme was further improved by [18,26] to obtain quantum-resistant signature via Fiat-Shamir transformation [23]. The resulting signature Picnic [16], which was submitted to the NIST post-quantum standardization effort, is very competitive, since its security is based entirely on symmetric-key primitives. But it is still less efficient than lattice-based CRYSTALS-DILITHIUM [4] and multivariate-based Rainbow [21]. So we would like to ask whether we could reduce the overall complexity of Picnic when considering multiple sequential signing requests. In other words, *how can we resume the signing/verifying procedure of Picnic efficiently?*

1.1 Our Contributions

We introduce the notion of *resumable honest verifier ZK proof of knowledge* (resumable HVZKPoK) to capture the security and efficiency of HVZKPoK in the scenario of session resumption. In that scenario, the prover and the verifier can perform the ZK proofs multiple times sequentially, where each proof is executed in a session. Informally, we say an HVZKPoK is resumable if it satisfies (1) *resumable zero-knowledge*, i.e., no additional information about the witness is revealed from *all* sessions; (2) *resumable knowledge soundness*, i.e., the witness can be extracted from *every* session; (3) *resumption efficiency*, i.e., the cost of the resumed session in terms of both computation and communication should be much less than that of the initial session (or the original ZK proofs).

The main challenge of constructing resumable HVZKPoK is to achieve resumable knowledge soundness while preserving resumption efficiency, since removing or reusing partial transcripts of the original ZK proofs would undermine or break its soundness property in general. To overcome that problem, we investigate the "MPC-in-the-head" paradigm in the preprocessing model proposed by Katz, Kolesnikov and Wang (KKW) [38], and find that their proofs can be separated according to the decomposition of circuits, where the proofs for the corresponding partial circuits can be further rerandomized without breaking the security. By making use of such separability of the KKW proofs, we provide a general construction of the resumable HVZKPoK. The main idea is that the underlying circuits are decomposed into two parts, where the proofs for the partial circuits with smaller size can be rerandomized. Once the initial session of proofs for the entire circuits is finished, both the prover and the verifier can resume a session by running the rerandomized proofs for the partial circuits only. Since the cost of the KKW proofs is closely related to the number of the AND gates of the circuits, the cost of the resumed session is reduced significantly due to the size of the partial circuits.

Notice that only proofs for the partial circuits usually cannot achieve knowledge soundness implied by the proofs for the entire circuits. To mitigate that problem, we identify a property called extractable decomposition, which guarantees that the witness can be extracted from the inputs of the separated partial circuits. Interestingly, most block ciphers satisfy this property. In addition, the circuits of block ciphers can be decomposed such that the separated partial circuits have no AND gates. Hence, the cost for resuming sessions can be made very small when implemented with block ciphers. By applying the Fiat-Shamir heuristic [23], our resumable HVZKPoK can be transformed into a stateful post-quantum signature scheme. Comparing with the typical chain-based stateful signature [37], the main advantage of our scheme is that, once the initial signature has been generated, the subsequent signatures are much more efficient than the initial one.

We implement our signature and give a comparison with Picnic3 [36]. The sign/verify time of our subsequent signatures can be reduced to 3.1%/3.3%-9.2%/8.8% of Picnic3 and the corresponding signature size can be reduced to 36.0%-38.1%. (For the fixed verifier, the size of the state information needed to be stored is about 2.9 KB-10.9KB.) Although the complexity of our first signature is slightly higher than Picnic3, it is worthy for the reducing cost of subsequent signatures. In particular, our stateful signatures make the symmetric-based signatures, such as Picnic3, suitable for the post-quantum secure blockchain protocol. That is, the previous signatures (or the states) can be stored in the history blocks efficiently and publicly. The verifier only needs to check the validity of the current signature without checking all previous signatures, since the validity of the previous signatures is implied by the consistency of the underlying consensus protocol.

Moreover, applying our method to the Cramer, Damgård and Schoenmakers (CDS) technique [20], we construct a compressed one-out-of N proof, where most of the transcripts for the simulation in CDS technique can be removed. Furthermore, we can construct a ring signature from symmetric key primitives using our compressed one-out-of N proof (without resorting to the Merkle-tree based accumulator). Comparing with the ring signatures from symmetric key primitives proposed by [38], the size of our ring signature is about 1/3 of [38] when the ring size is less than 2^4.

1.2 Related Works

Zero-Knowledge from Symmetric Primitives. Most efficient ZK proofs exist for a restricted set of languages, e.g., languages relying on algebraic structures. To construct efficient ZK proofs for a larger class of languages, many works focus on ZK proofs for arbitrary circuits [8–11,13,15,19,25,31,32,39,43], which have relatively short proofs size and verification time. However, most efficient constructions require a trusted setup or rely on assumptions, which are insecure in the quantum setting.

Ishai et al. [35] introduced a novel way of constructing ZK proofs, called "MPC-in-the-head", which is based on secure multi-party computation (MPC) protocols. Following the idea of [35], Giacomelli et al. [26] proposed ZKBoo which supports efficient non-interactive (NI) ZKPoKs for arbitrary circuits. Chase et al. [18] improved the performance of ZKBoo and proposed ZKB++, which is used to construct the post-quantum secure signature scheme, called Picnic. Compared with other post-quantum secure signature candidates, Picnic relies on the security of the underlying symmetric-key primitives instead of structured hardness assumptions. Although Picnic has good performance on the speed when implemented on hardware, it has a large signature size which is linear in the size of the circuits. Ames et al. [3] proposed Ligero with sublinear proof size, which asymptotically outperforms ZKBoo and ZKB++. Katz et al. [38] instantiated the MPC-in-the-head paradigm in the preprocessing model, which can reduce the number of parallel repetitions, and provided an improved version of Picnic, called Picnic2. Guilhem et al. [42] applied the "MPC-in-the-head with preprocessing" approach to the arithmetic circuit of AES and implemented a signature scheme, called BBQ, whose security is based on AES. Baum et al. [6] proposed a novel way to construct an AES-based signature scheme, called Banquet, which reduces the signature size compared with BBQ and its implementation results show that Banquet can be made almost as efficient as Picnic2 . Baum and Nof [5] incorporated the "sacrificing" paradigm into "MPC-in-the-head" to reduce the proof size for arithmetic circuits. Kales and Zaverucha [36] made further optimizations and presented a new parameter set for Picnic2, called Picnic3. Goel et al. [27] introduced a general framework for constructing Σ-protocols of disjunctive proof which can be used to implement ring signature. Recently, Goel et al. [28] proposed a novel technique for efficiently adding set membership proofs to any MPC-in-the-head based ZK protocol and the resulting ring signatures outperform Katz et al.'s construction by a factor of 5 to 8.

2 Preliminaries

Notations. Let $[n]$ denote $\{1, \ldots, n\}$ and κ denote the security parameters. Let C and C' be the boolean circuit representation of F and f, respectively, where C and C' consists of XOR and AND gates. Let $|C|$ denotes the number of AND gates in the circuit C and $|C_{in/out}|$ denotes the number of input/output wires of C. Let $L_R \subseteq \{0,1\}^*$ be an NP language and R be the related NP-relation for circuit C. $A \approx_c B$ denotes computational indistinguishability between distributions A and B. Let Com denote a commitment scheme. A commitment to a message m is denoted as $\mathsf{com} = \mathsf{Com}(m; r)$ where $r \in \{0,1\}^\kappa$ is chosen uniformly. We say Com is secure if it satisfies the following properties: (1) *Hiding*: $\mathsf{Com}(m; r)$ reveals noting about m; (2) *Binding*: it is hard to find two messages $m \neq m'$ such that $\mathsf{Com}(m; r) = \mathsf{Com}(m'; r')$. Let H denote the hash function. We say H is collision-resistant if the probability that any PPT adversary finds x and x' such that $H(x) = H(x')$ and $x \neq x'$ is negligible.

2.1 MPC-in-the-head with Preprocessing

MPC-in-the-head paradigm [35] is a novel technique to construct ZK proofs from MPC protocols. Suppose the statement to be proven is (C, y), where $C(w) = y$ and w is the witness. Following the MPC-in-the-head paradigm, the prover simulates an MPC protocol which evaluates the circuit C among all the parties *in his head* and the input of each party is a secret sharing of the witness w. The prover then commits to the views of each party in the execution of the MPC protocol. The verifier randomly chooses a subset of these commitments as the challenge. Once receiving the challenge, the prover opens the challenged commitments. The verifier checks the correctness and consistency of these views.

MPC-in-the-head with preprocessing (KKW) [38] improves MPC-in-the-head paradigm and the resulting scheme can achieve the required soundness with much shorter proofs. Loosely speaking, the KKW protocol has two phases, the *preprocessing* phase and the *online* phase. In the *preprocessing* phase, the prover generates random masks for each party, which are used to hide the witness. In the *online* phase, the prover simulates the execution of the MPC protocol using the masked shares of each party and the masked input (or the masked witness) of the circuit. Note that the verifier needs to challenge both phases. The main techniques of MPC-in-the-head with preprocessing are described below.

