pqARKG-H: An extension of pqARKG for Hierarchical Deterministic Keys

Emil Lundberg
Yubico AB
emil@yubico.com

February 3, 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 element (SE) and new public keys can be generated without invoking the SE. However, when ARKG is used in HDK, the original ARKG procedures would require one SE invocation per HDK layer when exercising derived private keys. This is undesirable in contexts where each SE invocation would require a distinct user gesture.

To address this we analyze the security of an ARKG extension pqARKG-H that enables any number of HDK layers with a single SE invocation. We base pqARKG-H on the ARKG generalization pqARKG, and demonstrate that it retains msKS security by a straightforward reduction of the 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 element (SE) and new public keys can be generated without invoking the SE. However, when ARKG is used in HDK, the original ARKG procedures would require one SE invocation of DeriveSK per HDK layer when exercising derived private keys. This is undesirable in contexts where each SE 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 sign-

ing 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 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 DerivePK and DeriveSK functions. This b is an additional blinding factor in the key blinding scheme Δ, allowing the ARKG subordinate party (the party generating public keys) to add any number of additional blinding layers on top of the one performed by the ARKG delegating party (the party holding the ARKG private seed). To prevent choosing $b = -\tau$ so that it cancels the blinding factor τ computed in step 2 of 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 extraction of the private seed sk by a malicious choice of b even if a derived private key sk' is leaked.

The new argument mixed into the PRF is however not b directly, but a blinded public key pk^b_Δ incorporating the blinding factor b. This enables the subordinate party to prove to a third party that a public key was generated using pqARKG-H without having to disclose b. This is desirable if the subordinate party does not wish to reveal the relationship with keys from other branches of an HDK tree which might be used for unrelated purposes;

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

$\mathsf{Setup}(\lambda)$

1 : **return** $\mathsf{pp} = (\Delta, \Pi, \mathsf{PRF})$

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

1 : **return** $\Delta.\mathsf{Check}(\mathsf{sk}', \mathsf{pk}')$

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

1 : $(\mathsf{sk}_\Delta, \mathsf{pk}_\Delta) \leftarrow \Delta.\mathsf{KeyGen}()$
2 : $(\mathsf{sk}_\Pi, \mathsf{pk}_\Pi) \leftarrow \Pi.\mathsf{KeyGen}()$
3 : **return** $\mathsf{sk} = (\mathsf{sk}_\Delta, \mathsf{sk}_\Pi), \mathsf{pk} = (\mathsf{pk}_\Delta, \mathsf{pk}_\Pi)$

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

1 : $(c, k) \leftarrow \Pi.\mathsf{Encaps}(\mathsf{pk}_\Pi)$
2 : $\tau \leftarrow \mathsf{PRF}(k, \mathsf{aux})$
3 : $\mathsf{pk}' \leftarrow \Delta.\mathsf{BlindPK}(\mathsf{pk}_\Delta, \tau)$
4 : **return** $\mathsf{pk}', \mathsf{cred} = (c, \mathsf{aux})$

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

1 : $k \leftarrow \Pi.\mathsf{Decaps}(\mathsf{sk}_\Pi, c)$
2 : $\tau \leftarrow \mathsf{PRF}(k, \mathsf{aux})$
3 : **return** $\Delta.\mathsf{BlindSK}(\mathsf{sk}_\Delta, \tau)$

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

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

1 : $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
2 : $\mathsf{PKList} \leftarrow \emptyset$
3 : $\mathsf{SKList} \leftarrow \emptyset$
4 : $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{KeyGen}()$
5 : $(\mathsf{sk}^\star, \mathsf{pk}^\star, \mathsf{cred}^\star) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}_{\mathsf{pk}'}, \mathcal{O}_{\mathsf{sk}'}}(\mathsf{pp}, \mathsf{pk})$
6 : $\mathsf{sk}' \leftarrow \mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}, \mathsf{cred}^\star)$
7 : **return** $\mathsf{Check}(\mathsf{pp}, \mathsf{sk}^\star, \mathsf{pk}^\star)$
8 : $\qquad \wedge\ \mathsf{Check}(\mathsf{pp}, \mathsf{sk}', \mathsf{pk}^\star)$
9 : $\qquad \wedge\ [\mathsf{cred}^\star \notin \mathsf{SKList}]$

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

1 : $(c, k) \leftarrow\!\!\$\ \Pi.\mathsf{Encaps}(\mathsf{pk}_\Pi)$
2 : $\tau \leftarrow \mathsf{PRF}(k, \mathsf{aux})$
3 : $\mathsf{pk}' \leftarrow \Delta.\mathsf{BlindPK}(\mathsf{pk}_\Delta, \tau)$
4 : $\mathsf{PKList} \leftarrow \mathsf{PKList} \cup \{(\mathsf{pk}', (c, \mathsf{aux}))\}$
5 : **return** $\mathsf{pk}', (c, \mathsf{aux})$

