Supplementary Material

Zero-knowledge proofs

A zero-knowledge (ZK) proof [31] is a two party protocol executed between a prover and a verifier that allows the prover to convince the verifier about the validity of certain statement, without revealing any other information, e.g., why the statement is true. More formally, given a binary relation $R : \mathcal{X} \times \mathcal{W} \to \{0,1\}$ defined over a set of statements $\mathcal{X}$ and a set of witnesses $\mathcal{W}$, let L_R be the language defined as $L_R \coloneqq \{x \in \mathcal{X} \mid \exists w \in \mathcal{W} : R(x,w) = 1\}$. A zero-knowledge proof system allows a prover in possession of $(x,w) \in R$ to convince a verifier of the fact that $x \in L_R$, without revealing any information about w. Non-interactive ZK proof systems [13] are a version of ZK proof systems where the prover sends one single message to the verifier.

A ZK proof system is said to be *correct* if an honest prover (in possession of a witness for certain instance) can always convince an honest verifier. A ZK proof system is said to have the *zero-knowledge property* if there exists a *simulator* that, without any witness (but possibly some useful piece of information like a trapdoor), can produce proofs that look indistinguishable from proofs produced by an honest prover in possession of a valid witness. For security parameter λ, we denote by $\varepsilon_{\mathsf{zk}}(\lambda)$ an upper-bound on advantage of any PPT machine that tries to distinguish between real and simulated proofs. A ZK proof system is said to be a *proof-of-knowledge* if there exists an *extractor* that, given an instance $x \in \mathcal{X}$ and given access to an arbitrary prover algorithm, can extract a valid witness for x with almost the same probability the prover produces a valid proof for x. We denote by $\varepsilon_{\mathsf{ext}}(\lambda)$ the *extraction error*, an upper-bound on the gap of the probabilities of the extractor being successful and the prover being successful.

We denote a *zero-knowledge proof of knowledge of secret w satisfying $R(x,w)$ for some public R and x* as $\mathrm{PoK}\{(x) : R(x,w) = 1\}$.

A Σ-protocol is a simple 3-move interactive protocol used for building very efficient ZK proof systems. We refer to [29] for a formal definition of Σ-protocols.

Dual System Groups

The compilers of attribute-based encryption from predicate encodings [20] and pair encodings [3,8] rely on dual system groups (DSG) [23,24] in a black-box way. A dual system group is a tuple of six efficiently computable algorithms:

- $\mathsf{SampP}(1^\lambda, 1^n)$: on input the security parameter and an integer n, outputs public parameters pp and secret parameters sp such that:
  - The public parameters, pp, include a triple of abelian groups $(\mathsf{G}, \mathsf{H}, \mathsf{G_t})$ (that are $\mathbb{Z}_p$-modules for some λ-bits prime p), a non-degenerate bilinear map $e : \mathsf{G} \times \mathsf{H} \to \mathsf{G_t}$, an homomorphism μ (defined over H) and additional parameters required by SampP and SampH.
  - Given pp, it is possible to uniformly sample to H.

- The secret parameters, sp, include a distinguished element $h^* \in \mathsf{H}$ (different from the unit) and additional parameters required by $\widehat{\mathsf{SampG}}$ and $\widehat{\mathsf{SampH}}$.

- $\mathsf{SampG}(\mathsf{pp})$ and $\widehat{\mathsf{SampG}}(\mathsf{pp}, \mathsf{sp})$ output an element from G^{n+1}.
- $\mathsf{SampH}(\mathsf{pp})$ and $\widehat{\mathsf{SampH}}(\mathsf{pp}, \mathsf{sp})$ output an element from H^{n+1}.
- SampGT is a function defined from $Im(\mu)$ to $\mathsf{G_t}$.