Let $[x]$ denote an n-out-of-n (XOR-based) secret sharing scheme of a bit x, i.e., $x = [x]_1 \oplus \cdots \oplus [x]_n$, where $[x]_i$ for $1 \leq i \leq n$ is the secret share. Suppose the underlying n-party MPC protocol is Π, which is executed by n parties $P_1, \cdots, P_n$. Let z_α denote the value of wire α of $C(w)$. z_α will be masked by a random bit λ_α, say, $\hat{z}_\alpha = z_\alpha \oplus \lambda_\alpha$. Each party P_i will hold $[\lambda_\alpha]_i$, which is a share of λ_α.

- **Preprocessing phase**. In the preprocessing phase, the prover generates the masks for each party P_i. More precisely, P_i is given the following values.
  - $[\lambda_\alpha]_i$ for each input wire α.
  - $[\lambda_\gamma]_i$ for the output wire γ of each AND gate.
  - $[\lambda_{\alpha,\beta}]_i$ for each AND gate with input wires α and β such that $\lambda_{\alpha,\beta} = \lambda_\alpha \cdot \lambda_\beta$.

  $[\lambda_\alpha]_i$ and $[\lambda_\gamma]_i$ can be generated using a pseudorandom generator (PRG) with a random seed seed_i, for $i = 1, \ldots, n$, where $[\lambda_\alpha]_1 \oplus \cdots \oplus [\lambda_\alpha]_n = \lambda_\alpha$ and $[\lambda_\gamma]_1 \oplus \cdots \oplus [\lambda_\gamma]_n = \lambda_\gamma$. Hence, seed_i instead of $\{[\lambda_\alpha]_i\}$ and $\{[\lambda_\gamma]_i\}$ is given to P_i so that the total proof size can be reduced. Notice that $[\lambda_{\alpha,\beta}]_n$ cannot be generated using seed_n only due to $\lambda_{\alpha,\beta} = \lambda_\alpha \cdot \lambda_\beta$. Actually, $n-1$ shares of $\lambda_{\alpha,\beta}$ are generated using PRG, while the share of P_n is computed by $[\lambda_{\alpha,\beta}]_n := \lambda_\alpha \lambda_\beta \oplus [\lambda_{\alpha,\beta}]_1 \oplus \cdots [\lambda_{\alpha,\beta}]_{n-1}$, which plays the role of "correction bits". Therefore, party P_n needs to be given $\mathsf{aux}_n = \{[\lambda_{\alpha,\beta}]_n\}$ for all AND gates in addition to seed_n.
- **Online phase**. During the online phase, each party P_i runs the underlying n-party MPC protocol Π to evaluate the circuit C gate-by-gate in topological order. For each gate with input wires α and β and output wire γ,
  - For an XOR gate, P_i can locally compute $\hat{z}_\gamma = \hat{z}_\alpha \oplus \hat{z}_\beta$ and $[\lambda_\gamma]_i = [\lambda_\alpha]_i \oplus [\lambda_\beta]_i$, since P_i already holds $\hat{z}_\alpha$, $[\lambda_\alpha]_i$, $\hat{z}_\beta$ and $[\lambda_\beta]_i$.
  - For an AND gate, P_i locally computes $[s]_i = \hat{z}_\alpha[\lambda_\beta]_i \oplus \hat{z}_\beta[\lambda_\alpha]_i \oplus [\lambda_{\alpha,\beta}]_i \oplus [\lambda_\gamma]_i$, publicly reconstructs s, and computes $\hat{z}_\gamma = s \oplus \hat{z}_\alpha \hat{z}_\beta$ which satisfies $\hat{z}_\gamma = z_\gamma \oplus \lambda_\gamma$. Note that party P_i holds $[\lambda_{\alpha,\beta}]_i$ and $[\lambda_\gamma]_i$ in addition to $\hat{z}_\alpha$, $[\lambda_\alpha]_i$, $\hat{z}_\beta$ and $[\lambda_\beta]_i$ for each AND gate.

Finally, each party P_i can compute $\hat{z}_\gamma$ for the output wire γ of the circuit, and the output value z_γ is computed as $z_\gamma = \hat{z}_\gamma \oplus \lambda_\gamma$, where λ_γ is reconstructed publicly.

Security of the underlying MPC protocol. [17, Lemma 6.1] proves that the underlying MPC protocol Π_{mpc} is secure against an all-but-one corruption in the semi-honest model by showing that there exists a simulator for the MPC protocol Π_{mpc} such that the real execution of Π_{mpc} is computational indistinguishable from the simulated execution of Π_{mpc} under the assumption of secure PRG.

3 Resumable HVZK Proof of Knowledge

Let R be an efficiently decidable binary NP-relation which is polynomially bounded, and L_R be the NP-language defined by R. That is, $\exists w$ such that $(x, w) \in R$ iff $x \in L_R$. In our setting, the prover and the verifier can sequentially perform the zero-knowledge proofs for L_R polynomially-many times, say $q(\kappa)$ times, where each proof is modeled as a session and the t-th session is denoted as $\mathsf{session}(t)$, for $t \in \{1, \ldots, q(\kappa)\}$. In each $\mathsf{session}(t)$, the prover aims to convince the verifier that he knows the witness w for statement x by running the HVZKPoK protocol $\Pi = \{(\mathcal{P}^{(t)}, \mathcal{V}^{(t)})\}$. Let $\mathcal{P}^{(t)} = \mathcal{P}(x, w, pr_t, ps_t)$ denote the prover's strategy of $\mathsf{session}(t)$, which takes as input the common-input x, witness w, prover's randomness pr_t and state ps_t. Here, ps_t is the prover's state after $\mathsf{session}(t-1)$. Similarly, let $\mathcal{V}^{(t)} = \mathcal{V}(x, vr_t, vs_t)$ denote the verifier's strategy of $\mathsf{session}(t)$, which takes as inputs the common-input x, verifier's randomness vr_t and state vs_t.

In this paper, we transform an "ordinary" HVZKPoK $\Pi' = (\mathcal{P}', \mathcal{V}')$ to a resumable HVZKPoK $\Pi = \{(\mathcal{P}^{(t)}, \mathcal{V}^{(t)})\}$, the security of which is more subtle. In particular, we have to ensure that the adversary who does not have the knowledge of the witness cannot convince the verifier in any session, even that the adversary can have access to the transcripts of all previous sessions. Consider the following game on soundness. The adversary A can invoke the "honest" prover to run Π for $x \in L_R$ with the verifier for polynomially-many sequential sessions, say, $\mathsf{session}(1), \ldots,$ $\mathsf{session}(q(\kappa) - 1)$, where A can get all the transcripts of these sessions. For the next session, say $\mathsf{session}$ $q(\kappa)$, A runs Π with the verifier for $x \in L_R$, trying to convince the verifier without the help of the "honest" prover. The soundness of the resumable HVZK is defined according the above game, which requires that A can win the game only with negligible probability. Formal definition of resumable HVZKPoK is described below.

Definition 1 (Resumable HVZK Proof of Knowledge). $\Pi = \{(\mathcal{P}^{(t)}, \mathcal{V}^{(t)})\}$ *is a resumable honest verifier zero-knowledge proof of knowledge for the relation* R *with soundness error* ξ *if the following properties hold:*

- **Completeness:** *If the prover and the verifier follow the protocol* $(\mathcal{P}^{(t)}, \mathcal{V}^{(t)})$ *on inputs* $x \in L_R$ *and witness* $w \in R_x$*, then the verifier always accepts in each* $\mathsf{session}(t)$*, for* $t \in \{1, \ldots, q(\kappa)\}$*.*

- **Resumable Honest Verifier Zero-Knowledge:** *Let* $\mathsf{view}_{\mathcal{V}}^{\mathcal{P}}(x, w)$ *be the transcripts of all the sessions run by the prover and the verifier. There exists a PPT simulator* Sim *such that* $\mathsf{Sim}(x) \approx_c \mathsf{view}_{\mathcal{V}}^{\mathcal{P}}(x, w)$ *for all* $x \in L_R$ *and* $w \in R_x$*.*
- **Resumable Knowledge Soundness:** *For each* $\mathsf{session}(t)$*, there exists a probabilistic knowledge extractor* $\mathcal{E}$*, such that for every* $\hat{\mathcal{P}}^{(t)}$ *and every* $x \in L_R$*, the algorithm* $\mathcal{E}$ *satisfies the following condition:*
  - *Let* $\delta_t(x)$ *denote the probability that the verifier accepts on input* x *for* $(\hat{\mathcal{P}}^{(t)}, \mathcal{V}^{(t)})$ *of* $\mathsf{session}(t)$*. If* $\delta_t(x) > \xi_t(x)$*, then upon input* $x \in L_R$ *and oracle access to* $\hat{\mathcal{P}}^{(t)}$*, the algorithm* $\mathcal{E}$ *outputs a valid witness* $w \in R_x$ *in expected number of steps bounded by* $O(\frac{1}{\delta_t(x) - \xi_t(x)})$*.*

  Here, ξ_t *denotes the soundness error of* $\mathsf{session}(t)$ *for* $t \in \{1, \ldots, q(\kappa)\}$*. Let* $\xi = \max\{\xi_1, \ldots, \xi_{q(\kappa)}\}$*.*
- **Resumption efficiency:** *For each* $\mathsf{session}(t)$ *with* $t > 1$*, the computational and communicational complexity of* $(\mathcal{P}^{(t)}, \mathcal{V}^{(t)})$ *should be less than that of the original HVZKPoK* $\Pi' = (\mathcal{P}', \mathcal{V}')$ *for* R*. That is, resumable HVZK proof of knowledge should be efficient in each resumed session.*

Here, the statement x of R is of the form (F, y), such that $(x, w) \in R$ iff $F(w) = y$, where F denotes a function. Let C be the circuit representation of F. So the statement can be rewritten as (C, y), such that $(x, w) \in R$ iff $C(w) = y$. As mentioned in Sect. 1, the function F needs to satisfy a special property called extractable decomposition, which is defined as follows.