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

1 : **if** $(\cdot, (c, \mathsf{aux})) \notin \mathsf{PKList}$ **then return** $\perp$
2 : $\mathsf{SKList} \leftarrow \mathsf{SKList} \cup \{(c, \mathsf{aux})\}$
3 : $k \leftarrow \Pi.\mathsf{Decaps}(\mathsf{sk}_\Pi, c)$
4 : $\tau \leftarrow \mathsf{PRF}(k, \mathsf{aux})$
5 : **return** $\Delta.\mathsf{BlindSK}(\mathsf{sk}_\Delta, \tau)$

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

knowing b would enable the third party to unblind the derived public key pk' to reveal the root public seed pk_Δ. Instead, the third party may receive k and aux and recompute steps 3-5 of $\mathsf{DerivePK}$ with $\mathsf{pk}_\Delta = \mathsf{pk}_\Delta^b$ and $b = 1$, the identity blinding factor, and thus be convinced that pk' was generated from the claimed public seed pk_Δ^b.

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 1, such that

$$\Delta.\mathsf{BlindPK}(\mathsf{pk}, 1) = \mathsf{pk} \text{ and } \Delta.\mathsf{BlindSK}(\mathsf{sk}, 1) = \mathsf{sk}$$

   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_Δ^b. Note that if $\Delta.\mathsf{BlindSK}(\mathsf{sk}, b)$ is linear in b, then steps 4-5 of $\mathsf{DeriveSK}$ may be optimized as "4. **return** $\Delta.\mathsf{BlindSK}(\mathsf{sk}_\Delta, b\tau)$".

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.
KeyGen: Same as pqARKG.

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

1 : $(c, k) \leftarrow \Pi.\mathsf{Encaps}(\mathsf{pk}_\Pi)$
2 : $\mathsf{pk}_\Delta^b \leftarrow \Delta.\mathsf{BlindPK}(\mathsf{pk}_\Delta, b)$
3 : $\tau \leftarrow \mathsf{PRF}(k, \mathsf{pk}_\Delta^b \| \mathsf{aux})$
4 : $\mathsf{pk}' \leftarrow \Delta.\mathsf{BlindPK}(\mathsf{pk}_\Delta^b, \tau)$
5 : **return** $\mathsf{pk}', \mathsf{cred} = (c, \mathsf{aux})$

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

1 : $k \leftarrow \Pi.\mathsf{Decaps}(\mathsf{sk}_\Pi, c)$
2 : $\mathsf{pk}_\Delta^b \leftarrow \Delta.\mathsf{BlindPK}(\mathsf{pk}_\Delta, b)$
3 : $\tau \leftarrow \mathsf{PRF}(k, \mathsf{pk}_\Delta^b \| \mathsf{aux})$
4 : $\mathsf{sk}_\Delta^b \leftarrow \Delta.\mathsf{BlindSK}(\mathsf{sk}_\Delta, b)$
5 : **return** $\Delta.\mathsf{BlindSK}(\mathsf{sk}_\Delta^b, \tau)$

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

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

1 : $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
2 : $\mathsf{PKList} \leftarrow \emptyset$
3 : $\mathsf{SKList} \leftarrow \emptyset$
4 : $(\mathsf{sk}, \mathsf{pk}) \leftarrow\$ \mathsf{KeyGen}()$
5 : $(\mathsf{sk}^\star, \mathsf{pk}^\star, b^\star, \mathsf{cred}^\star) \leftarrow\$ \mathcal{B}^{\mathcal{O}_{\mathsf{pk}'}, \mathcal{O}_{\mathsf{sk}'}}(\mathsf{pp}, \mathsf{pk})$
6 : $\mathsf{sk}' \leftarrow \mathsf{DeriveSK}(\mathsf{pp}, \mathsf{pk}_\Delta, \mathsf{sk}, b^\star, \mathsf{cred}^\star)$
7 : **return** $\mathsf{Check}(\mathsf{pp}, \mathsf{sk}^\star, \mathsf{pk}^\star)$
8 : $\quad \wedge \mathsf{Check}(\mathsf{pp}, \mathsf{sk}', \mathsf{pk}^\star)$
9 : $\quad \wedge [(b^\star, \mathsf{cred}^\star) \notin \mathsf{SKList}]$

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

1 : $(\mathsf{pk}', \mathsf{cred}) \leftarrow\$ \mathsf{DerivePK}(\mathsf{pp}, \mathsf{pk}, b, \mathsf{aux})$
2 : $\mathsf{PKList} \leftarrow \mathsf{PKList} \cup \{(\mathsf{pk}', \mathsf{cred})\}$
3 : **return** $\mathsf{pk}', \mathsf{cred}$

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

1 : **if** $(\cdot, (c, \mathsf{aux})) \notin \mathsf{PKList}$ **then return** $\perp$
2 : $\mathsf{SKList} \leftarrow \mathsf{SKList} \cup \{(c, \mathsf{aux})\}$
3 : **return** $\mathsf{DeriveSK}(\mathsf{pp}, \mathsf{pk}_\Delta, \mathsf{sk}, b, (c, \mathsf{aux}))$

