A Note on Unforgeability of MuSig2 with Key Tweaking

No Author Given

No Institute Given

Abstract. *Key Tweaking* refers to the process of producing a new pair of secret and public key from a given keypair. This is used, for example, to derive fresh keys from a master keypair or to create a commitment to a value such that the commitment is also a public key. In this note, we demonstrate that a variant of MuSig2 with naive support for key tweaking is insecure and show an alternative method of tweaking that is not vulnerable to the attack.

1 The Vulnerable Scheme

Figure 1 shows MuSig2NaiveTweak, a multi-signature scheme identical to MuSig2 [NRS21] except that it additionally allows signing for a tweaked public key. There are multiple variants of tweaking, which differ mainly in how the tweak t is derived. We chose a variant of tweaking for MuSig2NaiveTweak that gives the adversary the minimal capability necessary to execute the attack. In particular, the honest MuSig2NaiveTweak signer has an algorithm KeyTweak that generates public, uniformly random tweaks without input from the adversary. The signer accepts an externally provided tweak when signing but only if it has been output by KeyTweak.

MuSig2NaiveTweak uses *additive* tweaking, but the attack similarly applies to *multiplicative* tweaking.

2 Generalized Birthday Problem

The attack against MuSig2NaiveTweak described below makes use of Wagner's algorithm [Wag02] for solving the Generalized Birthday Problem. It can be defined as follows for the purpose of this note: Given a constant value $t \in \mathbb{Z}_p$, an integer $k_{\max}$, and access to random oracle H mapping onto $\mathbb{Z}_p$, find a set $\{q_1, \ldots, q_{k_{\max}}\}$ of $k_{\max}$ queries such that $\sum_{k=1}^{k_{\max}} H(q_k) = t$. For $k_{\max} = 2^{\sqrt{\log_2(p)}-1}$ the complexity of this algorithm is $O(2^{2\sqrt{\log_2(p)}})$.

3 Description of the Attack against MuSig2NaiveTweak

The adversary calls KeyTweak $\ell_{\max} \in O(2^{2\sqrt{\log_2(p)}})$ times to obtain values $t^{(1)}, \ldots, t^{(\ell_{\max})}$ and computes the multiset of public keys L and aggregate key $\widetilde{X}$ for the (untweaked) public key of the honest signer $X_1' = g^{x_1}$ as

$$\begin{aligned} L &= \{X_1' g^{t^{(1)}}, \ldots, X_1' g^{t^{(\ell_{\max})}}\} \\ \widetilde{X} &= \mathsf{KeyAgg}(L). \end{aligned}$$

Then, the adversary opens $k_{\max} = 2^{\sqrt{\log_2(p)}-1}$ concurrent signing sessions by requesting $k_{\max}$ nonce tuples $R_{1,1}^{(1)}, \ldots, R_{1,\nu}^{(1)}, \ldots, R_{1,1}^{(k_{\max})}, \ldots, R_{1,\nu}^{(k_{\max})}$ from the honest signer and computes

$$\begin{aligned} R_j &= \sum_{k=1}^{k_{\max}} R_{1,j}^{(k)}, \quad j \in [1, \nu] \\ b &= \mathsf{H}_{\mathrm{non}}(\widetilde{X}, (R_1, \ldots, R_\nu), m) \\ R^* &= \prod_{j=1}^{\nu} R_j^{b^{j-1}}. \end{aligned}$$

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

$(\mathbb{G}, p, g) \leftarrow \mathsf{GrGen}(1^\lambda)$
Select three hash functions
$\mathsf{H}_{\text{agg}}, \mathsf{H}_{\text{non}}, \mathsf{H}_{\text{sig}} : \{0,1\}^* \to \mathbb{Z}_p$
$par := ((\mathbb{G}, p, g), \mathsf{H}_{\text{agg}}, \mathsf{H}_{\text{non}}, \mathsf{H}_{\text{sig}})$
return par

$\mathsf{KeyGen}()$

$x \leftarrow\!\!\$\ \mathbb{Z}_p;\ X := g^x$
$sk := x;\ pk := X$
return (sk, pk)

$\mathsf{KeyTweak}()$

$\mathbf{t} \leftarrow\!\!\$\ \mathbb{Z}_\mathbf{p}$
return t

$\mathsf{KeyAggCoef}(L, X_i)$

return $\mathsf{H}_{\text{agg}}(L, X_i)$

$\mathsf{KeyAgg}(L)$

$\{X_1, \dots, X_n\} := L$
for $i := 1 \dots n$ **do**
$a_i := \mathsf{KeyAggCoef}(L, X_i)$
return $\widetilde{X} := \prod_{i=1}^n X_i^{a_i}$