Definition 2 (Extractable Decomposition). *Let* $F : \{0,1\}^\kappa \rightarrow \{0,1\}^{\kappa'}$ *be a function which has a decomposition as* $F = f \circ g$*. We say the decomposition is extractable if, for all* $x \in \{0,1\}^\kappa$*, there exists an efficient extractor* $\mathcal{E}_D$ *such that* $\mathcal{E}_D(g(x)) = x$*.*

Consider the case that $F(w) = \mathsf{Enc}(w, m)$, where $\mathsf{Enc}(w, m)$ is a block cipher with the private key w and the plaintext m. It is known that a typical block cipher consists of multiple rounds, where each round takes as inputs the output of previous round and the corresponding subkey (or round key) derived from the *master* key w. Note that the subkey schedule of most block ciphers is reversable, which implies most block ciphers naturally satisfy the property of extractable decomposition. Concretely, given a block cipher with n rounds, the first $n-1$ rounds as well as the key schedule can be taken as g, and the last round is taken as f. Suppose the output of $g(w)$ is w', which consists of the output of the $(n-1)$-th round and n-th round key k_n. f takes as input w' and outputs the final ciphertext. Obviously, w can be extracted from w', which implies the extractability of the decomposition.

4 General Construction for Resumable HVZKPoK

In this section, we present the general construction of resumable HVZKPoK from the KKW protocol. We first abstract the construction of the original KKW protocol [38] for $F(w) = y$, where F has a decomposition as $F = f \circ g$. Then, we show how to efficiently resume HVZKPoK for w' such that $f(w') = y$ and $w' = g(w)$. Recall that C and C' denote the circuit representation of F and f, respectively. So $F(w) = y$ and $f(w') = y$ can be rewritten as $C(w) = y$ and $C'(w') = y$, respectively.

4.1 KKW Protocol for F

The KKW protocol π^F for $C(w) = y$, i.e., $F(w) = y$, consists of the preprocessing phase π^F_{pre} and the online phase π^F_{on}. π^F_{pre} shows that the n parties' states are generated randomly and correctly by "cut-and-choose", and π^F_{on} ensures that each party's view in the MPC protocol are correct and consistent. Let M denote the number of repetitions for reducing the soundness error.
Preprocessing phase $\pi^F_{pre}(1^\kappa)$.

- Round 1. *Commit* to the masks of M instances.
  The prover prepares the masks λ_j of the MPC protocol for the circuit C as described in Sect. 2.1 for each instance $j \in [M]$. Since λ_j is determined by n parties' states $\{\mathsf{state}_{j,1}, \ldots, \mathsf{state}_{j,n}\}$, which are the random seeds and the n-th party's auxiliary information, the prover only needs to commit to those states. The corresponding commitments are denoted as com^F_{pre}. The prover sends com^F_{pre} to the verifier.
- Round 2. *Challenge* for the preprocessing phase.
  The verifier chooses a random subset $\mathcal{C} \in [M]$ with $|\mathcal{C}| = \tau$, which is used to challenge the prover to open the commitments of instances in $[M]\backslash\mathcal{C}$, so that the verifier can check the randomness and correctness of λ_j for each instance $j \in [M]\backslash\mathcal{C}$. The verifier sends $\mathcal{C}$ to the prover.
- Round 3-a. *Respond* to the challenge for the preprocessing phase.
  The prover computes the openings of the commitments of instances in $[M]\backslash\mathcal{C}$. Denote these openings as resp^F_{pre}. The prover sends resp^F_{pre} to the verifier.

Online phase $\pi^F_{on}(w, \{\mathsf{state}_{j,i}\}_{j\in\mathcal{C}, i\in[n]})$.

- Round 3-b. *Commit* to the views of each party.
  The prover runs the n-party MPC protocol for $C(w) = y$ using the masks λ_j and the witness w for each instance $j \in \mathcal{C}$, and computes the commitments to the views of each party in the MPC protocol execution. Let com^F_{on} denote these commitments. (For simplicity, the masked values of input, e.g., $\hat{w}_j = w \oplus \lambda_{j,w}$, is considered to be part of com^F_{on}.) The prover sends com^F_{on} to the verifier.
- Round 4. *Challenge* for the online phase.
  The verifier chooses a random set $\mathcal{P} = \{p_j\}_{j\in\mathcal{C}}$ with $p_j \in [n]$, which is used to challenge the prover to open the views of all but the p_j-th party for each instance $j \in \mathcal{C}$, so that the verifier can check the consistency of $n-1$ parties' views for that instance. The verifier sends $\mathcal{P}$ to the prover.

– Round 5. *Respond* to the challenge for the online phase.
The prover computes the openings of all but the p_j-th party's commitments for each instance $j \in \mathcal{C}$. Let resp_{on}^F denote these openings. The prover sends resp_{on}^F to the verifier.

Verification Strategy.

1. For the opened instances in $[M]\backslash\mathcal{C}$, the verifier uses resp_{pre}^F to recover parts of the openings of com_{pre}^F, which are also used to check the randomness and correctness of the masks.
2. For each unopened instance in $\mathcal{C}$, the verifier uses resp_{on}^F and the masked values of input to simulate the MPC protocol for $C(w) = y$ and recover the openings of com_{on}^F and the remaining openings of com_{pre}^F.
3. The verifier checks the output of the simulation of the MPC protocol and the consistency of com_{pre}^F and com_{on}^F.

4.2 Intuitive Construction for Resumable HVZKPoK

Once the verifier accepts the proof, he is convinced not only that the prover has the witness in the current session, but also the correctness and randomness of the transcripts implied by the proof. We notice that the verifier's trust on some transcripts of KKW protocol can be "reused" to reduce the cost of proofs when resuming sessions. An intuitive construction of resumable HVZKPoK for F is described as follows, where the decomposition $F = f \circ g$ is public. For simplicity, we only consider the case of two sessions.

– **HVZKPoK for the initial session.** It is similar to the original KKW protocol, except that the prover needs to prepare preprocessing values for the next session and proves the consistency of w' and w. Let $\pi^f = (\pi^f_{pre}, \pi^f_{on})$ denote the KKW proof for $C'(w') = y$, i.e., $f(w') = y$. HVZKPoK for the initial session consists of the following phases.
  1. π^F: The original KKW proof for w such that $C(w) = y$.
  2. π^f_{pre}: Prepare the preprocessing values of $C'(w') = y$ for the next session. In particular, π^F and π^f_{pre} can be merged with the same preprocessing challenge $\mathcal{C}$, which will be explained later.
  3. π_{cert}: Consistency proof for w and w'. That is, we need to guarantee that w' used in the next session is the correct intermediate value of $C(\cdot)$ when evaluating on input w.
– **HVZKPoK for the second session.**
  1. π^f_{on}: Online phase of the KKW proof for $C'(w') = y$.

Note that π^f_{pre} and π^f_{on} constitutes the complete KKW protocol for $C'(w') = y$. Intuitively, if the verifier can be convinced that the preprocessing data in π^f_{pre} is generated correctly, the prover only needs to run π^f_{on} for the second session. Suppose the verifier has accepted the initial session. Combining with the consistency proof π_{cert} for w and w', the verifier can be convinced that the prover has the knowledge of w in the second session. Therefore, the prover needs to provide efficient consistency proof for w and w', while ensuring that the preprocessing data in π^f_{pre} are generated by the "honest" prover, who has the knowledge of w. (Recall the soundness game mentioned in Sect. 3, where we do not consider the malicious prover who has the knowledge of witness.) To do so, we modify the KKW protocol π^f_{pre} for $C'(w') = y$.

4.3 Modified KKW for f and Consistency Proof

Suppose $\lambda_{w'}$ is the random masks for w' in π^F, i.e., the masked intermediate value $\hat{w}' = w' \oplus \lambda_{w'}$. The main modification of π^f is that the generation of the preprocessing data in π^f_{pre} is based on $\lambda_{w'}$. More specifically, the prover rerandomizes $\hat{w}'$ using a random and public value Δ, i.e., $\bar{w}' = w' \oplus \lambda_{w'} \oplus \Delta$. Then, the prover generates the corresponding preprocessing values for the KKW proof for $C'(w') = y$ using $\lambda'_{w'} = \lambda_{w'} \oplus \Delta$ as the mask. Here, the prover generates $n-1$ secret shares of $\lambda'_{w'}$ by running PRG with $n-1$ random seeds, while the n-th secret share $[\lambda'_{w'}]_n$ is determined by $[\lambda'_{w'}]_n = \lambda'_{w'} \oplus [\lambda'_{w'}]_1 \oplus \cdots \oplus [\lambda'_{w'}]_{n-1}$ and is sent to the verifier. (com^f_{pre} commits to the corresponding seeds for each party.) Hence, the verifier only needs to challenge the prover to open $n-2$ parties' views in the online phase.