Additional conditions are required for correctness and security:

projective: For all public parameters, pp, every $h \in \mathsf{H}$ and all coin tosses σ, it holds $\mathsf{SampGT}(\mu(h); \sigma) = e(g_0, h)$, where $(g_0, g_1, \ldots, g_n) \leftarrow \mathsf{SampG}(\mathsf{pp}; r)$.

associative: Let $(g_0, g_1, \ldots, g_n) \leftarrow \mathsf{SampG}(\mathsf{pp})$, $(h_0, h_1, \ldots, h_n) \leftarrow \mathsf{SampH}(\mathsf{pp})$, it holds $e(g_0, h_i) = e(g_i, h_0)$ for every $i \in [n]$.

H-subgroup: $\mathsf{SampH}(\mathsf{pp})$ is the uniform distribution over a subgroup of H^{n+1}.

orthogonality: $h^* \in Kernel(\mu)$.

non-degeneracy: For every $(h_0, h_1, \ldots, h_n) \leftarrow \mathsf{SampH}(\mathsf{pp})$, $h^* \in \langle h_0 \rangle$. Furthermore, for every $(\hat{g}_0, \hat{g}_1, \ldots, \hat{g}_n) \leftarrow \widehat{\mathsf{SampG}}(\mathsf{pp}, \mathsf{sp})$, $(\alpha \xleftarrow{\$} \mathbb{Z}_p;\ \mathsf{return}\ e(\hat{g}_0, h^*)^\alpha)$ is the uniform distribution over $\mathsf{G_t}$.

left-subgroup indistinguishability: $(\mathsf{pp},\ \boldsymbol{g}) \approx_c (\mathsf{pp},\ \boldsymbol{g}\cdot\hat{\boldsymbol{g}})$.

right-subgroup indistinguishability: $(\mathsf{pp},\ h^*,\ \boldsymbol{g}\cdot\hat{\boldsymbol{g}},\ \boldsymbol{h}) \approx_c (\mathsf{pp},\ h^*,\ \boldsymbol{g}\cdot\hat{\boldsymbol{g}},\ \boldsymbol{h}\cdot\hat{\boldsymbol{h}})$.

parameter-hiding: $(\mathsf{pp},\ h^*,\ \hat{\boldsymbol{g}},\ \hat{\boldsymbol{h}}) \equiv (\mathsf{pp},\ h^*,\ \hat{\boldsymbol{g}}\cdot\hat{\boldsymbol{g}}',\ \hat{\boldsymbol{h}}\cdot\hat{\boldsymbol{h}}')$.

Where, $\approx_c$ denotes a distinguishing probability upper-bounded by a negligible function on λ and, for any $n \in \mathbb{N}$, the above elements are sampled as:

$$(\mathsf{pp}, \mathsf{sp}) \leftarrow \mathsf{SampP}(1^\lambda, 1^n)$$

$$\begin{array}{lll} \boldsymbol{g} \leftarrow \mathsf{SampG}(\mathsf{pp}) & \hat{\boldsymbol{g}} \leftarrow \widehat{\mathsf{SampG}}(\mathsf{pp}, \mathsf{sp}) & \hat{\boldsymbol{g}}' \coloneqq (1_\mathsf{G}, \hat{\boldsymbol{g}}_0^{z_1}, \ldots, \hat{\boldsymbol{g}}_0^{z_n}) \\ \boldsymbol{h} \leftarrow \mathsf{SampG}(\mathsf{pp}) & \hat{\boldsymbol{h}} \leftarrow \widehat{\mathsf{SampG}}(\mathsf{pp}, \mathsf{sp}) & \hat{\boldsymbol{h}}' \coloneqq (1_\mathsf{H}, \hat{\boldsymbol{h}}_0^{z_1}, \ldots, \hat{\boldsymbol{h}}_0^{z_n}) \end{array}$$

for $z_1, \ldots, z_n \xleftarrow{\$} \mathbb{Z}_p$.

Remark. Observe that we have presented the version of dual system groups defined in [20]. Other works consider slightly different conditions (e.g., the non-degeneracy of [3]). However, the instantiation of DSG from k-Lin also satisfies the properties of those variations.

