A Note on Unforgeability of MuSig2 with Key Tweaking

Jonas Nick[1], Tim Ruffing[1], and Yannick Seurin[2]

[1] Blockstream
[2] ANSSI, Paris, France

Abstract. *Key Tweaking* refers to the process of producing a new pair of secret and public key from a given pair. 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 show that a variant of MuSig2 with naive tweaking is insecure and propose a variant that is not vulnerable against the attack.

1 The Vulnerable Scheme

Figure 1 shows MuSig2NaiveTweak, a multi-signature scheme that is identical to MuSig2 except that it has the additional capability of 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 power necessary to make the attack work. In particular, the honest signer generates uniformly random tweaks without input from the adversary and outputs possible tweaks. MuSig2NaiveTweak uses *additive* tweaking, but the attack similarly applies to *multiplicative* tweaking.

2 Generalized Birthday Problem

The attack against MuSig2NaiveTweak makes use of Wagner's algorithm for solving the Generalized Birthday Problem It can be defined as follows for the purpose of this paper: 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 $O(2^{2\sqrt{\log_2(p)}})$.

Jonas' note: Perhaps can use BLOR? "If the attacker is able to open more sessions concurrently, the improved polynomial-time attack by Benhamouda *et al.* [**add:BLOR20**] assumes $k_{\max} > \log_2 p$ sessions, but then has complexity $O(k_{\max} \log_2 p)$ and a negligible running time in practice."

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

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

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

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

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

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

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

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

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

```
$\mathsf{SignAgg}(out_1, \ldots, out_n)$
for $i := 1 \ldots n$ do
    $(R_{i,1}, \ldots, R_{i,\nu}) := out_i$
for $j := 1 \ldots \nu$ do
    $R_j := \prod_{i=1}^n R_{i,j}$
return $out := (R_1, \ldots, R_\nu)$
```

```
$\mathsf{Sign}'(state_1, out, sk_1, m, (pk_2, \ldots, pk_n), \mathbf{t})$
// $\mathsf{Sign}'$ 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
$(r_{1,1}, \ldots, r_{1,\nu}) := state_1$
$x_1 := sk_1 + \mathbf{t} \bmod \mathbf{p}$ ; $X_1 := g^{x_1}$
$(X_2, \ldots, X_n) := (pk_2, \ldots, pk_n)$
$L := \{X_1, \ldots, X_n\}$
$a_1 := \mathsf{KeyAggCoef}(L, X_1)$
$\widetilde{X} := \mathsf{KeyAgg}(L)$
$(R_1, \ldots, R_\nu) := out$
$b := \mathsf{H}_{\mathrm{non}}(\widetilde{X}, (R_1, \ldots, R_\nu), m)$
$R := \prod_{j=1}^{\nu} R_j^{b^{j-1}}$
$c := \mathsf{H}_{\mathrm{sig}}(\widetilde{X}, R, m)$
$s_1 := ca_1x_1 + \sum_{i=1}^{\nu} r_{1,j} b^{j-1} \bmod p$
$state_1' := R$ ; $out_1' := s_1$
return $(state_1', out_1')$
```

```
$\mathsf{SignAgg}'(out_1', \ldots, out_n')$
$(s_1, \ldots, s_n) := (out_1', \ldots, out_n')$
$s := \sum_{i=1}^n s_i \bmod p$
return $out' := s$
```

```
$\mathsf{Sign}''(state_1', out')$
$R := state_1'$ ; $s := out'$
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 value $t^{(\ell)}$ 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 honest signer is asked for a partial signature using value $C^{f(k)}$ 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 last summand can be subtracted as

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

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

4 BLOR attack

Benhamouda *et al.* [**add:BLOR20**] give an algorithm that solves the ROS problem and can be applied to attack to break 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 that only have two elements.

5 Where the security proof of MuSig2 fails against MuSig2NaiveTweak

TODO Look at ROM proof of MuSig (section) We have a != a but b = b

Jonas' note: this section is not strictly necessary

6 A Fixed MuSig2 Variant with Tweaking

TODO Section's bla and blub indicate that is secure when Make sure that attacker can not choose the signers pubkey after seeing the signers nonces. Then the specific attack can not work.

7 Conclusion

Other multisignature schemes that use a key agg coefficient are similarly vulnerable. Other multisigs and fix?

- MuSig1 with naive tweaking: if tweak comes in
- multisig with PoK and naive tweaking: there's no "naive" tweaking with PoK

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

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

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

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

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

```
$\mathsf{SignAgg}(out_1, \ldots, out_n)$
for $i := 1 \ldots n$ do
    $(R_{i,1}, \ldots, R_{i,\nu}) := out_i$
for $j := 1 \ldots \nu$ do
    $R_j := \prod_{i=1}^{n} R_{i,j}$
return $out := (R_1, \ldots, R_\nu)$

$\mathsf{Sign}'(state_1, out, sk_1, m, (pk_2, \ldots, pk_n))$
// $\mathsf{Sign}'$ must be called at most once per $state_1$.
$(r_{1,1}, \ldots, r_{1,\nu}, \mathbf{t}) := state_1$
$x_1 := sk_1 + \mathbf{t} \bmod \mathbf{p}\,;\ X_1 := g^{x_1}$
$(X_2, \ldots, X_n) := (pk_2, \ldots, pk_n)$
$L := \{X_1, \ldots, X_n\}$
$a_1 := \mathsf{KeyAggCoef}(L, X_1)$
$\widetilde{X} := \mathsf{KeyAgg}(L)$
$(R_1, \ldots, R_\nu) := out$
$b := \mathsf{H}_{\mathrm{non}}(\widetilde{X}, (R_1, \ldots, R_\nu), m)$
$R := \prod_{j=1}^{\nu} R_j^{b^{j-1}}$
$c := \mathsf{H}_{\mathrm{sig}}(\widetilde{X}, R, m)$
$s_1 := c a_1 x_1 + \sum_{i=1}^{\nu} r_{1,j} b^{j-1} \bmod p$
$state_1' := R\,;\ out_1' := s_1$
return $(state_1', out_1')$

$\mathsf{SignAgg}'(out_1', \ldots, out_n')$
$(s_1, \ldots, s_n) := (out_1', \ldots, out_n')$
$s := \sum_{i=1}^{n} s_i \bmod p$
return $out' := s$

$\mathsf{Sign}''(state_1', out')$
$R := state_1'\,;\ s := out'$
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**.