Based on the above modification, we can provide a simple and efficient construction for the consistency proof π_{cert}. After π^f_{pre}, the prover computes a commitment com_{on}, which commits to $\mathsf{com}^F_{on}||\mathsf{com}^f_{pre}||\Delta||[\lambda'_{w'}]_n$, and sends com_{on} as well as $\mathsf{com}^f_{pre}||\Delta||[\lambda'_{w'}]_n$ to the verifier. Since the verifier accepted the initial session, the consistency of com^F_{on} and the preprocessing data of the initial session, e.g., $\lambda_{w'}$, has been checked. Due to the binding property of com_{on}, the rerandomized mask $\lambda'_{w'} = \lambda_{w'} \oplus \Delta$, which is determined by $\mathsf{com}^f_{pre}||\Delta$, is hard to be modified. In the second session, by checking the openings of com^f_{pre}, it is implicitly guaranteed that $\lambda'_{w'}$ is generated by the same "honest" prover of the initial session. Therefore, the witness w' implied by $\bar{w}'$ is the same as that of the initial session, i.e., $\bar{w}' = w' \oplus \lambda'_{w'}$, and the verifier does not need to check the correctness and randomness of the preprocessing data for the second session by "cut-and-choose". Notice that only the unopened instances in the initial session need the simulated executions of the MPC protocol, which means only in these instances, we need to rerandomize the masked intermediate value $\hat{w}'$. That is why π^F and π^f_{pre} can be merged with the same preprocessing challenge $\mathcal{C}$. To summarize, the modified KKW $\pi^f = (\pi^f_{pre}, \pi^f_{on})$ for the partial circuits C' is described as follows.

Preprocessing phase $\pi^f_{pre}(\{\lambda_{j,w'}\}_{j\in\mathcal{C}}, \mathcal{C})$.

- Round 1-a. *Rerandomize* the masks for the instances in $\mathcal{C}$.
  1. The prover chooses a random seed^{Δ} to generate Δ_j for each instance $j \in \mathcal{C}$. Then, the prover computes the rerandomized masks $\lambda'_{j,w'} = \lambda_{j,w'} \oplus \Delta_j$, where $\lambda_{j,w'}$ is the mask of $\hat{w}'_j$.
  2. For each instance $j \in \mathcal{C}$, the prover chooses a random $\mathsf{seed}_{j,i}$ for each party P_i to generate the share $[\lambda'_{j,w'}]_i$ for $i \in [n-1]$, while $[\lambda'_{j,w'}]_n$ is computed by $[\lambda'_{j,w'}]_n = \lambda'_{j,w'} \oplus [\lambda'_{j,w'}]_1 \oplus \cdots \oplus [\lambda'_{j,w'}]_{n-1}$. Other preprocessing values

are generated as described in Sect. 2.1. $[\lambda'_{j,w'}]_n$ is included as part of the auxiliary information aux_n. So we have $\mathsf{aux}_n \in \{0,1\}^{|C'|+|C'_{in}|}$. The prover sets $\mathsf{state}'_{j,i} = \mathsf{seed}_{j,i}$ for $i \in [n-1]$, and $\mathsf{state}'_{j,n} = \mathsf{seed}_{j,n}||\mathsf{aux}_n$. Compute com^f_{pre}, which commits to the n parties' states.

3. The prover sends $\mathsf{state}'_{j,n}$, seed^{Δ} and com^f_{pre} to the verifier.

Online phase $\pi^f_{on}(w', \{\mathsf{state}'_{j,i}\}_{j\in\mathcal{C}, i\in[n]})$.

- Round 1-b. *Commit* to the views of each party.
  The prover runs the MPC protocol for $C'(w') = y$ using the rerandomized masks $\lambda'_{j,w'}$ (determined by $\{\mathsf{state}'_{j,i}\}_{i\in[n]}$) and the witness w' for each instance $j \in \mathcal{C}$. The prover computes the commitment to the views of each party during the MPC protocol. Denote these commitments as com^f_{on}. Send com^f_{on} to the verifier.
- Round 2. *Challenge* for the online phase.
  The verifier chooses a random set $\mathcal{P} = \{p_j\}_{j\in\mathcal{C}}$ with $p_j \in [n-1]$ in order to challenge the prover to open the views of all but the p_j-th party for each instance $j \in \mathcal{C}$, so that the verifier can check the consistency of $n-2$ views for that instance. The verifier sends $\mathcal{P}$ to the prover.
- Round 3. *Respond* to the challenge for the online phase.
  The prover computes the openings of the commitments of those challenged parties for each instance $j \in \mathcal{C}$. Denote the response by resp^f_{on}. Send resp^f_{on} to the verifier.

Verification Strategy. The verification strategy is similar to that of the original KKW, except that there is no need to check the randomness and correctness of the masks by cut-and-choose.

1. For each instance $j \in \mathcal{C}$, the verifier uses resp^f_{on}, $\hat{w}'_j$, seed^{Δ} and $\mathsf{state}'_{j,n}$ to simulate the MPC protocol for $C'(w') = y$ and recover the openings of com^f_{on} and com^f_{pre}. (Here, the verifier can get $\hat{w}'_j$ after the initial session.)
2. The verifier checks the output of the simulation of the MPC protocol for C' and the consistency of com^f_{on} and com^f_{pre}.

5 Resumable HVZKPoK from KKW

Using the KKW protocol π^F for $C(w) = y$ and the modified version π^f for $C'(w') = y$ as building blocks, we can construct the resumable HVZKPoK protocol Π_{Res}, which consists of two sub protocols $\Pi_{Res,1}$ and $\Pi_{Res,2}$. $\Pi_{Res,1}$ is for the initial session and $\Pi_{Res,2}$ is for the resumed session. Figure 1 shows the relations of the sub protocols of our general construction.

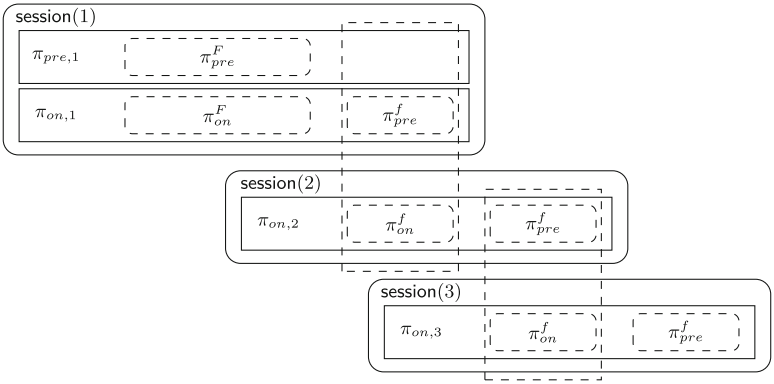

Fig. 1. General construction for resumable HVZKPoK

5.1 Resumable HVZKPoK for Initial Session $\mathit{\Pi}_{Res,1}$

The resumable HVZKPoK $\mathit{\Pi}_{Res,1} = (\pi_{pre,1}, \pi_{on,1})$ for the initial session is described as follows.

Preprocessing phase $\pi_{pre,1} = \pi^F_{pre}(1^\kappa)$. The preprocessing phase is the same as that of $\pi^F_{pre}(1^\kappa)$.

- Round 1. *Commit* to the masks of M instances.
  The prover runs round 1 of $\pi^F_{pre}(1^\kappa)$, which computes the commitments $\mathsf{com}^F_{pre,1}$ to $\{\mathsf{state}_{j,i,1}\}_{j\in[M],i\in[n]}$ for M instances, and sends $\mathsf{com}^F_{pre,1}$ to the verifier.
- Round 2. *Challenge* for the preprocessing phase.
  The verifier runs round 2 of $\pi^F_{pre}(1^\kappa)$ to send a random τ-sized set $\mathcal{C}$ to the prover.
- Round 3-a. *Respond* to the challenge for the preprocessing phase.
  The prover runs round 3 of $\pi^F_{pre}(1^\kappa)$ to generate the corresponding response, denoted by $\mathsf{resp}^F_{pre,1}$, and sends it to the verifier.

Online phase $\pi_{on,1} = (\pi^F_{on}(w, \{\mathsf{state}_{j,i,1}\}_{j\in\mathcal{C},i\in[n]}) \wedge \pi^f_{pre}(\{\lambda_{j,w'}\}_{j\in\mathcal{C}}, \mathcal{C}) \wedge \pi_{cert})$. The prover's strategy of $\pi_{on,1}$ includes: (1) Run $\pi^F_{on}(w, \{\mathsf{state}_{j,i,1}\}_{j\in\mathcal{C},i\in[n]})$ as the original KKW protocol; (2) Prepare the masks for the next session; (3) Certify these masks for the next session to ensure that the witness w' which will be used in the next session is consistent with w.

- Round 3-b.
  1. Run round 3-b of $\pi^F_{on}(\omega, \{\mathsf{state}_{j,i,1}\}_{j\in\mathcal{C},i\in[n]})$ to generate the corresponding commitment $\mathsf{com}^F_{on,1}$. Denote the corresponding intermediate mask as $\lambda_{j,w'}$ for each instance $j \in \mathcal{C}$.
  2. Run round 1-a of $\pi^f_{pre}(\{\lambda_{j,w'}\}_{j\in\mathcal{C}}, \mathcal{C})$ to generate a random seed^Δ_2, the state of each party $\{\mathsf{state}'_{j,i,2}\}_{j\in\mathcal{C},i\in[n]}$ and $\mathsf{com}^f_{pre,2}$ as described above for the next session.

