DAG-Σ: A DAG-based Sigma Protocol for Relations in CNF

Gongxian Zeng[1] Junzuo Lai[2] Zhengan Huang[1]
Yu Wang[1] Zhiming Zheng[1,3]

[1]Peng Cheng Laboratory, Shenzhen, China

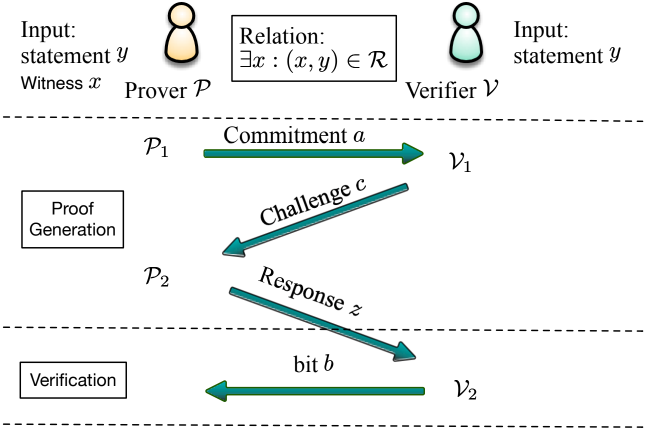
[2]College of Information Science and Technology, Jinan University, Guangzhou, China

[3]Institute of Artificial Intelligence, LMIB, NLSDE, Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing, Beihang University, Beijing, China

Agenda

Background I: Sigma protocols

- Sigma protocols are popular and widely used as a building block in many cryptographic protocols.



Background II: k-out-of-n

- Proving k-out-of-n partial knowledge is well studied.
  - In 1994, Cramer, Damgård and Schoenmakers [CDS94] showed a *general* method.
  - Groth and Kohlweiss [GK15] show how to achieve *logarithmic* (in n) communication when $k = 1$.
  - Attema, Cramer and Fehr [ACF21] achieve logarithmic communication for general k and n in the DL setting.

A relation of k-out-of-n partial knowledge can be informally expressed in disjunctive normal formula (DNF), e.g., when $k = 2$ and $n = 3$,

$$(y_1 \wedge y_2) \vee (y_1 \wedge y_3) \vee (y_2 \wedge y_3),$$

where y_1, y_2, y_3 are 3 different statements, and we call $(y_1 \wedge y_2)$, $(y_1 \wedge y_3)$, $(y_2 \wedge y_3)$ 3 Type-$\wedge$ clauses. There are $C_n^k = 3$ clauses, so we call such relations *complete k-DNF relations*.

Given the relation of complete k-DNF (k-out-of-n), e.g., $(y_1 \wedge y_2) \vee (y_1 \wedge y_3) \vee (y_2 \wedge y_3)$, it is nature to consider the extensions.

- incomplete k-DNF, e.g., $(y_1 \wedge y_2) \vee (y_1 \wedge y_3)$ (less than $C_n^k = 3$ Type-$\wedge$ clauses).
- If we reverse the "$\wedge$" and "$\vee$", we get a relation like

  $$(y_1 \vee y_2) \wedge (y_1 \vee y_3),$$

  where we call $(y_1 \vee y_2)$ and $(y_1 \vee y_3)$ are called 2 Type-$\vee$ clauses. The relation is in *conjunctive normal formula (CNF)*, so we can such relations k-CNF relations.

This paper mainly focus on k-CNF relations (in the discrete logarithm setting).

Motivation II: applications

Relations expressed in CNF are an important collection of relations in practice, e.g.,

- many access control policies are naturally set in CNF and they have been discussed in some attribute-based encryption schemes [JK10, LDL11, CT16, Tsa19];
- instances of the k-SAT problem [IP01].

We also provide a potential application here.
A start-up company wants to show the investors a business plan (building at least a shopping mall in every k neighbouring blocks) in a zero-knowledge manner, avoiding the business roadmap being leaked.

