Quantum-safe Hierarchical Deterministic Keys with the pqARKG-H extension

Emil Lundberg
Yubico AB
emil@yubico.com

February 2, 2025

Abstract

Hierarchical Deterministic Keys (HDK) is a draft specification using key blinding techniques including ARKG to create hierarchies of keys all derived from a single root secret key, in such a way that only the root secret key needs to be kept in a secure environment and new public keys can be generated without access to the root secret key. However, when ARKG is used in HDK, the original ARKG procedures would require one invocation of DeriveSK per HDK layer when exercising derived private keys. This is undesirable in contexts where each invocation would require a distinct user gesture.

To address this we analyze the security of pqARKG-H, an extension of pqARKG for better compatibility with HDK. pqARKG is in turn a post-quantum compatible generalization of ARKG. pqARKG-H enables any number of HDK layers with a single call to the root private key. We demonstrate that pqARKG-H retains the key security properties of pqARKG by a straightforward reduction of the key security experiment.

1 Introduction

Hierarchical Deterministic Keys (HDK) [Dij25] is a draft specification using key blinding techniques including ARKG [LB25] to create hierarchies of keys all derived from a single root secret key, in such a way that only the root secret key needs to be kept in a secure environment and new public keys can be generated without access to the root secret key. However, when ARKG is used in HDK, the original ARKG procedures would require one invocation of DeriveSK per HDK layer when exercising derived private keys. This is undesirable in contexts where each invocation would require a distinct user gesture. One such context is the WebAuthn “sign” extension [Lun24] proposed to introduce general-purpose signing capabilities to the Web Authentication API (WebAuthn) [Bal+25], including signing using private keys derived via ARKG. Since WebAuthn requires a user gesture for any signing operation, HDK operations based on this “sign” extension would also require one user gesture per HDK layer.

The ARKG construction in [LB25] is based upon pqARKG [Wil23], a post-quantum compatible generalization of the original ARKG [Fry+20]. Here we propose and analyze pqARKG-H, an extension of pqARKG for better compatibility with HDK. pqARKG-H enables any number of HDK layers with a single invocation of the $\mathsf{DeriveSK}$ procedure.

The next section repeats the definition of pqARKG and its msKS security experiment. Then we propose the extension pqARKG-H and its corresponding security experiment, and finally demonstrate by a straightforward reduction that pqARKG-H retains the msKS property of pqARKG.

2 pqARKG

Here we repeat the definition of pqARKG and its msKS security experiment [Wil23]. The instance parameters are a key blinding scheme Δ, a key encapsulation mechanism (KEM) Π and a pseudo-random function PRF outputting blinding factors in the domain of Δ.

pqARKG is defined as the suite of procedures in Figure 2. The msKS security experiment for pqARKG is defined in Figure 2. [1]

3 pqARKG-H

Here we define our modified pqARKG-H scheme and a corresponding msKS security experiment.

A new parameter b is added to the $\mathsf{DerivePK}$ and $\mathsf{DeriveSK}$ functions. This b is an additional blinding factor in the key blinding scheme Δ, allowing the ARKG delegating party (the party holding the ARKG private seed) to add any number of additional blinding layers on top of the one performed by the ARKG subordinate party (the party generating public keys). To prevent choosing $b = -\tau$ so that it cancels the blinding factor τ computed in step 2 of $\mathsf{DeriveSK}$ of pqARKG, this b is also mixed into the PRF arguments to compute τ. This disrupts any algebraic relationship between b and τ, thus preventing the a compromised subordinate party from extracting the private seed sk by a malicious choice of b.

The new argument mixed into the PRF is however not b directly, but a blinded public key pk_{Δ}^{b} incorporating the blinding factor b. This enables the delegating party to share a derived public pqARKG-H seed with a subordinate party without having to disclose b. This is desirable if the delegating

[1] We have corrected a misprint in [Wil23] writing τ in place of k on line 3 of $\mathcal{O}'_{\mathsf{sk}}$, and renamed the variable ck to τ in $\mathcal{O}'_{\mathsf{pk}}$.