$\mathsf{Ver}(\widetilde{pk}, m, \sigma)$

$\widetilde{X} := \widetilde{pk};\ (R, s) := \sigma$
$c := \mathsf{H}_{\text{sig}}(\widetilde{X}, R, m)$
return $(g^s = R\widetilde{X}^c)$

$\mathsf{Sign}()$

// Local signer has index 1.
for $j := 1 \dots \nu$ **do**
$r_{1,j} \leftarrow\!\!\$\ \mathbb{Z}_p;\ R_{1,j} := g^{r_{1,j}}$
$out_1 := (R_{1,1}, \dots, R_{1,\nu})$
$state_1 := (r_{1,1}, \dots, r_{1,\nu})$
return $(out_1, state_1)$

$\mathsf{SignAgg}(out_1, \dots, out_n)$

for $i := 1 \dots n$ **do**
$(R_{i,1}, \dots, R_{i,\nu}) := out_i$
for $j := 1 \dots \nu$ **do**
$R_j := \prod_{i=1}^n R_{i,j}$
return $out := (R_1, \dots, R_\nu)$

$\mathsf{Sign}'(state_1, out, sk_1, m, (pk_2, \dots, pk_n), \mathbf{t})$

// Sign' must be called at most once per $state_1$.
if $\mathbf{t} \neq \mathbf{0}$ and has not been output by $\mathsf{KeyTweak}$
then return `false`
$(r_{1,1}, \dots, r_{1,\nu}) := state_1$
$x_1 := sk_1 + \mathbf{t} \bmod \mathbf{p};\ X_1 := g^{x_1}$
$(X_2, \dots, X_n) := (pk_2, \dots, pk_n)$
$L := \{X_1, \dots, X_n\}$
$a_1 := \mathsf{KeyAggCoef}(L, X_1)$
$\widetilde{X} := \mathsf{KeyAgg}(L)$
$(R_1, \dots, R_\nu) := out$
$b := \mathsf{H}_{\text{non}}(\widetilde{X}, (R_1, \dots, R_\nu), m)$
$R := \prod_{j=1}^\nu R_j^{b^{j-1}}$
$c := \mathsf{H}_{\text{sig}}(\widetilde{X}, R, m)$
$s_1 := ca_1x_1 + \sum_{i=1}^\nu r_{1,j}b^{j-1} \bmod p$
$state'_1 := R;\ out'_1 := s_1$
return $(state'_1, out'_1)$

$\mathsf{SignAgg}'(out'_1, \dots, out'_n)$

$(s_1, \dots, s_n) := (out'_1, \dots, out'_n)$
$s := \sum_{i=1}^n s_i \bmod p$
return $out' := s$

$\mathsf{Sign}''(state'_1, out')$

$R := state'_1;\ s := out'$
return $\sigma := (R, s)$

Fig. 1. The multi-signature scheme $\mathsf{MuSig2NaiveTweak}[\mathsf{GrGen}, \nu]$. The differences to $\mathsf{MuSig2}[\mathsf{GrGen}, \nu]$ are displayed in **red**.

Now it is possible to use Wagner's algorithm to find a function $f : [1, k_{\max}] \to [1, \ell_{\max}]$ that associates a tweak $t^{(\ell)}$ to each session k such that

$$\sum_{k=1}^{k_{\max}} \underbrace{\mathsf{H}_{\mathrm{agg}}(L, X_1' g^{t^{(f(k))}})}_{=:\, a_1^{(k)}} \underbrace{\mathsf{H}_{\mathrm{sig}}(\widetilde{X}, R^*, m)}_{=:\, c^{(k)}} = \underbrace{\mathsf{H}_{\mathrm{sig}}(X_1', R^*, m^*)}_{=:\, c^*} . \tag{1}$$

for a forgery target message m^*. For all $k \in [1, k_{\max}]$, the adversary asks the honest signer for a partial signature using value $t^{f(k)}$ which is answered with $s_1^{(k)} = r_{1,1}^{(k)} + br_{1,2}^{(k)} + c^{(k)} \cdot a_1^{(k)}(x_1 + t^{(f(k))})$. This allows the adversary to compute

$$\begin{aligned} s_1^{*\prime} &= \sum_{k=1}^{k_{\max}} s_1^{(k)} && (2)\\ &= \sum_{k=1}^{k_{\max}} r_{1,1}^{(k)} b r_{1,2}^{(k)} + \left(\sum_{k=1}^{k_{\max}} c^{(k)} a_1^{(k)} \right) \cdot x_1 + \sum_{k=1}^{k_{\max}} c^{(k)} a_1^{(k)} t^{(f(k))} && (3)\\ &= \log_g(R^*) + c^* x_1 + \sum_{k=1}^{k_{\max}} c^{(k)} a_1^{(k)} t^{(f(k))} && (4) \end{aligned}$$

where the last equality follows from Equation (1). The adversary can subtract the last summand

$$s_1^* = s_1^{*\prime} - \sum_{k=1}^{k_{\max}} c^{(k)} a_1^{(k)} t^{(f(k))}$$

to obtain (R^*, s^*), a valid forgery on message m^* for public key X_1'.