Motivation III: problem

To the best of our knowledge, schemes for k-CNF relations:

- Cramer et al.'s scheme [CDS94]. However, it may lead to super-polynomial communication cost.
- Acyclicity program, proposed by Abe et al. [AAB+21], also works for k-CNF relations, but it is designed for non-interactive zero-knowledge proofs (NIZK), not Sigma protocols. More importantly, it seems impossible to transfer their scheme [AAB+21] into a standard Sigma protocol, so acyclicity program [AAB+21] does not have the strengths of Sigma protocols (i.e., low soundness error by design, high efficiency relative to their generic counterparts, and more flexible).

Therefore, a question is raised naturally: *Is it possible to construct a more efficient Sigma protocol for k-CNF relations?*

Contributions

The contributions of this paper are listed as follows:

- We firstly formally define *partial knowledge for k-CNF relations*. Then, we propose a construction of a Sigma protocol for k-CNF relations in the discrete logarithm (DL) setting, by transferring the k-CNF relations to directed acyclic graphs. Then, we call it DAG-Σ protocol.
- As an extension, we apply our DAG-Σ protocols to construct Sigma protocols for incomplete k-DNF relations in the DL setting, by restricting the choices of statements.
- Finally, we provide an implementation of our DAG-Σ protocol based on elliptic curve groups with key size of 512 bits. It shows that our DAG-Σ protocol saves more than **95%** communication costs and more than **90%** running time, compared with [CDS94], when proving the relations in our experiments.

Theoretical comparison

Table 1: Comparison of some existing protocols (in the DL setting)*

Schemes	Σ?	k-CNF	incomplete k-DNF	complete k-DNF
[CDS94]	Yes	$O(k \cdot num)(\|\mathbb{G}\| + \|\mathbb{Z}_p^*\|)$	$O(k \cdot num)(\|\mathbb{G}\| + \|\mathbb{Z}_p^*\|)$	$O(n)(\|\mathbb{G}\| + \|\mathbb{Z}_p^*\|)$
[GK15]**	Yes	\	\	$O(\log n)(\|\mathbb{G}\| + \|\mathbb{Z}_p^*\|)$
[AAB+20]	Yes	\	$O(n)\|\mathbb{G}\| + O(num)\|\mathbb{Z}_p^*\|$	$O(n)\|\mathbb{G}\| + O(C_n^k)\|\mathbb{Z}_p^*\|$
[AAB+21]	No	$O(n)(\|\mathbb{G}\| + \|\mathbb{Z}_p^*\|)$	\	\
[ACF21]	Yes	\	\	$O(\log(2n-k))\|\mathbb{G}\| + 4 \times \|\mathbb{Z}_p^*\|$
[GGHAK21]	Yes	\	\	$O(k \cdot n)^{\star\star\star}$
Sec. 5.2	Yes	$O(n-k)\|\mathbb{G}\| + O(\|V\|)\|\mathbb{Z}_p^*\|$	\	$O(k)\|\mathbb{G}\| + O(\|V\|)\|\mathbb{Z}_p^*\|^{\dagger}$
Sec. 6‡	Yes	\	$O(n)\|\mathbb{G}\| + O(\|V\|)\|\mathbb{Z}_p^*\|$	\

* The results here are obtained by trivially applying the corresponding protocols. There are n statements and num clauses in the expression of the k-CNF or (in)complete k-DNF relations, where each clause contains k different statements. V denotes the vertices of the DAG in our DAG-Σ protocol ($|V| \leq k \cdot num$, in most cases $|V| \ll k \cdot num$).

** The solution in [GK15] only works for $k = 1$.

*** [GGHAK21] presents a discussion on this kind of relation and the result is directly obtained from the discussion. It involves a special commitment scheme, so we do not have $|\mathbb{G}|$ and $|\mathbb{Z}_p^*|$ here.

† The result is obtained from Remark 1 in the paper.

‡ Our solution in Sec. 6 only works for special language.