3. Run π_{cert} to commit to $\mathsf{com}^F_{on,1}||\mathsf{com}^f_{pre,2}||\mathsf{seed}^\Delta_2||\{\mathsf{state}'_{j,n,2}\}_{j\in\mathcal{C}}$. Denote the corresponding commitment as $\mathsf{com}_{on,1}$. Send $\mathsf{com}_{on,1}$, $\mathsf{com}^f_{pre,2}$, seed^Δ_2 and $\{\mathsf{state}'_{j,n,2}\}_{j\in\mathcal{C}}$ to the verifier.

– Round 4. *Challenge* for the online phase.
The verifier runs round 4 of $\pi^F_{on}(w, \{\mathsf{state}_{j,i,1}\}_{j\in\mathcal{C},i\in[n]})$ to send a random set $\mathcal{P} = \{p_j\}_{j\in\mathcal{C}}$ with $p_j \in [n]$ to the prover.
– Round 5. *Respond* to the challenge for the online phase.
The prover runs round 5 of $\pi^F_{on}(w, \{\mathsf{state}_{j,i,1}\}_{j\in\mathcal{C},i\in[n]})$ to generate the corresponding response $\mathsf{resp}^F_{on,1}$, and sends it to the verifier.

Verification Strategy. The verification strategy is similar to that of π^F, except that the verifier needs to check the consistency of $\mathsf{com}_{on,1}$.

1. For the opened instances in $[M]\backslash\mathcal{C}$, the verifier uses $\mathsf{resp}^F_{pre,1}$ to recover parts of the openings of $\mathsf{com}^F_{pre,1}$, which are also used to check the randomness and correctness of masks.
2. For each unopened instance in $\mathcal{C}$, the verifier uses $\mathsf{resp}^F_{on,1}$ to simulate the MPC protocol for $C(w) = y$ and recover the openings of $\mathsf{com}^F_{on,1}$ and the remaining openings of $\mathsf{com}^F_{pre,1}$.
3. The verifier checks the output of the simulation of the MPC protocol and the consistency of $\mathsf{com}^F_{pre,1}$.
4. The verifier checks the consistency of $\mathsf{com}^F_{on,1}$ and $\mathsf{com}_{on,1}$.

State Update. The prover and the verifier need to maintain states for session resumption. The prover's initial state is $\mathsf{pstate} = w$, and the verifier's is $\mathsf{vstate} = \perp$. After the initial session, the prover and the verifier update their states as follows.

– Prover's state update: $\mathsf{pstate} = \{\hat{w}'_j\}_{j\in\mathcal{C}}||\mathsf{seed}^\Delta_2||\{\mathsf{state}'_{j,i,2}\}_{j\in\mathcal{C},i\in[n]}$, where $\hat{w}'_j$ is the masked intermediate value for each instance $j \in \mathcal{C}$.
– Verifier's state update: $\mathsf{vstate} = \{\hat{w}'_j\}_{j\in\mathcal{C}}||\mathsf{seed}^\Delta_2||\{\mathsf{state}'_{j,n,2}\}_{j\in\mathcal{C}}||\mathsf{com}^f_{pre,2}$.

We emphasize that vstate can be made public.

5.2 Resumable HVZKPoK for Second Session $\Pi_{Res,2}$

For simplicity, we present the resumable HVZKPoK $\Pi_{Res,2}$ for the second session, which can be easily extended to the case of $\mathsf{session}(t)$ for any $t > 1$.
Online phase $\pi_{on,2} = (\pi^f_{on}(w', \{\mathsf{state}'_{j,i,2}\}_{j\in\mathcal{C},i\in[n]}) \wedge \pi^f_{pre}(\{\lambda_{j,w'}\}_{j\in\mathcal{C}}, \mathcal{C}) \wedge \pi_{cert})$.

The prover's strategy $\pi_{on,2}$ is similar to $\pi_{on,1}$, except that he simulates the MPC protocol for $C'(w') = y$ instead of $C(w) = y$. Note that all the inputs of $\pi_{on,2}$ can be extracted from pstate.

– Round 1.
1. Run round 3-b of $\pi^f_{on}(w', \{\mathsf{state}'_{j,i,2}\}_{j\in\mathcal{C},i\in[n]})$ to generate the corresponding commitment $\mathsf{com}^f_{on,2}$.

2. Run round 1-a of $\pi^f_{pre}(\{\lambda_{j,w'}\}_{j\in\mathcal{C}},\mathcal{C})$ to generate a random seed^{Δ}_3, the state of each party $\{\mathsf{state}'_{j,i,3}\}_{j\in\mathcal{C},i\in[n]}$ and $\mathsf{com}^f_{pre,3}$ as described above for the next session.
3. Run π_{cert} to generate $\mathsf{com}_{on,2}$, which is the commitment of $\mathsf{com}^f_{on,2}||\mathsf{com}^f_{pre,3}\ ||\mathsf{seed}^{\Delta}_3\ ||\{\mathsf{state}'_{j,n,3}\}_{j\in\mathcal{C}}$. Send $\mathsf{com}_{on,2}$, $\mathsf{com}^f_{pre,3}$, seed^{Δ}_3 and $\{\mathsf{state}'_{j,n,3}\}_{j\in\mathcal{C}}$ to the verifier.

– Round 2. *Challenge* for the online phase.
The verifier runs round 2 of $\pi^f_{on}(w',\{\mathsf{state}'_{j,i,2}\}_{j\in\mathcal{C},i\in[n]})$ to send a random set $\mathcal{P}=\{p_j\}_{j\in\mathcal{C}}$ with $p_j\in[n-1]$ to the prover.
– Round 3. *Respond* to the challenge for the online phase.
The prover runs round 3 of $\pi^f_{on}(w',\{\mathsf{state}'_{j,i,2}\}_{j\in\mathcal{C},i\in[n]})$ to generate the corresponding response $\mathsf{resp}^f_{on,2}$. Send $\mathsf{resp}^f_{on,2}$ to the verifier.

Verification Strategy.

1. For each unopened instance $j\in\mathcal{C}$, the verifier uses $\mathsf{resp}^f_{on,2}$, $\hat{w}'_j$, seed^{Δ}_2 and $\mathsf{state}'_{j,n,2}$ to simulate the MPC protocol for $C'(w')=y$ and recover the openings of $\mathsf{com}^f_{on,2}$ and $\mathsf{com}^f_{pre,2}$. Note that $\hat{w}'_j$, seed^{Δ}_2 and $\mathsf{state}'_{j,n,2}$ can be extracted from vstate.
2. The verifier checks the output of the simulation of the MPC protocol.
3. The verifier checks the consistency of $\mathsf{com}^f_{pre,2}$, $\mathsf{com}^f_{on,2}$ and $\mathsf{com}_{on,2}$.

State Update. The prover and the verifier update their states as follows.

– Prover's state update: $\mathsf{pstate}=\{\hat{w}'_j\}_{j\in\mathcal{C}}||\mathsf{seed}^{\Delta}_3||\{\mathsf{state}'_{j,i,3}\}_{j\in\mathcal{C},i\in[n]}$.
– Verifier's state update: $\mathsf{vstate}=\{\hat{w}'_j\}_{j\in\mathcal{C}}||\mathsf{seed}^{\Delta}_3||\{\mathsf{state}'_{j,n,3}\}_{j\in\mathcal{C}}||\mathsf{com}^f_{pre,3}$.

5.3 Security

Theorem 1. *Assume that π^F, the underlying commitment scheme and pseudorandom generator are secure, and F has an extractable decomposition as $f\circ g$. Then Π_{Res} is a resumable honest verifier zero-knowledge proof of knowledge.*

The proof of zero-knowledge in Theorem 1 is similar to that of [38], while we should consider the zero-knowledge property of all the sessions as a whole. For the proof of resumable knowledge soundness, we show the consistency between w' of $\mathsf{session}(t)$ and w, and use the method of [5,38] to construct a witness extractor $\mathcal{E}$ for each session. Formal proof of Theorem 1 is in Appendix A.

Parallel Repetition. The soundness error ξ_t of $\Pi_{Res,2}$ for $\mathsf{session}(t)$ may be higher than ξ_1. We could reduce ξ_t with parallel executions of $\Pi_{Res,2}$ as follows.

– In round 3 of the online phase $\pi_{on,1}$ (or round 1 of $\pi_{on,2}$) for $\mathsf{session}(t-1)$, the prover repeats round 1 of π^f_{pre} for ℓ times. In other words, the prover rerandomizes intermediate masked values for ℓ times with ℓ random $\{\Delta_i\}_{i\in[\ell]}$.

– For $\mathsf{session}(t)$, the prover and the verifier run $\Pi_{Res,2}$ for ℓ times with the corresponding masked values generated in the previous session, where the verifier needs to send ℓ random challenges in round 2 of $\pi_{on,2}$.