4 BLLOR Attack

Benhamouda, Lepoint, Loss, Orrù, and Raykova [BLL+21] describe an algorithm that solves the ROS problem and can be applied to attack to break the unforgeability of MuSig2NaiveTweak. If the adversary can open at least $\log_2 p$ sessions, then the algorithm has complexity $O(\log_2^2 p)$ and a negligible running time in practice (otherwise, a variant of the algorithm can be applied that has a higher complexity). In contrast to the attack based on Wagner's algorithm, this attack allows using multisets of public keys with only two elements.

5 Where the Security Proof of MuSig2 Fails when Adding Naive Tweaking

Jonas' note: TODO: In the ROM proof we would have two exections where $a \neq a'$ but $b = b'$.

6 A Fixed MuSig2 Variant with Tweaking

The attack in section 3 relies on selecting a tweak $t^{(k)}$ for a signing session k *after* obtaining nonces $R_{1,1}^{(k)}, \ldots, R_{1,\nu}^{(k)}$. If $t^{(k)}$ was determined before the nonces of the session, then the function f in Equation (1) is fixed and can not be influenced by the attacker. This is accomplished by the scheme shown in Figure 2, which associates a fixed tweak when generating the nonces and is, therefore, not vulnerable to the attack.

Instead of generating the tweak in Sign as shown in Figure 2, one can also prevent the attack by modifying Sign to take the tweaked secret key $sk_1 + t^{(k)} \bmod p$ or tweaked public key $X_1 g^{t^{(k)}}$ as input. This modified scheme would then write the tweaked secret or public key into $state_1$ and make sure that $\mathsf{Sign}'(state_1, \ldots)$ outputs a signature for the tweaked secret or public key contained in $state_1$.

It is also possible to stop the attack without having to determine tweak $t^{(k)}$ before outputting the session's nonces. This can be achieved by changing the scheme so that each signer i gets a different nonce coefficient b_i instead of using a single nonce coefficient b for all signers of a signing session. However, using a separate nonce coefficient b_i increases the communication complexity of the scheme because it prevents nonce aggregation via $\mathsf{SignAgg}$. In this scheme, all other signers' nonces $R_{2,1}, \dots, R_{2,\nu}, \dots, R_{n,1}, \dots, R_{n,\nu}$ are required to be input to Sign and b_i, R and s_1 are computed as

$$
\begin{aligned}
&b_i := \mathsf{H}_{\text{non}}(\widetilde{X}, (\boldsymbol{R_{1,1}, \dots, R_{1,\nu}, \dots, R_{n,1}, \dots, R_{n,\nu}}), m, \mathbf{X_i}) \quad i \in [1, n] \\
&R := \prod_{\mathbf{i=1}}^{\mathbf{n}} \prod_{j=1}^{\nu} R_{\mathbf{i},j}^{b_{\mathbf{i}}^{j-1}} \\
&s_1 := c a_1 x_1 + \sum_{i=1}^{\nu} r_{1,j} b_{\mathbf{1}}^{j-1} \bmod p.
\end{aligned}
$$

7 Which Schemes are Vulnerable?

The attack discussed in this note targets the key aggregation coefficient a_1 in $\mathsf{MuSig2}$. $\mathsf{MuSig1}$ would be similarly vulnerable if the adversary can select the tweak to sign with after seeing the nonce. Schemes without a key aggregation coefficient, e.g., those relying on proof-of-knowledge of the public key instead of MuSig-style key aggregation, are not affected.

Acknowledgments

We thank Yannick Seurin for identifying a vulnerability in a related scheme that ultimately led to the discovery of the attack discussed in this note.

References

[BLL+21] F. Benhamouda, T. Lepoint, J. Loss, M. Orrù, and M. Raykova. "On the (in)security of ROS". In: *EUROCRYPT 2021*. 2021.

[NRS21] J. Nick, T. Ruffing, and Y. Seurin. "MuSig2: simple two-round Schnorr multi-signatures". In: *Annual International Cryptology Conference*. Springer. 2021, pp. 189–221.

[Wag02] D. Wagner. "A Generalized Birthday Problem". In: 2002, pp. 288–303. DOI: 10.1007/3-540-45708-9_19.