Experimental results I: when $k = 4$

our DAG-Σ protocol vs. [CDS94] for k-CNF relations

Communication cost when $k = 4$ ($\times 10^4$ bits)[1]

n	[CDS94]	Our scheme	ratio
10	65.54	1.72	97.37% ↓
15	538.62	4.07	99.24% ↓
20	1964.03	7.45	99.62% ↓
25	5160.96	11.90	99.77% ↓
30	11204.6	17.48	99.84% ↓
40	37412.9	31.92	99.91% ↓
50	94310.4	50.92	99.94% ↓

Running time when $k = 4$ (s)[2]

n	$\mathcal{P}_1$			$\mathcal{P}_2$			$\mathcal{V}_2$		
	[CDS94]	Ours	ratio	[CDS94]	Ours	ratio	[CDS94]	Ours	ratio
10	8.91	0.72	91.87% ↓	0.0049	1.40×10^{-4}	97.11% ↓	10.04	0.85	91.56% ↓
15	57.47	1.92	96.66% ↓	0.033	8.63×10^{-4}	97.27% ↓	65.08	2.13	96.72% ↓
20	182.23	3.91	97.85% ↓	0.11	2.20×10^{-3}	97.95% ↓	187.41	4.13	97.80% ↓
25	456.37	6.54	98.57% ↓	0.33	5.97×10^{-3}	98.20% ↓	477.74	6.66	98.61% ↓
30	1046.45	10.09	99.04% ↓	0.63	5.21×10^{-2}	91.78% ↓	1058.25	10.08	99.05% ↓

[1] ratio $= 1 - \frac{\text{bits of our scheme}}{\text{bits of [CDS94]}} \times 100\%$

[2] ratio $= 1 - \frac{\text{time of our scheme}}{\text{time of [CDS94]}} \times 100\%$.

Experimental results II: more detailed results

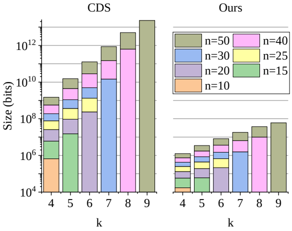


Figure 1: Communication cost

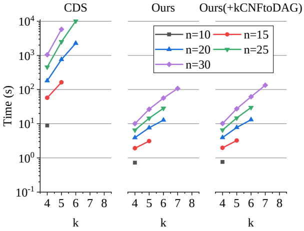


Figure 2: Running time of $\mathcal{P}_1$

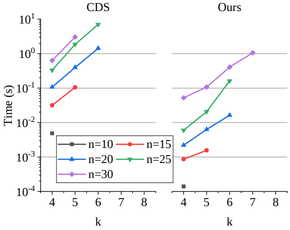


Figure 3: Running time of $\mathcal{P}_2$

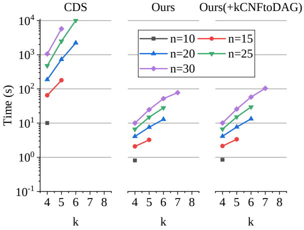


Figure 4: Running time of $\mathcal{V}_2$

Definition of partial knowledge for k-CNF

Let y denote a statement, and $S_k := \{\{i_1, \ldots, i_k\} \mid 1 \leq i_1 < \ldots < i_k \leq n, \{i_1, \ldots, i_k\} \subset [n]\}$. Besides, $(x_l, y_l) \in \mathcal{R}_l$ ($l \in [n]$) denotes a valid witness-statement pair belonging to a relation $\mathcal{R}_l$.

Definition 1 (Partial knowledge for k-CNF)

Given n different statements $(y_l)_{l\in[n]}$, n sub-relations $(\mathcal{R}_l)_{l\in[n]}$, and $S'_k \subseteqq S_k$, the prover proves that for all $\{i_1, \ldots, i_k\} \in S'_k$, she knows the witnesses for at least one of $y_{i_1}, \cdots, y_{i_k}$.

The relation can be presented in CNF as follows,

$$\mathcal{R}_{k\text{-CNF},S'_k} = \{(\mathbf{x}, \mathbf{y}) : \wedge_{\{i_1,\ldots,i_k\}\in S'_k}(\vee_{j\in[k]}(x_{i_j}, y_{i_j}) \in \mathcal{R}_{i_j})\}, \qquad (1)$$

where $\mathbf{x}$, $\mathbf{y}$ are two n-dimension vectors, and $\mathcal{R}_{i_j} \in \{\mathcal{R}_l \mid l \in [n]\}$ is a sub-relation. We denote the relation defined in Eq. (1) as a k-**CNF relation**.