$\mathsf{Setup}(\lambda)$

$\textbf{return } \mathsf{pp} = (\Delta, \Pi, \mathsf{PRF})$

$\mathsf{Check}(\mathsf{pp}, \mathsf{sk}', \mathsf{pk}')$

$\textbf{return } \Delta.\mathsf{Check}(\mathsf{sk}', \mathsf{pk}')$

$\mathsf{KGen}(\mathsf{pp})$

$(\mathsf{sk}_\Delta, \mathsf{pk}_\Delta) \leftarrow \Delta.\mathsf{KGen}()$

$(\mathsf{sk}_\Pi, \mathsf{pk}_\Pi) \leftarrow \Pi.\mathsf{KGen}()$

$\textbf{return } \mathsf{sk} = (\mathsf{sk}_\Delta, \mathsf{sk}_\Pi), \mathsf{pk} = (\mathsf{pk}_\Delta, \mathsf{pk}_\Pi)$

$\mathsf{DerivePK}(\mathsf{pp}, \mathsf{pk} = (\mathsf{pk}_\Delta, \mathsf{pk}_\Pi), \mathsf{aux})$

$(c, k) \leftarrow \Pi.\mathsf{Encaps}(\mathsf{pk}_\Pi)$

$\tau \leftarrow \mathsf{PRF}(k, \mathsf{aux})$

$\mathsf{pk}' \leftarrow \Delta.\mathsf{BlindPK}(\mathsf{pk}_\Delta, \tau)$

$\textbf{return } \mathsf{pk}', \mathsf{cred} = (c, \mathsf{aux})$

$\mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk} = (\mathsf{sk}_\Delta, \mathsf{sk}_\Pi), \mathsf{cred} = (c, \mathsf{aux}))$

$k \leftarrow \Pi.\mathsf{Decaps}(\mathsf{sk}_\Pi, c)$

$\tau \leftarrow \mathsf{PRF}(k, \mathsf{aux})$

$\textbf{return } \Delta.\mathsf{BlindSK}(\mathsf{sk}_\Delta, \tau)$

Figure 1: Algorithms of the pqARKG scheme [Wil23].

$\mathsf{Exp}^{\mathsf{msKS}}_{pqARKG,\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$

$\mathsf{PKList} \leftarrow \emptyset$

$\mathsf{SKList} \leftarrow \emptyset$

$(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{KGen}()$

$(\mathsf{sk}^\star, \mathsf{pk}^\star, \mathsf{cred}^\star) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}'_{\mathsf{pk}}, \mathcal{O}'_{\mathsf{sk}}}(\mathsf{pp}, \mathsf{pk})$

$\mathsf{sk}' \leftarrow \mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}, \mathsf{cred}^\star)$

$\textbf{return } \Delta.\mathsf{Check}(\mathsf{sk}^\star, \mathsf{pk}^\star)$

$\quad \wedge\ \Delta.\mathsf{Check}(\mathsf{sk}', \mathsf{pk}^\star)$

$\quad \wedge\ [\mathsf{cred}^\star \notin \mathsf{SKList}]$

$\mathcal{O}'_{\mathsf{pk}}(\mathsf{aux})$

$(c, k) \leftarrow\!\!\$\ \Pi.\mathsf{Encaps}(\mathsf{pk}_\Pi)$

$\tau \leftarrow \mathsf{PRF}(k, \mathsf{aux})$

$\mathsf{pk}' \leftarrow \Delta.\mathsf{BlindPK}(\mathsf{pk}_\Delta, \tau)$

$\mathsf{PKList} \leftarrow \mathsf{PKList} \cup \big\{\big(\mathsf{pk}', (c, \mathsf{aux})\big)\big\}$

$\textbf{return } \mathsf{pk}', (c, \mathsf{aux})$

$\mathcal{O}'_{\mathsf{sk}}(c, \mathsf{aux})$

$\textbf{if } (\cdot, (c, \mathsf{aux})) \notin \mathsf{PKList} \textbf{ then return } \perp$

$\mathsf{SKList} \leftarrow \mathsf{SKList} \cup \{(c, \mathsf{aux})\}$

$k \leftarrow \Pi.\mathsf{Decaps}(\mathsf{pk}_\Pi, c)$

$\tau \leftarrow \mathsf{PRF}(k, \mathsf{aux})$

$\textbf{return } \Delta.\mathsf{BlindSK}(\mathsf{sk}_\Delta, \tau)$

Figure 2: The msKS security experiment for pqARKG [Wil23].

party does not wish to reveal the relationship with other keys in an HDK tree which might be used for unrelated purposes; knowing b would enable the subordinate party to unblind the derived public key pk' to reveal the root public seed pk_Δ. Instead, the subordinate party may determine k and aux and compute steps 3–5 of $\mathsf{DerivePK}$ with $\mathsf{pk}_\Delta = \mathsf{pk}^b_\Delta$ and $b = \mathsf{id}_\Delta$, the identity blinding factor, and thus be convinced that pk' was generated from the claimed public seed pk^b_Δ.