Indeed, the above method can be interpreted as compacting ℓ executions of $\Pi_{Res,2}$ for ℓ sessions into one session, which will not break the security of our resumable HVZKPoK due to the honest verifier setting. In this way, we can reduce ξ_t to $\frac{1}{(n-1)^{\tau\cdot\ell}}$. By choosing appropriate ℓ, M, n and τ, we can gain a better soundness error for $\mathsf{session}(t)$. (Note that there is a trade-off between the soundness error and the proof size.) The proof of the following theorem are similar to that of Theorem 1 and hence omitted.

Theorem 2. *Assume that π^F, the underlying commitment scheme and pseudorandom generator are secure, and F has an extractable decomposition as $f \circ g$. Then Π_{Res} with parallel executions is a resumable honest verifier zero-knowledge proof of knowledge.*

5.4 3-Round Resumable HVZKPoK

Our 5-round $\Pi_{Res,1}$ can be transformed into a 3-round protocol using the similar method of [38] with the modification that the prover needs to prepare the random masks of the next session for every instance in $[M]$. Such modification has no effect on the security of the initial session, as those random masks could be considered as redundant information if there are no subsequent sessions. A concrete construction of our 3-round resumable HVZKPoK Π^3_{Res}, in which F is instantiated with LowMC [2] as in [38], is shown in our full paper [44]. The security proof of the following theorem is similar to that of Theorem 1 and hence omitted.

Theorem 3. *Assume that the underlying hash function, commitment scheme and pseudorandom generator are secure. Then Π^3_{Res} is a resumable honest verifier zero-knowledge proof of knowledge.*

6 Resumable-Picnic

As in the previous works [18,38], our 3-round protocol can be transformed into a resumable *non-interactive* ZKPoK (NIZKPoK) using the Fiat-Shamir heuristic in each session, and the resulting NIZKPoK can be used to construct a stateful signature scheme. More precisely, we instantiate F with $\mathsf{Enc}(\cdot, 0^\kappa)$ for some symmetric encryption scheme $\mathsf{Enc}(\cdot,\cdot)$ in which the first input is the key and the second input is the plaintext. The signing key is a uniform $\mathsf{sk} \in \{0,1\}^\kappa$ and the verification key is $\mathsf{pk} = \mathsf{Enc}(\mathsf{sk}, 0^\kappa)$. By applying Fiat-Shamir heuristic to each session of our 3-round protocol for the relation $(\mathsf{pk}, \mathsf{sk}) \in R$, we can obtain a sequence of signatures, where the t-th signature is denoted as σ_t. We denote our stateful signature scheme as Resumable-Picnic.

Theorem 4. *Resumable-Picnic is strongly unforgeable under chosen message attacks in the QROM when* Com *is a collapse-binding commitment scheme and* H *is a collapsing hash function.*

Due to page restrictions, more details of Resumable-Picnic and the proof of Theorem 4 are presented in the full version [44].

Application. One of the possible applications of Resumable-Picnic is in the blockchain setting, where each σ_i can be stored in the corresponding block publicly. The verifier only needs to check the validity of σ_t of the current block, since the validity of $\{\sigma_i\}_{i<t}$ in previous blocks are implied by the consistency of the underlying consensus protocol. More specifically, the signer can sign a transaction tx_1 using Resumable-Picnic with pk and generate the signature σ_1. Then the miner generates a block B_i for a set of transactions with corresponding signatures, which includes $(\mathsf{tx}_1, \sigma_1)$. Afterwards, when the signer wants to sign another transaction tx_2 with pk, he can generate σ_2 efficiently using the state ss_1. Due to the blockchain protocol, $(\mathsf{tx}_2, \sigma_2)$ will be included in some block, say B_j, for $j > i$. If block B_i has been confirmed, which implies the validity of the signatures included in B_i have been confirmed by the majority of miners, the verifier of σ_2 does not need to check the validity of σ_1 any more. Due to the efficiency of session resumption, σ_2 is more efficient than the original Picnic. (Note that there is usually a confirmation delay in most blockchain protocols. For instance, the confirmation delay of Bitcoin is about 6 blocks, which means a block is confirmed if it is followed by at least 6 blocks.)

Table 1. Comparison between Picnic3 and resumable-Picnic. "Size" denotes the signature size. The results are the median time for running 10000 times.

Scheme	M	n	τ	ℓ	Sign (ms)	Verify (ms)	Size (Bytes)
Picnic3-Level 1	252	16	36		71.68	51.37	12595 ± 223
Resumable-Picnic [session(1)]	252	16	36	1	99.99	76.23	14277 ± 243
Resumable-Picnic [session(2)]	252	16	36	1	8.31	4.78	4796
Resumable-Picnic [session($t > 2$)]	252	16	36	1	6.59	4.11	4796
Picnic3-Level 3	419	16	52		170.37	119.45	27104 ± 455
Resumable-Picnic [session(1)]	419	16	52	1	220.45	163.91	31166 ± 466
Resumable-Picnic [session(2)]	419	16	52	1	13.15	7.90	10088
Resumable-Picnic [session($t > 2$)]	419	16	52	1	10.60	6.67	10088
Picnic3-Level 5	601	16	68		487.45	290.22	48716 ± 721
Resumable-Picnic [session(1)]	601	16	68	1	512.56	332.24	55043 ± 673
Resumable-Picnic [session(2)]	601	16	68	1	18.28	11.26	17536
Resumable-Picnic [session($t > 2$)]	601	16	68	1	14.97	9.52	17536

Experimental Results and Comparison. We implement Resumable-Picnic using the same parameters as Picnic3 [36], and give an efficiency comparison with Picnic3. Our benchmarks run on a platform with an Intel Core i7-8700 CPU clocked at 3.2 GHz and 16GB RAM. The parameters are chosen as Picnic3 did which fits security level 1, 3, and 5 recommended by NIST. The comparison between Picnic3 and Resumable-Picnic are shown in Table 1. As shown in Table 1, although the cost of Resumable-Picnic's initial session is slightly higher than that of Picnic3, the efficiency of Resumable-Picnic for the subsequent sessions are improved dramatically. Compared with Picnic3 with security level 1, 3 and 5, the sign/verify time of Resumable-Picnic for session$(t > 2)$ is reduced to 9.2%/8.0%, 6.2%/5.6%, and 3.1%/3.3%, respectively, and the signature size is reduced to 38.1%, 37.2% and 36.0%, respectively.

7 Compressed 1-out-of-N Proof and Ring Signatures

[20] provides a novel method of the one-out-of-N proof for the relation R_{OR} defined by $(x_1, \ldots, x_N \in L_R; w) \in R_{\text{OR}} \iff \exists t \in [N], s.t. (x_t, w) \in R$. By applying the parallel version of $\Pi_{Res,2}$ described in Sect. 5.3 to the CDS method [20], we can get a compressed one-out-of-N proof when the N statements share the same circuit. The main idea is that, for the $N-1$ statements which the prover does not know the witness, the prover runs the simulator of the resumable HVZKPoK $\Pi_{Res,2}$ for the partial circuit C' in parallel. Hence, most transcripts of the simulation for the $N-1$ statements can be removed. Furthermore, based on our compressed one-out-of-N proof, we can construct a ring signature from symmetric-key primitives. More details of our compressed one-out-of-N proof and the resulting ring signature are presented in the full version [44].

Using the same parameter set of Picnic2, we make a comparison between the ring signature of [38] and ours in Table 2. It shows that the size of our ring signature is smaller than that of [38] when the ring size is less than 2^6. In particular, it is just about 1/3 of the ring signature size of [38] when the ring size is less than 2^4.

Table 2. Comparison between ring signature [38] and our work.

Ring size	2	2^2	2^3	2^4	2^5	2^6	2^7
$\lvert\sigma\rvert$ ([38])	70KB	106KB	142KB	177KB	213KB	249KB	285KB
$\lvert\sigma\rvert$ (Ours)	21KB	30KB	47KB	82KB	151KB	290KB	567KB

Acknowledgements. We would like to thank the anonymous reviewers for their insightful and helpful comments. Handong Zhang, Puwen Wei, Jinsong Li, Wei Wang and Guoxiao Liu were supported by the National Key Research and Development Program of China (Grant No. 2018YFA0704702), Shandong Provincial Key Research and

Development Program (Major Scientific and Technological Innovation Project) (Grant No.2019JZZY010133) and Shandong Provincial Natural Science Foundation (Grant No. ZR2020MF053). Haiyang Xue was supported by the National Natural Science Foundation of China (Grant No. 62172412). Yi Deng was supported by the National Natural Science Foundation of China (Grant No. 61932019 and No. 61772522), the Key Research Program of Frontier Sciences, CAS (Grant No. QYZDB-SSW-SYS035) and Natural Science Foundation of Beijing (Grant No. M22003).

A Proof of Theorem 1

Proof. **Completeness.** This property follows from the correctness of the underlying MPC protocol Π used in π^F and π^f.

Resumable Honest Verifier Zero-Knowledge. We need to consider the simulator for all $q(\kappa)$ sessions instead of only one, where the simulation for the transcripts generated by π^F and π^f follows the idea of [38]. Let Sim_Π denotes the simulator of the MPC protocol Π. The simulator Sim of Π_{Res} is described as follows.