Building block I: kCNFtoDAG (1)

Algorithm kCNFtoDAG is a deterministic algorithm, which transfers k-CNF relations to DAGs. We require that the DAG output by kCNFtoDAG should have the following properties:

- **Property-(i):** Each node in some path corresponds to a statement in the corresponding Type-$\vee$ clause.
- **Property-(ii):** The number of paths from the nodes in S^{source} to the nodes in S^{sink} equals the number of Type-$\vee$ clauses in the expression of $\mathcal{R}_{k\text{-CNF},S'_k}$, and the lengths of these paths are k.

A simple method to implement kCNFtoDAG.

E.g., $\mathcal{R}_1 = \{(\mathbf{x},\mathbf{y}) : (\Sigma_1 \vee \Sigma_2 \vee \Sigma_3) \wedge (\Sigma_1 \vee \Sigma_2 \vee \Sigma_4) \wedge (\Sigma_2 \vee \Sigma_3 \vee \Sigma_5) \wedge (\Sigma_3 \vee \Sigma_4 \vee \Sigma_5)\}$

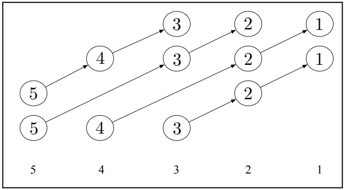


Figure 5: A simple idea

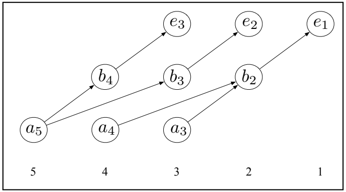


Figure 6: An example for CNF

Building block I: kCNFtoDAG (2)

A counter example that makes the simple method fail.

$$\mathcal{R}_2 = \{(\mathbf{x},\mathbf{y}) : (\Sigma_1 \vee \Sigma_2 \vee \Sigma_3) \wedge (\Sigma_1 \vee \Sigma_2 \vee \Sigma_4) \wedge (\Sigma_1 \vee \Sigma_3 \vee \Sigma_4) \wedge (\Sigma_2 \vee \Sigma_3 \vee \Sigma_5) \wedge (\Sigma_3 \vee \Sigma_4 \vee \Sigma_5)\} \quad (2)$$

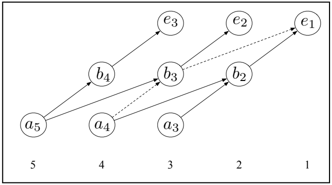


Figure 7: A counter example

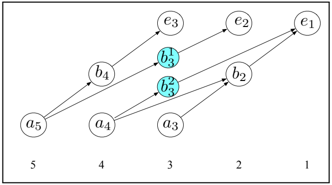


Figure 8: A fixed graph

Building block I: kCNFtoDAG (3)

<u>kCNFtoDAG:</u> 1) Preparing node; 2) Merging prefixes; 3) Merging suffixes

$$\mathcal{R}_2 = \{(\mathbf{x},\mathbf{y}) : (\Sigma_1 \vee \Sigma_2 \vee \Sigma_3) \wedge (\Sigma_1 \vee \Sigma_2 \vee \Sigma_4) \wedge (\Sigma_1 \vee \Sigma_3 \vee \Sigma_4) \wedge (\Sigma_2 \vee \Sigma_3 \vee \Sigma_5) \wedge (\Sigma_3 \vee \Sigma_4 \vee \Sigma_5)\}$$