Finally, the $\mathsf{DeriveSK}$ function of pqARKG-H also receives the blinding key pk_Δ as a new parameter in order to reconstruct the same PRF output as $\mathsf{DerivePK}$. This pk_Δ parameter may be eliminated in instantiations where pk_Δ can be computed from sk_Δ.

pqARKG-H requires three additional properties of the the key blinding scheme Δ:

1. There exists an *identity blinding factor*, denoted id_Δ, such that

$$\Delta.\mathsf{BlindPK}(\mathsf{pk}, \mathsf{id}_\Delta) = \mathsf{pk} \text{ and } \Delta.\mathsf{BlindSK}(\mathsf{sk}, \mathsf{id}_\Delta) = \mathsf{sk}$$

   for all pk and sk.

2. Δ supports *public key unblinding* in addition to private key unblinding: A function $\Delta.\mathsf{UnblindPK}(\mathsf{pk}, b)$ such that

$$\Delta.\mathsf{UnblindPK}(\Delta.\mathsf{BlindPK}(\mathsf{pk}, b), b) = \mathsf{pk}$$

   for all pk and b.

3. Δ is *commutative* in the blinding factor: for all pk, sk, a and b,

$$\Delta.\mathsf{BlindPK}(\Delta.\mathsf{BlindPK}(\mathsf{pk}, a), b) = \Delta.\mathsf{BlindPK}(\Delta.\mathsf{BlindPK}(\mathsf{pk}, b), a)$$

$$\Delta.\mathsf{BlindSK}(\Delta.\mathsf{BlindSK}(\mathsf{sk}, a), b) = \Delta.\mathsf{BlindSK}(\Delta.\mathsf{BlindSK}(\mathsf{sk}, b), a).$$

For example, any construction based on cyclic groups is likely to satisfy these properties.

pqARKG-H is defined as the suite of procedures in Figure 2. The operator $\|$ denotes binary concatenation, and we assume some well-known encoding is used for pk^b_Δ. Note that if $\Delta.\mathsf{BlindSK}(\mathsf{sk}, b)$ is linear in b with operation $\circ$, then steps 4-5 of $\mathsf{DeriveSK}$ may be optimized as "4. **return** $\Delta.\mathsf{BlindSK}(\mathsf{sk}_\Delta, b \circ \tau)$".

We also modify the msKS security experiment accordingly, resulting in the security experiment $\mathsf{Exp}^{\mathsf{msKS}}_{pqARKG\text{-}H,\mathcal{B}}$. The main difference is that the adversary $\mathcal{B}$ also returns the value $b^\star$ to be used as the b argument to $\mathsf{DeriveSK}$. The public and private key oracles $\mathcal{O}'_{\mathsf{pk}}$ and $\mathcal{O}'_{\mathsf{sk}}$ are also updated to include the b parameter and the additional blinding step. Finally, the check against $\mathcal{B}$ trivially querying $\mathcal{O}'_{\mathsf{sk}}$ for a solution is relaxed to forbid only the exact combination of $b^\star$ and $\mathsf{cred}^\star$ returned as the solution.

Setup: Same as pqARKG.
Check: Same as pqARKG.
KGen: Same as pqARKG.

$\mathsf{DerivePK}(\mathsf{pp}, \mathsf{pk} = (\mathsf{pk}_\Delta, \mathsf{pk}_\Pi), b, \mathsf{aux})$