Dual system groups from k-lin

Here, we consider the following instantiation (from [20]), which is a variant of the DSG construction from [24] and realizes dual system groups based on the k-lin assumption, for $k \in \mathbb{N}$.

Definition 1 (k-lin assumption). *Let $\mathcal{D}_k$ be the distribution that, for a λ-bits prime p, outputs a matrix $D \in \mathbb{Z}_p^{(k+1)\times k}$ and a vector in $\boldsymbol{d}^\perp \in \mathbb{Z}_p^{k+1}$ defined as: $d_1, \ldots, d_k \xleftarrow{\$} \mathbb{Z}_p^*$,*

$$D := \begin{pmatrix} d_1 & & \\ & \ddots & \\ & & d_k \\ 1 & \ldots & 1 \end{pmatrix} \qquad \boldsymbol{d}^\perp := \begin{pmatrix} d_1^{-1} \\ \vdots \\ d_k^{-1} \\ -1 \end{pmatrix}$$

The k-linear assumption over a group G of order p (with generator g), establishes that all PPT algorithms have negligible advantage (in λ) on distinguishing between the following two distributions:

$$\{\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_p^k; \mathsf{return}\ (g^D, g^{D\boldsymbol{s}})\} \quad \approx_c \quad \{\boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_p^{k+1}; \mathsf{return}\ (g^D, g^{\boldsymbol{r}})\} \ .$$

The following is the construction of dual system groups based on distribution $\mathcal{D}_k$ from [20]:

- $\mathsf{SampP}(1^\lambda, 1^n)$:
  1. Let $\mathcal{G}$ be an asymmetric prime-order bilinear group generator and compute $(p, G_1, G_2, G_\mathsf{t}, g_1, g_2, \hat{e}) \leftarrow \mathcal{G}(1^\lambda)$.
  2. Set G, H and $\mathsf{G_t}$ as G_1^{k+1}, G_2^{k+1} and G_t respectively and let $e : \mathsf{G} \times \mathsf{H} \to \mathsf{G_t}$ be defined as $\hat{e}$ applied coordinate-wise.
  3. Sample $(A, \boldsymbol{a}^\perp) \leftarrow \mathcal{D}_k$, $(B, \boldsymbol{b}^\perp) \leftarrow \mathcal{D}_k$.
  4. Sample matrices $W_1, \ldots, W_n$, uniformly from $\mathbb{Z}_p^{(k+1)\times(k+1)}$.
  5. Let $\mu : \mathsf{H} \to \mathsf{G_t^k}$ be defined $[\![\boldsymbol{k}]\!]_2 \mapsto [\![A^\top \boldsymbol{k}]\!]_\mathsf{t}$ and $h^* := [\![\boldsymbol{a}^\perp]\!]_2$.
  6. Define

$$\mathsf{pp} := \begin{pmatrix} (p, \mathsf{G}, \mathsf{H}, \mathsf{G_t}, e) \\ [\![A]\!]_1, [\![W_1^\top A]\!]_1, \ldots, [\![W_n^\top A]\!]_1 \\ [\![B]\!]_2, [\![W_1 B]\!]_2, \ldots, [\![W_n B]\!]_2 \end{pmatrix} \qquad \mathsf{sp} := (\boldsymbol{a}^\perp, \boldsymbol{b}^\perp, W_1, \ldots, W_n)$$