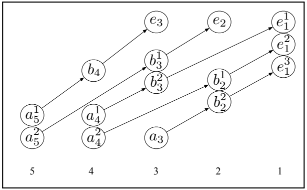


Figure 9: Graph after step 1

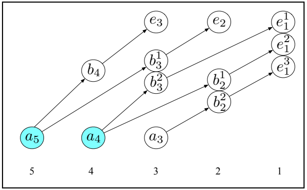


Figure 10: Graph after step 2

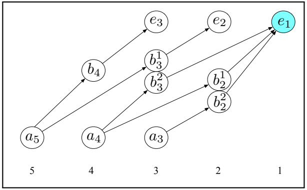


Figure 11: Merging nodes to e_1

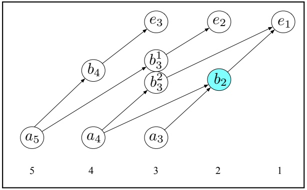


Figure 12: Graph after step 3

Building block I: kCNFtoDAG (4)

Theorem 2 (Upper bound of $|V|$)

Given a k-CNF relation $\mathcal{R}_{k\text{-CNF},S'_k}$ for n statements, the number of vertices $|V|$ in the DAG, output by the above transfer algorithm kCNFtoDAG, *satisfies that* $|V| \leq \mathrm{Min}(V_{\mathrm{bound}}, (k \cdot num))$, *where num is the number of the clauses in the expression of* $\mathcal{R}_{k\text{-CNF},S'_k}$, *and*

$$V_{\mathrm{bound}} = 2^d + 2(n-2d+1) + (n-2d+2)C_n^{\lfloor \frac{d}{2} \rfloor + 1} \begin{cases} d = k \ \ (2 \leq k < \frac{n+1}{2}) \\ d = n-k+1 \\ \qquad (\frac{n+1}{2} \leq k \leq n-1) \end{cases}$$

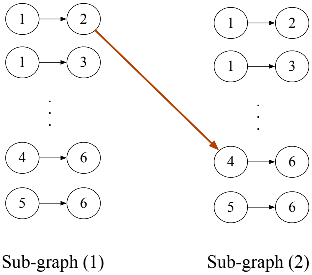
Advantage: it achieves nearly quadratic saving, when comparing the number of vertices in the DAG with the number of statements in the original expression of k-CNF (i.e., $k \cdot num$, where $num \in [1, C_n^k]$).

Building block I: kCNFtoDAG (5)

Another method to analyze the upper bound. Suppose k is an even,

- Prepare two sub-graphs each of which has $C_n^{k/2}$ paths with lengths $k/2$.
- Then for each clause $(y_1 \vee y_2 \vee \ldots \vee y_k)$, find the corresponding path for $(y_1 \vee \ldots \vee y_{k/2})$ in the sub-graph (1) and find the corresponding path for $(y_{k/2+1} \vee \ldots \vee y_k)$ in the sub-graph (2). After that, we add another arrow between the two paths and form a new path with length k.
- Finally, we remove those paths with length $k/2$ and get a DAG.

We can check that the obtained DAG satisfies the properties as defined above.



$e.g., n = 6, k = 4, (y_1 \vee y_2 \vee y_4 \vee y_6)$

$$|V| \leq k/2 \cdot 2 \cdot C_n^{k/2} = k \cdot C_n^{k/2}$$

Building block II: 1-out-of-k in DL setting (1)

Let $\mathcal{R}_{\text{1-OR}}$ be a 1-out-of-k relation in the DL setting, i.e.,

$$\mathcal{R}_{\text{1-OR}} = \{(\mathbf{x}, \mathbf{y}) : y_1 = g^{x_1} \vee \ldots \vee y_k = g^{x_k}\}, \tag{3}$$

where $\mathbf{x} \in (\mathbb{Z}_p^* \cup \{\perp\})^k \setminus \{(\perp)^k\}$ and $\mathbf{y} \in \mathbb{G}^k$.

Recall Schnorr's Sigma protocol in Fig. 13.