$(c, k) \leftarrow \Pi.\mathsf{Encaps}(\mathsf{pk}_\Pi)$

$\mathsf{pk}^b_\Delta \leftarrow \Delta.\mathsf{BlindPK}(\mathsf{pk}_\Delta, b)$

$\tau \leftarrow \mathsf{PRF}(k, \mathsf{pk}^b_\Delta \| \mathsf{aux})$

$\mathsf{pk}' \leftarrow \Delta.\mathsf{BlindPK}(\mathsf{pk}^b_\Delta, \tau)$

$\mathbf{return}\ \mathsf{pk}', \mathsf{cred} = (c, \mathsf{aux})$

$\mathsf{DeriveSK}(\mathsf{pp}, \mathsf{pk}_\Delta, \mathsf{sk} = (\mathsf{sk}_\Delta, \mathsf{sk}_\Pi), b, \mathsf{cred} = (c, \mathsf{aux}))$

$k \leftarrow \Pi.\mathsf{Decaps}(\mathsf{sk}_\Pi, c)$

$\mathsf{pk}^b_\Delta \leftarrow \Delta.\mathsf{BlindPK}(\mathsf{pk}_\Delta, b)$

$\tau \leftarrow \mathsf{PRF}(k, \mathsf{pk}^b_\Delta \| \mathsf{aux})$

$\mathsf{sk}^b_\Delta \leftarrow \Delta.\mathsf{BlindSK}(\mathsf{sk}_\Delta, b)$

$\mathbf{return}\ \Delta.\mathsf{BlindSK}(\mathsf{sk}^b_\Delta, \tau)$

Figure 3: Algorithms of the pqARKG-H scheme. The Setup, Check and KGen algorithms are unchanged from pqARKG.

$\mathsf{Exp}^{\mathsf{msKS}}_{pqARKG\text{-}H,\mathcal{B}}(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$

$\mathsf{PKList} \leftarrow \emptyset$

$\mathsf{SKList} \leftarrow \emptyset$

$(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{KGen}()$

$(\mathsf{sk}^\star, \mathsf{pk}^\star, b^\star, \mathsf{cred}^\star) \leftarrow\!\!\$\ \mathcal{B}^{\mathcal{O}'_{\mathsf{pk}}, \mathcal{O}'_{\mathsf{sk}}}(\mathsf{pp}, \mathsf{pk})$

$\mathsf{sk}' \leftarrow \mathsf{DeriveSK}(\mathsf{pp}, \mathsf{pk}_\Delta, \mathsf{sk}, b^\star, \mathsf{cred}^\star)$

$\mathbf{return}\ \Delta.\mathsf{Check}(\mathsf{sk}^\star, \mathsf{pk}^\star)$

$\quad \wedge\ \Delta.\mathsf{Check}(\mathsf{sk}', \mathsf{pk}^\star)$

$\quad \wedge\ [(b^\star, \mathsf{cred}^\star) \notin \mathsf{SKList}]$

$\mathcal{O}'_{\mathsf{pk}}(b, \mathsf{aux})$

$(\mathsf{pk}', \mathsf{cred}) \leftarrow\!\!\$\ \mathsf{DerivePK}(\mathsf{pp}, \mathsf{pk}, b, \mathsf{aux})$

$\mathsf{PKList} \leftarrow \mathsf{PKList} \cup \{(\mathsf{pk}', \mathsf{cred})\}$

$\mathbf{return}\ (\mathsf{pk}', \mathsf{cred})$

$\mathcal{O}'_{\mathsf{sk}}(b, c, \mathsf{aux})$

$\mathbf{if}\ (\cdot, (c, \mathsf{aux})) \notin \mathsf{PKList}\ \mathbf{then\ return}\ \perp$

$\mathsf{SKList} \leftarrow \mathsf{SKList} \cup \{(b, (c, \mathsf{aux}))\}$

$\mathbf{return}\ \mathsf{DeriveSK}(\mathsf{pp}, \mathsf{pk}_\Delta, \mathsf{sk}, b, (c, \mathsf{aux}))$

Figure 4: The msKS security experiment for pqARKG-H.

$\mathcal{A}^{\mathcal{O}'_{\mathsf{pk}},\mathcal{O}'_{\mathsf{sk}}}(\mathsf{pp} = (\Delta, \Pi, \mathsf{PRF}), \mathsf{pk} = (\mathsf{pk}_\Delta, \mathsf{pk}_\Pi))$