- Simulation for initial session $\mathsf{session}(1)$.
  1. Sim chooses random $\mathcal{C}$ and $\mathcal{P}$ as the challenge for the preprocessing phase and the online phase respectively.
  2. For each instance $j \notin \mathcal{C}$, Sim prepares λ_j using $\{\mathsf{state}_{j,i,1}\}_{i\in[n]}$ and generates the corresponding $\mathsf{resp}^F_{pre,1}$ as an honest prover would do in the preprocessing phase.
  3. For each instance $j \in \mathcal{C}$, Sim chooses a random masked input for the MPC protocol and $n-1$ random states for $n-1$ parties determined by $\mathcal{P}$. Then, Sim runs Sim_Π to simulate the views of the n parties during the MPC protocol and computes corresponding $\mathsf{com}^F_{on,1}$. Notice that Sim can get the corresponding intermediate masked value $\hat{w}'_j$ for each instance $j \in \mathcal{C}$ from the simulated views. As mentioned in Sect. 2.1, the indistinguishability between the simulated execution of Sim_Π and the real execution relies on the security of the underlying PRG.
  4. Sim computes $\mathsf{com}^F_{pre,1}$ and $\mathsf{resp}^F_{on,1}$ according to the transcripts generated in step 2 and 3. For the generation of $\mathsf{com}^F_{pre,1}$, the state of the party in $\mathcal{P}$ of each instance can be set by 0-string with appropriate length.
  5. Sim randomly chooses seed^Δ_2 and $\{\mathsf{state}'_{j,i,2}\}_{j\in\mathcal{C},i\in[n]}$, and computes the corresponding commitment $\mathsf{com}^f_{pre,2}$. Generate $\mathsf{com}_{on,1}$ as the commitment to $\mathsf{com}^F_{on,1}||\mathsf{com}^f_{pre,2}||\mathsf{seed}^\Delta_2||\{\mathsf{state}'_{j,n,2}\}_{j\in\mathcal{C}}$.
- Simulation for subsequent session $\mathsf{seesion}(t)$, where $1 < t \leq q(\kappa)$.
  1. Sim chooses a random $\mathcal{P}$ as the challenge for the online phase.
  2. For each instance $j \in \mathcal{C}$, Sim computes the rerandomized intermediate masked input $\hat{w}'_j \oplus \Delta_j$, in which $\hat{w}'_j$ is the intermediate masked value of $\mathsf{seesion}(1)$ and Δ_j is generated by seed^Δ_t. Note that Sim has $n-2$ parties' states determined by $\mathcal{P}$. Then, Sim runs Sim_Π to simulate the views of n parties during the MPC protocol, and computes corresponding $\mathsf{com}^f_{on,t}$.

3. Sim randomly chooses $\mathsf{seed}^{\Delta}_{t+1}$ and $\{\mathsf{state}'_{j,i,t+1}\}_{j\in\mathcal{C},i\in[n]}$, and computes the corresponding commitment $\mathsf{com}^{f}_{pre,t+1}$. Generate $\mathsf{com}_{on,t}$ as the commitment to $\mathsf{com}^{f}_{on,t}||\mathsf{com}^{f}_{pre,t+1}||\mathsf{seed}^{\Delta}_{t+1}||\{\mathsf{state}'_{j,n,t+1}\}_{j\in\mathcal{C}}$.

Following a standard hybrid argument, we have that the transcript generated by Sim is computationally indistinguishable from that of a real protocol, where the indistinguishability relies on the indistinguishability of the simulated transcripts generated by Sim_{Π} and the hiding property of the commitment scheme.

Resumable Knowledge Soundness. The proof of the resumable knowledge soundness is similar to that of [5,38], except that we need to show that there exists a witness extractor $\mathcal{E}$ for each session, especially the resumed session.

We first show the soundness error $\xi(M, n, \tau)$. Since $\Pi_{Res,1}$ is similar to that of the original KKW except additional processing for the masks of the next session. The soundness error ξ_1 of $\Pi_{Res,1}$ is the same as that of [38]. That is,

$$\xi_1(M, n, \tau) = \max_{0\leq c\leq\tau}\left\{\frac{\binom{M-c}{M-\tau}}{\binom{M}{M-\tau}\cdot n^{\tau-c}}\right\},$$

where c denotes the number of preprocessing emulations where the malicious prover cheats.

On the soundness error of $\Pi_{Res,2}$, recall the soundness game mentioned in Sect. 3, where the malicious prover can invoke the "honest" prover to interact with the verifier for polynomially-many sessions, say $\mathsf{session}(1), \ldots, \mathsf{session}(t-1)$ for $1 < t \leq q(\kappa)$, and tries to convince the verifier in $\mathsf{session}(t)$ without the help of the "honest" prover. Note that the masks for $\mathsf{session}(t)$ are generated by the honest prover in $\mathsf{session}(t-1)$. So a malicious prover of session t can cheat only in the online phase, where he must cheat in one of the views of the $n-1$ parties. Thus, the probability that the prover will not be detected in $\Pi_{Res,2}$ is $\xi_t(M, n, \tau) = \frac{1}{(n-1)^{\tau}}$. Therefore, we have $\xi(M, n, \tau) = \max\{\xi_1(M, n, \tau), \xi_t(M, n, \tau)\}$, for any $1 < t \leq q(\kappa)$. Next, we proceed to prove the resumable knowledge soundness property by showing how to construct $\mathcal{E}$ to extract a valid witness for each session. As explained above, the proof of knowledge soundness in [5] can be applied to $\Pi_{Res,1}$ directly. We focus on $\Pi_{Res,2}$ of $\mathsf{session}(t)$, where $1 < t \leq q(\kappa)$. For simplicity we assume that the commitment scheme is perfectly binding.

We first prove that if the success probability of cheating $\delta_t(x) > \xi_t(M, n, \tau)$, then there exists at least one MPC instance of $\mathcal{C}$, where the prover has committed to a valid intermediate value w'. Considering the deterministic prover with fixed random tape, let $\mathbf{v}$ be a 0/1-vector with length $(n-1)^{\tau}$, where each entry corresponds to a possible challenge for the online phase of $\mathcal{V}^{(t)}$ and 1 denotes the event of success. Hence, we have that $\delta_t(x)$ is the fraction of '1' entries in $\mathbf{v}$ and the number of '1' entries in $\mathbf{v}$ is higher than 1 due to $\delta_t(x) > \xi_t(M, n, \tau) = \frac{1}{(n-1)^{\tau}}$. That is, there must exist two accepting transcripts with different challenges $\{p_j\}_{j\in\mathcal{C}}$ and $\{p'_j\}_{j\in\mathcal{C}}$ such that $p_j \neq p'_j$ for an MPC instance j. That means all the views of the parties in instance j are correct and the witness used in this instance must be a valid intermediate value w'.

However, since f is just a part of F, it may be easy for a malicious prover to find a different $w^* \neq w'$ such that $f(w^*) = 1$. It seems that any malicious prover who can find such a w^* can cheat in the next session by computing $\lambda_{w^*} = w' \oplus \lambda_{w'} \oplus w^*$ and generating the corresponding n shares of $\lambda_{w^*} \oplus \Delta$. ($w' \oplus \lambda_{w'}$ can be extracted during the verification of the initial session.) Thanks to the binding property of the commitment $\mathsf{com}_{on,t}$ in π_{cert}, it is hard for the adversary to provide consistency proof using such w^* and λ_{w^*}. For instance, in $\mathsf{session}(t-1)$, $\mathsf{com}_{on,t-1}$ is the commitment of $\mathsf{com}^f_{on,t-1}||\mathsf{com}^f_{pre,t}||\mathsf{seed}^\Delta_t$ $||\{\mathsf{state}'_{j,n,t}\}_{j\in\mathcal{C}}$, where ($\mathsf{com}_{on,t-1}$, $\mathsf{com}^f_{pre,t}$, seed^Δ_t, $\{\mathsf{state}'_{j,n,t}\}_{j\in\mathcal{C}}$) are public. The rerandomized mask for $\mathsf{session}(t)$, say $\lambda_{w'} \oplus \Delta$, is determined by ($\mathsf{com}^f_{pre,t}$, seed^Δ_t, $\{\mathsf{state}'_{j,n,t}\}_{j\in\mathcal{C}}$) and is hard to be modified due to $\mathsf{com}_{on,t-1}$. (The use of mask λ_{w^*} such that $\lambda_{w^*} \neq \lambda_{w'} \oplus \Delta$ will be detected by checking the consistency of $\mathsf{com}_{on,t-1}$ and $\mathsf{com}^f_{pre,t}$.) Therefore, a malicious prover needs to (1) guess the challenge sent by the verifier successfully, which happens with probability $\frac{1}{n-1}$ for each instance, or (2) find $n-1$ random seeds which can be used to generate an $(n-1)$-out-of-$(n-1)$ secret-sharing of $\lambda_{w^*} \oplus \Delta \oplus [\lambda_{w'} \oplus \Delta]_n$, where each share is generated by running PRG with the corresponding random seed. This can be done with negligible probability assuming the underlying PRG is secure. Hence, $\mathsf{com}_{on,t-1}$ and $\mathsf{com}^f_{pre,t}$ guarantee the consistency of w' in $\mathsf{session}(t)$ with w.

Next, we show how to extract the witness using two accepting transcripts with $\{p_j\}_{j\in\mathcal{C}}$ and $\{p'_j\}_{j\in\mathcal{C}}$ when the challenge for j is different. Since $p_j \neq p'_j$, the transcripts with p_j reveals $n-1$ shares of the masks of the intermediate masked input, whereas the transcripts with p'_j reveals the remaining shares (Notice that the shares of the n-th party is public). Hence, we can get all the shares to recover the intermediate value w'. Due to the special property of the decomposition for F, the witness w can be further extracted from w'.