Standard mode:
(1) $\mathcal{P}_1(\perp, y)$
$r \leftarrow \mathbb{Z}_p^*$, $a \leftarrow g^r$
Send a to $\mathcal{V}$
(2) $\mathcal{V}_1(a)$:
$c \leftarrow \mathbb{Z}_p^*$
Send c to $\mathcal{P}$
(3) $\mathcal{P}_2(a, c, x, y)$
$z \leftarrow r + cx$
Send z to $\mathcal{V}$

$\mathcal{V}_2(y, a, c, z)$:
$a' \leftarrow g^z / y^c$
Return $(a' \stackrel{?}{=} a)$

Simulator $\mathsf{Sim}(y, c)$:
$z \leftarrow \mathbb{Z}_p^*$, $a \leftarrow g^z / y^c$
Return (a, z)

Chameleon mode:
(1) $\mathcal{P}_1'(\perp, y)$:
$c' \leftarrow \mathbb{Z}_p^*$
$r \leftarrow \mathbb{Z}_p^*$, $a \leftarrow g^r / y^{c'}$
$//\Sigma_{\text{Sch}}^{\mathcal{R}}.\mathsf{Sim}(y, c')$
Send a to $\mathcal{V}$
(2) $\mathcal{V}_1(a)$:
$c \leftarrow \mathbb{Z}_p^*$, Send c to $\mathcal{P}$
(3) $\mathcal{P}_2'(a, c, c', x, y)$:
$z \leftarrow r + (c - c')x$
Send z to $\mathcal{V}$

Figure 13: Schnorr's Sigma protocol $\Sigma_{\text{Sch}}^{\mathcal{R}}$

Building block II: 1-out-of-k in DL setting (2)

$$\mathcal{R}_{\text{1-OR}} = \{(\mathbf{x}, \mathbf{y}) : y_1 = g^{x_1} \vee \ldots \vee y_k = g^{x_k}\}$$

$$\mathcal{P}_1 \quad a_1 = g^{z_1}/y_1^{\mathsf{H}(a_2)} \leftarrow \ldots \leftarrow a_\mu = g^{z_\mu}/y_\mu^{\mathsf{H}(a_{\mu+1})} \ldots \leftarrow a_{k-1} = g^{z_{k-1}}/y_{k-1}^{\mathsf{H}(a_k)} \leftarrow a_k = g^r$$

$$\mathcal{P}_2 \quad \underbrace{a'_1 = g^{z'_1}/y_1^{\mathsf{H}(a'_2)} \leftarrow \ldots \leftarrow a'_\mu = g^{z'_\mu}/y_\mu^{\mathsf{H}(a'_{\mu+1})}}_{a_i = a'_i (1 \le i \le \mu)} \ldots \leftarrow a'_{k-1} = g^{z'_{k-1}}/y_{k-1}^{\mathsf{H}(a'_k)} \leftarrow a'_k = g^{z'_k}/y_k^c$$

$$(z'_1 = z_1\,,\ \ldots\ ,z'_{\mu-1} = z_{\mu-1}\,,\ \boxed{z'_\mu = z_\mu + (\mathsf{H}(a'_{\mu+1}) - \mathsf{H}(a_{\mu+1}))x_u}\,,\ z'_{\mu+1} \leftarrow \mathbb{Z}_p^*\,,\ \ldots\ ,\ z'_k \leftarrow \mathbb{Z}_p^*)$$

Figure 14: An example of the proof of 1-out-of-k partial knowledge

where

- $\mathbf{x} = (x_1, \ldots, x_k)$ and $\mathbf{y} = (y_1, \ldots, y_k)$ denote the witnesses and statements respectively;
- the witness x_μ for statement y_μ is known by the prover;
- $\mathsf{H} : \mathbb{G} \rightarrow \mathbb{Z}_p^*$ is a collision-resistance hash function.

Advantage: the prover $\mathcal{P}$ only needs send one commitment a_1 to the verifier $\mathcal{V}$.

A DAG-based Sigma protocol $\Sigma_{\mathrm{DAG}}^{\mathcal{R}^{\mathrm{dl}}_{k\text{-CNF},S'_k}}$ (1)

A k-CNF relation in DL setting is as follows:

$$\mathcal{R}^{\mathrm{dl}}_{k\text{-CNF},S'_k} = \{(\mathbf{x},\mathbf{y}) : \wedge_{\{i_1,\ldots,i_k\}\in S'_k}(\vee_{j\in[k]}\ y_{i_j} = g^{x_{i_j}})\},$$

where $\mathbf{x} \in (\mathbb{Z}_p^* \cup \{\perp\})^n \setminus \{(\perp)^n\}$, $\mathbf{y} \in \mathbb{G}^n$, S'_k is defined as previously, and for all $\{i_1,\ldots,i_k\} \in S'_k$, $1 \le i_1 < \ldots < i_k \le n$.