- $\mathsf{SampG}(\mathsf{pp})$: Sample $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_p^k$, return $([\![A\boldsymbol{s}]\!]_1,\ [\![W_1^\top A\boldsymbol{s}]\!]_1, \ldots,\ [\![W_n^\top A\boldsymbol{s}]\!]_1)$.
- $\widehat{\mathsf{SampG}}(\mathsf{sp}, \mathsf{pp})$: Sample $s \xleftarrow{\$} \mathbb{Z}_p^*$, return $([\![s\boldsymbol{b}^\perp]\!]_1, [\![sW_1\boldsymbol{b}^\perp]\!]_1, \ldots, [\![sW_n\boldsymbol{b}^\perp]\!]_1)$.
- $\mathsf{SampH}(\mathsf{pp})$: Sample $\boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_p^k$, return $([\![B\boldsymbol{r}]\!]_2,\ [\![W_1 B\boldsymbol{r}]\!]_2, \ldots,\ [\![W_n B\boldsymbol{r}]\!]_2)$.
- $\widehat{\mathsf{SampH}}(\mathsf{sp}, \mathsf{pp})$: Sample $r \xleftarrow{\$} \mathbb{Z}_p^*$, return $([\![r\boldsymbol{a}^\perp]\!]_2, [\![rW_1\boldsymbol{a}^\perp]\!]_2, \ldots, [\![rW_n\boldsymbol{a}^\perp]\!]_2)$.

Attribute-based encryption from pair encodings

Definition 2 (Attribute-based encryption from pair encodings). *Given a Υ-pair encoding* $(\mathsf{EncCt}, \mathsf{EncKey}, \mathsf{Pair})$*, with* $\Upsilon = (\bar{w}, w, \hat{w}, \bar{m}, m, \hat{m}, n)$*, for predicate* $P : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$*, an attribute-based encryption scheme can be constructed as follows:*

- $\mathsf{Setup}(1^\lambda, \mathcal{X}, \mathcal{Y})$*: run the DSG generation algorithm* $\mathsf{SampP}(1^\lambda, 1^n)$ *to obtain* pp *and* sp*. Let* $\mathsf{msk} \xleftarrow{\$} \mathsf{H}$ *and let* $\mathsf{mpk} = (\mathsf{pp}, \mu(\mathsf{msk}))$*. Output* $(\mathsf{mpk}, \mathsf{msk})$*.*

- $\mathsf{Enc}(\mathsf{mpk}, m, x)$*: run* $\mathsf{EncCt}(p, x)$ *to obtain polynomials* $\boldsymbol{c}_x(\boldsymbol{s}, \hat{\boldsymbol{s}}, \boldsymbol{b})$*. For every* $\ell \in [c]$*, let the* ℓ*-th polynomial in* $\boldsymbol{c}_x$ *be*

$$\sum_{i \in [\hat{w}]} \gamma_i^{(\ell)} \hat{s}_i + \sum_{i \in [0,w]} \sum_{j \in [n]} \gamma_{\{i,j\}}^{(\ell)} s_i b_j$$

for some coefficients $\gamma_i^{(\ell)}$ *and* $\gamma_{\{i,j\}}^{(\ell)}$ *in* $\mathbb{Z}_p$*. Now, run* SampG *to produce*

$$\begin{aligned}
(\hat{g}_{\{i,0\}}, \hat{g}_{\{i,1\}}, \ldots, \hat{g}_{\{i,n\}}) &\leftarrow \mathsf{SampG}(\mathsf{pp}) \qquad \text{for } i \in [\hat{w}] \\
(g_{\{i,0\}}, g_{\{i,1\}}, \ldots, g_{\{i,n\}}) &\leftarrow \mathsf{SampG}(\mathsf{pp}) \qquad \text{for } i \in [w] \\
(g_{\{0,0\}}, g_{\{0,1\}}, \ldots, g_{\{0,n\}}) &\leftarrow \mathsf{SampG}(\mathsf{pp}; \sigma)
\end{aligned}$$