To sum up, the extractor $\mathcal{E}$ is described as follows.

1. Run $\Pi_{Res,2}$ with the prover in session t until the event of success happens, in order to find an '1' entry of the vector $\mathbf{v}$, where the corresponding challenge is $\{p_j\}_{j\in\mathcal{C}}$.
2. Run $\Pi_{Res,2}$ with the prover in session t (using different challenges) until a different '1' entry is found, where the corresponding challenge is $\{p'_j\}_{j\in\mathcal{C}}$ such that $p_j \neq p'_j$.
3. Extract the witness ω in execution j using the related transcripts with $\{p_j\}_{j\in\mathcal{C}}$ and $\{p'_j\}_{j\in\mathcal{C}}$. If $F(w) = y$, output w and halt.

Let $\delta_t(x) = \xi_t(M, n, \tau) + \epsilon_t(x)$ for some $\epsilon_t(x) > 0$. The expected running time of the step 1 and 2 is $\frac{1}{\delta_t(x)} < \frac{1}{\epsilon_t(x)}$ and the running time of step 3 depends on the running time of $F(w)$ with common input x, which is supposed to be more efficient than step 1 and 2. Therefore, a valid witness can be extracted in $O(\frac{1}{\epsilon_t(x)})$ expected number of steps.

Resumption Efficiency. $\Pi_{Res,2}$ consists of π^f and the consistency proof π_{cert}. Since π^f is a simplified KKW proof for the partial circuits of F (without cut-and-choose), the complexity of π^f is much smaller than that of the original KKW proof for F. Recall that π_{cert} mainly consists of $\mathsf{com}_{on,2}$ and seed_3^{Δ}. So the complexity of π_{cert} just takes a very small portion of π^f. Hence, although the overall complexity of $\Pi_{Res,2}$ depends on the concrete decomposition of F, $\Pi_{Res,2}$ is much efficient than that of the original KKW proof Π' for F in general. □

References

1. Abe, M., Ambrona, M., Bogdanov, A., Ohkubo, M., Rosen, A.: Non-interactive composition of sigma-protocols via share-then-hash. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12493, pp. 749–773. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64840-4_25
2. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 430–454. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_17
3. Ames, S., Hazay, C., Ishai, Y., Venkitasubramaniam, M.: Ligero: lightweight sublinear arguments without a trusted setup. In: ACM CCS 2017, pp. 2087–2104. ACM Press, New York (2017). https://doi.org/10.1145/3133956.3134104
4. Avanzi, R., et al.: Crystals-kyber. NIST PQC Round **3**, 4 (2020)
5. Baum, C., Nof, A.: Concretely-efficient zero-knowledge arguments for arithmetic circuits and their application to lattice-based cryptography. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020. LNCS, vol. 12110, pp. 495–526. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45374-9_17
6. Baum, C., de Saint Guilhem, C.D., Kales, D., Orsini, E., Scholl, P., Zaverucha, G.: Banquet: short and fast signatures from AES. In: Garay, J.A. (ed.) PKC 2021. LNCS, vol. 12710, pp. 266–297. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-75245-3_11
7. Bellare, M., Goldwasser, S.: New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 194–211. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_19
8. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable zero knowledge with no trusted setup. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11694, pp. 701–732. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_23
9. Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., Virza, M.: SNARKs for C: verifying program executions succinctly and in zero knowledge. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 90–108. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_6
10. Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M., Ward, N.P.: Aurora: transparent succinct arguments for R1CS. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11476, pp. 103–128. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17653-2_4

11. Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Succinct non-interactive zero knowledge for a von neumann architecture. In: 23rd USENIX Security Symposium, pp. 781–796. USENIX Association, San Diego, CA (2014). https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/ben-sasson
12. Blum, M., De Santis, A., Micali, S., Persiano, G.: Noninteractive zero-knowledge. SIAM J. Comput. **20**(6), 1084–1118 (1991). https://doi.org/10.1137/0220068
13. Bootle, J., Cerulli, A., Chaidos, P., Groth, J., Petit, C.: Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 327–357. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_12
14. Bootle, J., Groth, J.: Efficient batch zero-knowledge arguments for low degree polynomials. In: Abdalla, M., Dahab, R. (eds.) PKC 2018. LNCS, vol. 10770, pp. 561–588. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76581-5_19
15. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy, pp. 315–334 (2018). https://doi.org/10.1109/SP.2018.00020
16. Chase, M., et al.: The picnic signature scheme, design document v2. 1 (2019)
17. Chase, M., et al.: The picnic signature scheme, design document v2. 2. Available at https://microsoft.github.io/Picnic/ (2020)
18. Chase, M., et al.: Post-quantum zero-knowledge and signatures from symmetric-key primitives. In: ACM CCS 2017, pp. 1825–1842. ACM Press, New York (2017). https://doi.org/10.1145/3133956.3133997
19. Costello, C., et al.: Geppetto: versatile verifiable computation. In: 2015 IEEE Symposium on Security and Privacy, pp. 253–270 (2015). https://doi.org/10.1109/SP.2015.23
20. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48658-5_19
21. Ding, J., Chen, M.S., Petzoldt, A., Schmidt, D., Yang, B.Y.: Rainbow. NIST PQC Round **3**, 4 (2020)
22. Feige, U., Lapidot, D., Shamir, A.: Multiple noninteractive zero knowledge proofs under general assumptions. SIAM J. Comput. **29**(1), 1–28 (1999). https://doi.org/10.1137/S0097539792230010
23. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_12
24. Fischlin, M., Harasser, P., Janson, C.: Signatures from sequential-or proofs. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12107, pp. 212–244. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45727-3_8
25. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 626–645. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_37
26. Giacomelli, I., Madsen, J., Orlandi, C.: ZKBoo: faster zero-knowledge for Boolean circuits. In: 25th USENIX Security Symposium, pp. 1069–1083. USENIX Association, Austin, TX (2016). https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/giacomelli
27. Goel, A., Green, M., Hall-Andersen, M., Kaptchuk, G.: Stacking sigmas: a framework to compose σ-protocols for disjunctions. Cryptology ePrint Archive, Report 2021/422 (2021). https://ia.cr/2021/422

28. Goel, A., Green, M., Hall-Andersen, M., Kaptchuk, G.: Efficient set membership proofs using MPC-in-the-head. In: Proceedings on Privacy Enhancing Technologies **2022**(2), 304–324 (2022). https://doi.org/10.2478/popets-2022-0047
29. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In: SFCS 1986, pp. 174–187. IEEE Computer Society Press (1986). https://doi.org/10.1109/SFCS.1986.47
30. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. **18**(1), 186–208 (1989). https://doi.org/10.1137/0218012
31. Groth, J.: Short pairing-based non-interactive zero-knowledge arguments. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_19
32. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 305–326. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_11
33. Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero knowledge for NP. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_21
34. Henry, R., Goldberg, I.: Batch proofs of partial knowledge. In: Jacobson, M., Locasto, M., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 2013. LNCS, vol. 7954, pp. 502–517. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38980-1_32
35. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: STOC 2007, pp. 21–30. ACM Press, New York (2007). https://doi.org/10.1145/1250790.1250794
36. Kales, D., Zaverucha, G.: Improving the performance of the picnic signature scheme. Cryptology ePrint Archive, Report 2020/427 (2020). https://eprint.iacr.org/2020/427
37. Katz, J.: Digital signatures. Springer Science & Business Media (2010)
38. Katz, J., Kolesnikov, V., Wang, X.: Improved non-interactive zero knowledge with applications to post-quantum signatures. In: ACM CCS 2018, pp. 525–537. ACM Press, New York (2018). https://doi.org/10.1145/3243734.3243805
39. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: nearly practical verifiable computation. In: 2013 IEEE Symposium on Security and Privacy, pp. 238–252 (2013). https://doi.org/10.1109/SP.2013.47
40. Peng, K., Bao, F.: Batch ZK proof and verification of OR logic. In: Yung, M., Liu, P., Lin, D. (eds.) Inscrypt 2008. LNCS, vol. 5487, pp. 141–156. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01440-6_13
41. Rescorla, E., Dierks, T.: The transport layer security (TLS) protocol version 1.3. RFC 8446, https://doi.org/10.17487/RFC8446, August 2018 (2018)
42. de Saint Guilhem, C.D., De Meyer, L., Orsini, E., Smart, N.P.: BBQ: using AES in picnic signatures. In: Paterson, K.G., Stebila, D. (eds.) SAC 2019. LNCS, vol. 11959, pp. 669–692. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-38471-5_27
43. Wahby, R.S., Tzialla, I., Shelat, A., Thaler, J., Walfish, M.: Doubly-efficient zksnarks without trusted setup. In: 2018 IEEE Symposium on Security and Privacy (SP), pp. 926–943 (2018). https://doi.org/10.1109/SP.2018.00060
44. Zhang, H., Wei, P., Xue, H., Deng, Y., Li, J., Wang, W., Liu, G.: Resumable zero-knowledge for circuits from symmetric key primitives. Cryptology ePrint Archive, Report 2022/556 (2022). https://eprint.iacr.org/2022/556