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

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

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

2 : $\mathsf{sk}^\star \leftarrow \Delta.\mathsf{UnblindSK}(\overline{\mathsf{sk}^\star},\overline{b^\star})$

3 : $\mathsf{pk}^\star \leftarrow \Delta.\mathsf{UnblindPK}(\overline{\mathsf{pk}^\star},\overline{b^\star})$

4 : $\mathsf{pk}_\Delta^{b^\star} \leftarrow \Delta.\mathsf{BlindPK}(\mathsf{pk}_\Delta,\overline{b^\star})$

5 : **return** $(\mathsf{sk}^\star,\mathsf{pk}^\star,(\overline{c^\star},\mathsf{pk}_\Delta^{b^\star}\|\,\overline{\mathsf{aux}^\star}))$

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

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

2 : $(\mathsf{pk}',(c,\cdot)) \leftarrow\$\ \mathcal{O}_{\mathsf{pk}'}(\mathsf{pk}_\Delta^b\|\,\mathsf{aux})$

3 : $\overline{\mathsf{pk}'} \leftarrow \Delta.\mathsf{BlindPK}(\mathsf{pk}',b)$

4 : **return** $(\overline{\mathsf{pk}'},(c,\mathsf{aux}))$

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

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

2 : $\mathsf{sk}' \leftarrow \mathcal{O}_{\mathsf{sk}'}(c,\mathsf{pk}_\Delta^b\|\,\mathsf{aux})$

3 : **return** $\Delta.\mathsf{BlindSK}(\mathsf{sk}',b)$

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 is msKS secure by showing that an adversary $\mathcal{B}$ that defeats $\mathsf{Exp}^{\mathsf{msKS}}_{pqARKG\text{-}H,\mathcal{B}}$ also defeats $\mathsf{Exp}^{\mathsf{msKS}}_{pqARKG,\mathcal{A}}$. Given such an adversary $\mathcal{B}$, we construct an adversary $\mathcal{A}$ that defeats $\mathsf{Exp}^{\mathsf{msKS}}_{pqARKG,\mathcal{A}}$ as defined in Figure 4.

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 DerivePK and DeriveSK functions as pqARKG does on line 2 of its DerivePK and DeriveSK functions.

We see that $\mathcal{A}$ passes the conditions on lines 7-9 of $\mathsf{Exp}^{\mathsf{msKS}}_{pqARKG,\mathcal{A}}$:

- $\mathsf{Check}(\mathsf{pp},\mathsf{sk}^\star,\mathsf{pk}^\star)$ succeeds because:

  $\mathsf{sk}^\star = \Delta.\mathsf{UnblindSK}(\overline{\mathsf{sk}^\star},\overline{b^\star})$ and $\mathsf{pk}^\star = \Delta.\mathsf{UnblindPK}(\overline{\mathsf{pk}^\star},\overline{b^\star})$. Therefore by the definitions of unblinding and unique blinding, $\mathcal{A}$ passes its $\mathsf{Check}(\mathsf{pp},\mathsf{sk}^\star,\mathsf{pk}^\star)$ condition precisely when $\mathcal{B}$ does.

- $\mathsf{Check}(\mathsf{pp},\mathsf{sk}',\mathsf{pk}^\star)$ succeeds because:

  First, observe that $c^\star = \overline{c^\star}$ and $\mathsf{aux}^\star = \mathsf{pk}_\Delta^{b^\star}\|\,\overline{\mathsf{aux}^\star}$, and therefore

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

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 sk' on line 6 of $\mathsf{Exp}^{\mathsf{msKS}}_{pqARKG\text{-}H,\mathcal{B}}$. Then

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

so by the definitions of unblinding and unique blinding, the $\mathsf{Check}(\mathsf{pp}, \mathsf{sk}', \mathsf{pk}^\star)$ condition in $\mathsf{Exp}^{\mathsf{msKS}}_{pqARKG\text{-}H,\mathcal{B}}$ is equivalent to

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

which is precisely the condition on line 8 of $\mathsf{Exp}^{\mathsf{msKS}}_{pqARKG,\mathcal{A}}$. Therefore $\mathcal{A}$ passes its $\mathsf{Check}(\mathsf{pp}, \mathsf{sk}', \mathsf{pk}^\star)$ condition precisely when $\mathcal{B}$ does.

- $(c^\star, \mathsf{aux}^\star) \notin \mathsf{SKList}$ succeeds because:

  SKList is appended to only when $\mathcal{A}$ invokes $\mathcal{O}_{\mathsf{sk}'}$, 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 this condition when $\mathcal{B}$ does.

In conclusion, we see that $\mathcal{A}$ wins its game precisely when $\mathcal{B}$ wins its game, therefore the advantages are equal:

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

Thus we conclude that pqARKG-H retains the msKS property of pqARKG.

5 Acknowledgements

We would like to thank Sander Dijkhuis for proposing the construction with mixing pk_Δ^b 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.