$\underline{\Sigma_{\mathrm{DAG}}^{\mathcal{R}^{\mathrm{dl}}_{k\text{-CNF},S'_k}}}$:

1) run kCNFtoDAG get a DAG;

2) run a proving algorithm (similar to that in 1-out-of-k) for each path in the DAG.

A DAG-based Sigma protocol $\Sigma_{\mathrm{DAG}}^{\mathcal{R}^{\mathrm{dl}}_{k\text{-CNF},S'_k}}$ (2)

The *difference* between the proving algorithm in $\Sigma_{\mathrm{DAG}}^{\mathcal{R}^{\mathrm{dl}}_{k\text{-CNF},S'_k}}$ and that in 1-out-of-k.

$$\mathcal{R}_{1\text{-OR}} \xrightarrow{\text{kCNFtoDAG}} \text{ⓚ} \rightarrow \cdots \rightarrow \text{②} \rightarrow \text{①}$$

$$\mathcal{R}_{1\text{-OR}} = \{(\mathbf{x},\mathbf{y}) : y_1 = g^{x_1} \vee \ldots \vee y_k = g^{x_k}\}$$

$$\mathcal{P}_1 \quad a_1 = g^{z_1}/y_1^{\mathsf{H}(a_2)} \leftarrow \ldots \leftarrow a_\mu = g^{z_\mu}/y_\mu^{\mathsf{H}(a_{\mu+1})} \ldots \leftarrow a_{k-1} = g^{z_{k-1}}/y_{k-1}^{\mathsf{H}(a_k)} \leftarrow a_k = g^r$$

$$\mathcal{P}_2 \quad \underbrace{a'_1 = g^{z'_1}/y_1^{\mathsf{H}(a'_2)} \leftarrow \ldots \leftarrow a'_\mu = g^{z'_\mu}/y_\mu^{\mathsf{H}(a'_{\mu+1})}}_{a_i = a'_i\,(1 \le i \le \mu)} \ldots \leftarrow a'_{k-1} = g^{z'_{k-1}}/y_{k-1}^{\mathsf{H}(a'_k)} \leftarrow a'_k = g^{z'_k}/y_k^c$$

$$(z'_1 = z_1\ ,\ \ldots\ ,z'_{\mu-1} = z_{\mu-1}\ ,\ \boxed{z'_\mu = z_\mu + (\mathsf{H}(a'_{\mu+1}) - \mathsf{H}(a_{\mu+1}))x_u}\ ,\ z'_{\mu+1} \leftarrow \mathbb{Z}_p^*\ ,\ \ldots\ ,\ z'_k \leftarrow \mathbb{Z}_p^*)$$

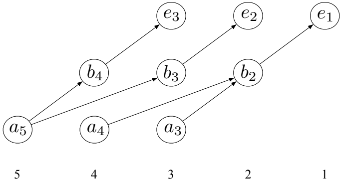
Figure 15: An example of the proof of 1-out-of-k partial knowledge

A DAG-based Sigma protocol $\Sigma_{\text{DAG}}^{\mathcal{R}^{\text{dl}}_{k\text{-CNF},S'_k}}$ (3)

The *difference* between the proving algorithm in $\Sigma_{\text{DAG}}^{\mathcal{R}^{\text{dl}}_{k\text{-CNF},S'_k}}$ and that in 1-out-of-k.

$E.g.$, $\mathcal{R}_1 = \{(\mathbf{x},\mathbf{y}) : (\Sigma_1 \vee \Sigma_2 \vee \Sigma_3) \wedge (\Sigma_1 \vee \Sigma_2 \vee \Sigma_4) \wedge (\Sigma_2 \vee \Sigma_3 \vee \Sigma_5) \wedge (\Sigma_3 \vee \Sigma_4 \vee \Sigma_5)\}$

$\mathcal{R}_1 \xrightarrow{\text{kCNFtoDAG}}$:



Then, when we compute the commitment of node b_2, it depends on the commitments of nodes a_3 and a_4 ($\varphi : \{0,1\}^* \to \mathbb{Z}_p^*$ is a collision-resistance hash function and $z_{b_2} \leftarrow \mathbb{Z}_p^*$):

$$a_{b_2} = g^{z_{b_2}} / y_{b_2}^{\varphi(a_{a_3} || a_{a_4})}.$$

Conclusion

Security analysis.