$\underline{\mathsf{Setup}(1^\lambda)}$

$(\mathbb{G}, p, g) \leftarrow \mathsf{GrGen}(1^\lambda)$
Select three hash functions
$\quad \mathsf{H}_{\mathrm{agg}}, \mathsf{H}_{\mathrm{non}}, \mathsf{H}_{\mathrm{sig}} : \{0,1\}^* \rightarrow \mathbb{Z}_p$
$par := ((\mathbb{G}, p, g), \mathsf{H}_{\mathrm{agg}}, \mathsf{H}_{\mathrm{non}}, \mathsf{H}_{\mathrm{sig}})$
return par

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

$x \leftarrow\!\!\$\ \mathbb{Z}_p\ ;\ X := g^x$
$sk := x\ ;\ pk := X$
return (sk, pk)

$\underline{\mathsf{KeyTweak}()}$

$\mathbf{t} \leftarrow\!\!\$\ \mathbb{Z}_\mathbf{p}$
return t

$\underline{\mathsf{KeyAggCoef}(L, X_i)}$

return $\mathsf{H}_{\mathrm{agg}}(L, X_i)$

$\underline{\mathsf{KeyAgg}(L)}$

$\{X_1, \ldots, X_n\} := L$
for $i := 1 \ldots n$ **do**
$\quad a_i := \mathsf{KeyAggCoef}(L, X_i)$
return $\widetilde{X} := \prod_{i=1}^n X_i^{a_i}$

$\underline{\mathsf{Ver}(\widetilde{pk}, m, \sigma)}$

$\widetilde{X} := \widetilde{pk}\ ;\ (R, s) := \sigma$
$c := \mathsf{H}_{\mathrm{sig}}(\widetilde{X}, R, m)$
return $(g^s = R\widetilde{X}^c)$

$\underline{\mathsf{Sign}()}$

$\mathbf{t} := \mathsf{KeyTweak}()$
// Local signer has index 1.
for $j := 1 \ldots \nu$ **do**
$\quad r_{1,j} \leftarrow\!\!\$\ \mathbb{Z}_p\ ;\ R_{1,j} := g^{r_{1,j}}$
$out_1 := (R_{1,1}, \ldots, R_{1,\nu})$
$state_1 := (r_{1,1}, \ldots, r_{1,\nu}, \mathbf{t})$
return $(out_1, state_1)$

$\underline{\mathsf{SignAgg}(out_1, \ldots, out_n)}$

for $i := 1 \ldots n$ **do**
$\quad (R_{i,1}, \ldots, R_{i,\nu}) := out_i$
for $j := 1 \ldots \nu$ **do**
$\quad R_j := \prod_{i=1}^n R_{i,j}$
return $out := (R_1, \ldots, R_\nu)$

$\underline{\mathsf{Sign}'(state_1, out, sk_1, m, (pk_2, \ldots, pk_n))}$

// Sign' must be called at most once per $state_1$.
$(r_{1,1}, \ldots, r_{1,\nu}, \mathbf{t}) := state_1$
$x_1 := sk_1 + \mathbf{t} \bmod \mathbf{p}\ ;\ X_1 := g^{x_1}$
$(X_2, \ldots, X_n) := (pk_2, \ldots, pk_n)$
$L := \{X_1, \ldots, X_n\}$
$a_1 := \mathsf{KeyAggCoef}(L, X_1)$
$\widetilde{X} := \mathsf{KeyAgg}(L)$
$(R_1, \ldots, R_\nu) := out$
$b := \mathsf{H}_{\mathrm{non}}(\widetilde{X}, (R_1, \ldots, R_\nu), m)$
$R := \prod_{j=1}^\nu R_j^{b^{j-1}}$
$c := \mathsf{H}_{\mathrm{sig}}(\widetilde{X}, R, m)$
$s_1 := ca_1x_1 + \sum_{i=1}^\nu r_{1,j} b^{j-1} \bmod p$
$state'_1 := R;\ out'_1 := s_1$
return $(state'_1, out'_1)$

$\underline{\mathsf{SignAgg}'(out'_1, \ldots, out'_n)}$

$(s_1, \ldots, s_n) := (out'_1, \ldots, out'_n)$
$s := \sum_{i=1}^n s_i \bmod p$
return $out' := s$

$\underline{\mathsf{Sign}''(state'_1, out')}$

$R := state'_1\ ;\ s := out'$
return $\sigma := (R, s)$

Fig. 2. The multi-signature scheme $\mathsf{MuSig2Tweak}[\mathsf{GrGen}, \nu]$. The differences to $\mathsf{MuSig2}[\mathsf{GrGen}, \nu]$ are displayed in **red**.