$$\begin{array}{l}
(\overline{\mathsf{sk}^\star}, \overline{\mathsf{pk}^\star}, \overline{b^\star}, (\overline{c^\star}, \overline{\mathsf{aux}^\star})) \leftarrow\!\!\$\; \mathcal{B}^{\overline{\mathcal{O}'_{\mathsf{pk}}},\overline{\mathcal{O}'_{\mathsf{sk}}}}(\mathsf{pp}, \mathsf{pk}) \\
\mathsf{sk}^\star \leftarrow \Delta.\mathsf{UnblindSK}(\overline{\mathsf{sk}^\star}, \overline{b^\star}) \\
\mathsf{pk}^\star \leftarrow \Delta.\mathsf{UnblindPK}(\overline{\mathsf{pk}^\star}, \overline{b^\star}) \\
\mathsf{pk}_\Delta^{b^\star} \leftarrow \Delta.\mathsf{BlindPK}(\mathsf{pk}_\Delta, \overline{b^\star}) \\
\textbf{return } (\mathsf{sk}^\star, \mathsf{pk}^\star, (\overline{c^\star}, \mathsf{pk}_\Delta^{b^\star} \,\|\, \overline{\mathsf{aux}^\star}))
\end{array}$$

$\overline{\mathcal{O}'_{\mathsf{pk}}}(b, \mathsf{aux})$

$$\begin{array}{l}
\mathsf{pk}_\Delta^b \leftarrow \Delta.\mathsf{BlindPK}(\mathsf{pk}_\Delta, b) \\
(\mathsf{pk}', (c, \cdot)) \leftarrow\!\!\$\; \mathcal{O}'_{\mathsf{pk}}(\mathsf{pk}_\Delta^b \,\|\, \mathsf{aux}) \\
\overline{\mathsf{pk}'} \leftarrow \Delta.\mathsf{BlindPK}(\mathsf{pk}', b) \\
\textbf{return } (\overline{\mathsf{pk}'}, (c, \mathsf{aux}))
\end{array}$$

$\overline{\mathcal{O}'_{\mathsf{sk}}}(b, c, \mathsf{aux})$

$$\begin{array}{l}
\mathsf{pk}_\Delta^b \leftarrow \Delta.\mathsf{BlindPK}(\mathsf{pk}_\Delta, b) \\
\mathsf{sk}' \leftarrow \mathcal{O}'_{\mathsf{sk}}(c, \mathsf{pk}_\Delta^b \,\|\, \mathsf{aux}) \\
\textbf{return } \Delta.\mathsf{BlindSK}(\mathsf{sk}', b)
\end{array}$$

Figure 5: Reduction of $\mathsf{Exp}^{\mathsf{msKS}}_{pqARKG,\mathcal{A}}$ to $\mathsf{Exp}^{\mathsf{msKS}}_{pqARKG\text{-}H,\mathcal{B}}$.

4 Reduction of pqARKG to pqARKG-H in msKS security experiment

We now show that pqARKG-H can satisfy malicious strong key security msKS. The proof requires two additional properties of some key blinding schemes that are defined in [Wil23]: unique blinding and private-key unblinding. Recall from [Wil23] that pqARKG satisfies msKS when instantiated with a key blinding scheme that provides these properties. Therefore a reduction proof from pqARKG-H to pqARKG suffices.

Theorem 4.1. *Let pqARKG-H be the ARKG construction described in Figure 3, instantiated with a key blinding scheme Δ that provides unique blinding and supports private-key unblinding. For any efficient adversary $\mathcal{B}$, there exists an efficient algorithm $\mathcal{A}$ such that*

$$\mathsf{Adv}^{\mathsf{msKS}}_{pqARKG\text{-}H,\mathcal{B}}(\lambda) = \mathsf{Adv}^{\mathsf{msKS}}_{pqARKG,\mathcal{A}}(\lambda)$$

where the security experiment msKS *is defined in Figure 3 for pqARKG-H and in Figure 2 for pqARKG.*

Proof. Given an adversary $\mathcal{B}$ that defeats $\mathsf{Exp}^{\mathsf{msKS}}_{pqARKG\text{-}H,\mathcal{B}}$, we construct an adversary $\mathcal{A}$ that defeats $\mathsf{Exp}^{\mathsf{msKS}}_{pqARKG,\mathcal{A}}$.