Observe that we have made explicit the coin tosses, σ*, used in the last sampling. Let* $\mathsf{ct}^\star \coloneqq m \cdot \mathsf{SampGT}(\mu(\mathsf{msk}); \sigma)$ *and let the ciphertext* ct_x *be* $\mathsf{ct}_x \coloneqq (\mathsf{ct}_0, \mathsf{ct}_1, \ldots, \mathsf{ct}_w, \widetilde{\mathsf{ct}}_1, \ldots, \widetilde{\mathsf{ct}}_c, \mathsf{ct}^\star)$ *where for every* $i \in [0, w]$*,* $\mathsf{ct}_i \coloneqq g_{\{i,0\}}$*; and for every* $\ell \in [c]$*,* $\widetilde{\mathsf{ct}}_\ell$ *is computed as*

$$\widetilde{\mathsf{ct}}_\ell \coloneqq \prod_{i \in [\hat{w}]} \hat{g}_{\{i,0\}}^{\gamma_i^{(\ell)}} \cdot \prod_{i \in [0,w]} \prod_{j \in [n]} g_{\{i,j\}}^{\gamma_{\{i,j\}}^{(\ell)}} \ .$$

Output ct_x*.*

- $\mathsf{KeyGen}(\mathsf{msk}, y)$*: run* $\mathsf{EncKey}(p, y)$ *to obtain polynomials* $\boldsymbol{k}_y(\boldsymbol{r}, \hat{\boldsymbol{r}}, \boldsymbol{b})$*. For every* $\ell \in [k]$*, let the* ℓ*-th polynomial in* $\boldsymbol{k}_y$ *be*

$$\phi^{(\ell)} \alpha + \sum_{i \in [\hat{m}]} \phi_i^{(\ell)} \hat{r}_i + \sum_{i \in [m]} \sum_{j \in [n]} \phi_{\{i,j\}}^{(\ell)} r_i b_j$$

for some coefficients $\phi^{(\ell)}$*,* $\phi_i^{(\ell)}$ *and* $\phi_{\{i,j\}}^{(\ell)}$ *in* $\mathbb{Z}_p$*. Now, run* SampH *to produce*

$$\begin{aligned}
(\hat{h}_{\{i,0\}}, \hat{h}_{\{i,1\}}, \ldots, \hat{h}_{\{i,n\}}) &\leftarrow \mathsf{SampH}(\mathsf{pp}) \qquad \text{for } i \in [\hat{m}] \\
(h_{\{i,0\}}, h_{\{i,1\}}, \ldots, h_{\{i,n\}}) &\leftarrow \mathsf{SampH}(\mathsf{pp}) \qquad \text{for } i \in [m]
\end{aligned}$$

Let the secret key sk_y *be* $\mathsf{sk}_y \coloneqq (\mathsf{sk}_1, \ldots, \mathsf{sk}_m, \widetilde{\mathsf{sk}}_1, \ldots, \widetilde{\mathsf{sk}}_k)$ *where for every* $i \in [m]$*,* $\mathsf{sk}_i \coloneqq h_{\{i,0\}}$*; and for every* $\ell \in [k]$*,* $\widetilde{\mathsf{sk}}_\ell$ *is computed as*

$$\widetilde{\mathsf{sk}}_\ell \coloneqq \mathsf{msk}^{\phi^{(\ell)}} \cdot \prod_{i \in [\hat{m}]} \hat{h}_{\{i,0\}}^{\phi_i^{(\ell)}} \cdot \prod_{i \in [m]} \prod_{j \in [n]} h_{\{i,j\}}^{\phi_{\{i,j\}}^{(\ell)}} \ .$$

Output sk_y*.*

- $\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_y, \mathsf{ct}_x, x)$*: run* $\mathsf{Pair}(p, x, y)$ *to obtain matrices* E, F *(note that* y *is assumed to be extractable from* sk_y*, whereas* x *is explicitly included as an input to* Dec*). Let* κ *be*

$$\kappa \coloneqq \prod_{i\in[w]} \prod_{\ell\in[k]} e(\mathsf{ct}_i, \widetilde{\mathsf{sk}}_\ell)^{E_{i,\ell}} \cdot \prod_{\ell\in[c]} \prod_{i\in[m]} e(\widetilde{\mathsf{ct}}_\ell, \mathsf{sk}_i)^{F_{\ell,i}}$$

Output the message $\mathsf{ct}^\star/\kappa$*.*