Theorem 3 (Security of $\Sigma_{\mathrm{DAG}}^{\mathcal{R}^{\mathrm{dl}}_{k\text{-CNF},S'_k}}$)

If φ is a collision-resistant hash function, $\Sigma_{\mathrm{DAG}}^{\mathcal{R}^{\mathrm{dl}}_{k\text{-CNF},S'_k}}$ provides computational knowledge soundness and is special HVZK.

Communication complexity.

It is clear that there are $|S^{\mathrm{sink}}| \leq (n-k+1)$ group elements and $(|V|+1)$ elements in $\mathbb{Z}_p^*$ in the communication of the 3-move Sigma protocol $\Sigma_{\mathrm{DAG}}^{\mathcal{R}^{\mathrm{dl}}_{k\text{-CNF},S'_k}}$. According to the theorem about kCNFtoDAG, $|V| \leq \mathrm{Min}(V_{\mathrm{bound}}, (k \cdot num))$, which implies that $|V| \leq k \cdot num$. Note that the communication complexity of [CDS94] is $O(k \cdot num)$, so we can draw such a conclusion that the communication complexity of $\Sigma_{\mathrm{DAG}}^{\mathcal{R}^{\mathrm{dl}}_{k\text{-CNF},S'_k}}$ is better than that of [CDS94].

References I

[AAB+20] Masayuki Abe, Miguel Ambrona, Andrej Bogdanov, Miyako Ohkubo, and Alon Rosen.
Non-interactive composition of sigma-protocols via share-then-hash.
In *ASIACRYPT 2020*, pages 749–773. Springer, 2020.

[AAB+21] Masayuki Abe, Miguel Ambrona, Andrej Bogdanov, Miyako Ohkubo, and Alon Rosen.
Acyclicity programming for sigma-protocols.
In *TCC 2021*, pages 435–465. Springer, 2021.

[ACF21] Thomas Attema, Ronald Cramer, and Serge Fehr.
Compressing proofs of k-out-of-n partial knowledge.
In *CRYPTO 2021*, pages 65–91. Springer, 2021.

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

[CT16] Sébastien Canard and Viet Cuong Trinh.
Constant-size ciphertext attribute-based encryption from multi-channel broadcast encryption.
In *ICISS 2016*, pages 193–211. Springer, 2016.

[GGHAK21] Aarushi Goel, Matthew Green, Mathias Hall-Andersen, and Gabriel Kaptchuk.
Stacking sigmas: A framework to compose σ-protocols for disjunctions.
Cryptology ePrint Archive, 2021.

[GK15] Jens Groth and Markulf Kohlweiss.
One-out-of-many proofs: Or how to leak a secret and spend a coin.
In *EUROCRYPT 2015*, pages 253–280. Springer, 2015.

[IP01] Russell Impagliazzo and Ramamohan Paturi.
On the complexity of k-sat.
Journal of Computer and System Sciences, 62(2):367–375, 2001.

References II

[JK10] Pascal Junod and Alexandre Karlov.
An efficient public-key attribute-based broadcast encryption scheme allowing arbitrary access policies.
In *Proceedings of the tenth annual ACM workshop on Digital rights management*, pages 13–24, 2010.

[LDL11] Junzuo Lai, Robert H Deng, and Yingjiu Li.
Fully secure cipertext-policy hiding cp-abe.
In *Information Security Practice and Experience 2011*, pages 24–39. Springer, 2011.

[Tsa19] Rotem Tsabary.
Fully secure attribute-based encryption for t-cnf from lwe.
In *CRYPTO 2019*, pages 62–85. Springer, 2019.

Thanks!