When invoked, this adversary $\mathcal{A}$ simply invokes the given adversary $\mathcal{B}$ with its own challenge. The $\mathcal{O}'_{\mathsf{pk}}$ and $\mathcal{O}'_{\mathsf{sk}}$ oracles are adapted for $\mathcal{B}$ by adding the pk_Δ^b prefix to aux and performing the additional blinding with the b argument, thus $\mathcal{A}$ faithfully simulates $\mathsf{Exp}^{\mathsf{msKS}}_{pqARKG\text{-}H,\mathcal{B}}$ for $\mathcal{B}$. The $\mathsf{aux}^\star$ returned to the challenger is adapted with the $\mathsf{pk}_\Delta^{b^\star}$ prefix computed using the blinding factor $\overline{b^\star} = b^\star$ returned by $\mathcal{B}$, thus pqARKG-H produces the same τ on line 3 of its $\mathsf{DerivePK}$ and $\mathsf{DeriveSK}$ functions as pqARKG does on line 2 of its $\mathsf{DerivePK}$ and $\mathsf{DeriveSK}$ functions.

To prove that $\mathcal{A}$ wins its game precisely when $\mathcal{B}$ wins its game, we observe that $\mathcal{A}$ passes each of the three conditions on lines 7–9 of $\mathsf{Exp}^{\mathsf{msKS}}_{pqARKG,\mathcal{A}}$ if and only if $\mathcal{B}$ passes the corresponding condition from the ones on lines 7–9 of $\mathsf{Exp}^{\mathsf{msKS}}_{pqARKG\text{-}H,\mathcal{B}}$.

For the first condition this holds because by definition of $\mathsf{sk}^\star$ and $\mathsf{pk}^\star$, and applying the unique blinding and private-key unblinding properties,

$$\begin{aligned}\Delta.\mathsf{Check}(\mathsf{sk}^\star, \mathsf{pk}^\star) &= \Delta.\mathsf{Check}(\Delta.\mathsf{UnblindSK}(\overline{\mathsf{sk}^\star}, \overline{b^\star}), \Delta.\mathsf{UnblindPK}(\overline{\mathsf{pk}^\star}, \overline{b^\star}))\\ &= \Delta.\mathsf{Check}(\overline{\mathsf{sk}^\star}, \overline{\mathsf{pk}^\star}).\end{aligned}$$

For the second condition this holds because of the following argument. First, observe that $c^\star = \overline{c^\star}$ and $\mathsf{aux}^\star = \mathsf{pk}_\Delta^{\overline{b^\star}} \,\|\, \overline{\mathsf{aux}^\star}$, and therefore

$$\tau = \mathsf{PRF}(\Pi.\mathsf{Decaps}(\mathsf{sk}_\Pi, c^\star), \mathsf{aux}^\star) = \mathsf{PRF}(\Pi.\mathsf{Decaps}(\mathsf{sk}_\Pi, \overline{c^\star}), \mathsf{pk}_\Delta^{\overline{b^\star}} \,\|\, \overline{\mathsf{aux}^\star})$$

so $\mathsf{DeriveSK}$ on line 6 of $\mathsf{Exp}^{\mathsf{msKS}}_{pqARKG,\mathcal{A}}$ computes the same τ as on line 6 of $\mathsf{Exp}^{\mathsf{msKS}}_{pqARKG\text{-}H,\mathcal{B}}$. The above gives

$$\begin{aligned}\mathsf{sk}' &= \mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}, \mathsf{cred}^\star)\\ &= \Delta.\mathsf{BlindSK}(\mathsf{sk}_\Delta, \mathsf{PRF}(\Pi.\mathsf{Decaps}(\mathsf{sk}_\Pi, c^\star), \mathsf{aux}^\star))\\ &= \Delta.\mathsf{BlindSK}(\mathsf{sk}_\Delta, \tau)\end{aligned}$$

where $\mathsf{cred}^\star = (c^\star, \mathsf{aux}^\star)$. Let $\overline{\mathsf{sk}'}$ be an alias of the target symbol on line 6 of $\mathsf{Exp}^{\mathsf{msKS}}_{pqARKG\text{-}H,\mathcal{B}}$. Then

$$\begin{aligned}\overline{\mathsf{sk}'} &= \mathsf{DeriveSK}(\mathsf{pp}, \mathsf{pk}_\Delta, \mathsf{sk}, \overline{b^\star}, \overline{\mathsf{cred}^\star})\\ &= \Delta.\mathsf{BlindSK}(\Delta.\mathsf{BlindSK}(\mathsf{sk}_\Delta, \overline{b^\star}), \tau)\\ &= \Delta.\mathsf{BlindSK}(\Delta.\mathsf{BlindSK}(\mathsf{sk}_\Delta, \tau), \overline{b^\star})\\ &= \Delta.\mathsf{BlindSK}(\mathsf{sk}', \overline{b^\star})\end{aligned}$$

and therefore

$$\mathsf{sk}' = \Delta.\mathsf{UnblindSK}(\overline{\mathsf{sk}'}, \overline{b^\star})$$

so by the definitions of unblinding and unique blinding, the second condition in $\mathsf{Exp}^{\mathsf{msKS}}_{pqARKG\text{-}H,\mathcal{B}}$ is equivalent to

$$\begin{aligned}\Delta.\mathsf{Check}(\overline{\mathsf{sk}'}, \overline{\mathsf{pk}^\star}) &= \Delta.\mathsf{Check}(\overline{\mathsf{sk}'}, \Delta.\mathsf{BlindPK}(\Delta.\mathsf{UnblindPK}(\overline{\mathsf{pk}^\star}, \overline{b^\star}), \overline{b^\star}))\\ &= \Delta.\mathsf{Check}(\overline{\mathsf{sk}'}, \Delta.\mathsf{BlindPK}(\mathsf{pk}^\star, \overline{b^\star}))\\ &= \Delta.\mathsf{Check}(\Delta.\mathsf{BlindSK}(\mathsf{sk}', \overline{b^\star}), \Delta.\mathsf{BlindPK}(\mathsf{pk}^\star, \overline{b^\star}))\\ &= \Delta.\mathsf{Check}(\mathsf{sk}', \mathsf{pk}^\star).\end{aligned}$$

For the third condition this holds because $\mathcal{A}$ only appends to SKList when invoking $\mathcal{O}'_{\mathsf{sk}}$, which it does only when $\mathcal{B}$ invokes $\overline{\mathcal{O}'_{\mathsf{sk}}}$. Therefore if $\mathcal{B}$ does not invoke $\overline{\mathcal{O}'_{\mathsf{sk}}}$ with arguments $(b^\star, c^\star, \mathsf{aux}^\star)$, then $\mathcal{A}$ also does not invoke $\mathcal{O}'_{\mathsf{sk}}$ with arguments $(c^\star, \mathsf{aux}^\star)$. This is the case when $\mathcal{B}$ wins its game, therefore $\mathcal{A}$ passes the condition $(c^\star, \mathsf{aux}^\star) \notin \mathsf{SKList}$ whenever $\mathcal{B}$ does.

□

5 Acknowledgements

We would like to thank Sander Dijkhuis for proposing the construction with mixing pk^b_Δ into the PRF instead of b, and for helping find a successful approach for the security reduction.

References

[Bal+25] Dirk Balfanz et al. *Web Authentication: An API for accessing Public Key Credentials*. 2025. URL: `https://www.w3.org/TR/2025/WD-webauthn-3-20250127/`. Level 3 working draft.

[Dij25] Sander Dijkhuis. *Hierarchical Deterministic Keys*. 2025. URL: `https://www.ietf.org/archive/id/draft-dijkhuis-cfrg-hdkeys-06.html`. Version 06.

[Fry+20] Nick Frymann et al. "Asynchronous Remote Key Generation: An Analysis of Yubico's Proposal for W3C WebAuthn". In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. CCS '20. Virtual Event, USA: Association for Computing Machinery, 2020, pp. 939–954. ISBN: 9781450370899. DOI: `10.1145/3372297.3417292`. URL: `https://doi.org/10.1145/3372297.3417292`.

[LB25] Emil Lundberg and John Bradley. *The Asynchronous Remote Key Generation (ARKG) algorithm*. 2025. URL: `https://www.ietf.org/archive/id/draft-bradleylundberg-cfrg-arkg-03.html`. Version 03.

[Lun24] Emil Lundberg. *Add "sign" extension*. 2024. URL: `https://github.com/w3c/webauthn/pull/2078`. W3C Web Authentication issue tracker, accessed 2025-01-31.

[Wil23] Spencer MacLaren Wilson. *Post-Quantum Account Recovery for Passwordless Authentication*. 2023. URL: `https://uwspace.uwaterloo.ca/items/d1f73f71-e3b2-438c-b261-11632becdbb2`.