A Note on Unforgeability of MuSig2 with 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 commit to a value in 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

TODO: fix nu everywhere TODO This is MuSig with some version of tweaking. weak because attacker only has control over the contract, not the tweak. TODO: what is tweaking For simplicity only two nonces See Figure 2.

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 first draws $\ell_{\max} \in O(2^{2\sqrt{\log_2(p)}})$ values $C^{(1)}, \ldots, C^{(\ell_{\max})}$ at random and computes tweaks $t^{(\ell)} = \mathsf{H}_{\text{tweak}}(X_1', C^{(\ell)})$ for all $\ell \in [1, \ell_{\max}]$. Let $L = \{X_1' g^{t^{(1)}}, \ldots, X_1' g^{t^{(\ell_{\max})}}\}$ be the multiset of public keys and $\widetilde{X} = \mathsf{KeyAgg}(L)$ be the corresponding aggregate public key.

The adversary opens $k_{\max} = 2^{\sqrt{\log_2(p)}-1}$ concurrent signing sessions by requesting $k_{\max}$ nonce pairs $R_{1,1}^{(1)}, R_{1,2}^{(1)}, \ldots, R_{1,1}^{(k_{\max})}, R_{1,2}^{(k_{\max})}$ from the honest signer with (untweaked) public key $X_1' = g^{x_1}$. Given $R_1 = \sum_{k=1}^{k_{\max}} R_{1,1}^{(k)}, R_2 = \sum_{k=1}^{k_{\max}} R_{1,2}^{(k)}$ and a forgery target message m^*, the adversary computes $b = \mathsf{H}_{\text{non}}(\widetilde{X}, (R_1, R_2), m)$ and $R^* = \prod_{k=1}^{k_{\max}} R_{1,1}^{(k)} (R_{1,2}^{(k)})^b$. and uses Wagner's algorithm to find a value $C^{(\ell)}, \ell \in [1, \ell_{\max}]$ for each session k (in other words a function $f : [1, k_{\max}] \to [1, \ell_{\max}]$) such that TODO

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

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

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

$\mathsf{KeyGen}()$

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

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

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

$\mathsf{KeyAgg}(L)$

$\{X_1, \ldots, X_n\} := L$
for $i := 1 \ldots n$ **do**
$\quad a_i := \mathsf{KeyAggCoef}(L, X_i)$
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**
$\quad r_{1,j} \leftarrow\!\$\ \mathbb{Z}_p\,;\ R_{1,j} := g^{r_{1,j}}$
$out_1 := (R_{1,1}, \ldots, R_{1,\nu})$
$state_1 := (r_{1,1}, \ldots, r_{1,\nu})$
return $(out_1, state_1)$

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

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 $out := (R_1, \ldots, R_\nu)$

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

// Sign' must be called at most once per $state_1$.
$(r_{1,1}, \ldots, r_{1,\nu}) := state_1$
$x_1 := sk_1\,;\ X_1' := g^{x_1}\,;\ \mathbf{t} := \mathsf{H}_{\mathrm{tweak}}(\mathbf{X_1'}, \mathbf{C})\,;\ \mathbf{X_1} := \mathbf{X_1'}\mathbf{g^t}$
$(X_2, \ldots, X_n) := (pk_2, \ldots, pk_n)$
$L := \{X_1, \ldots, X_n\}$
$a_1 := \mathsf{KeyAggCoef}(L, X_1)$
$\widetilde{X} := \mathsf{KeyAgg}(L)$
$(R_1, \ldots, R_\nu) := out$
$b := \mathsf{H}_{\mathrm{non}}(\widetilde{X}, (R_1, \ldots, R_\nu), m)$
$R := \prod_{j=1}^\nu R_j^{b^{j-1}}$
$c := \mathsf{H}_{\mathrm{sig}}(\widetilde{X}, R, m)$
$s_1 := ca_1(\mathbf{x_1} + \mathbf{t}) + \sum_{i=1}^\nu r_{1,j} b^{j-1} \bmod p$
$state_1' := R\,;\ out_1' := s_1$
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**.

The honest signer will reply with $k_{\max}$ partial signatures $s_1^{(k)} = r_{1,1}^{(k)} + br_{1,2}^{(k)} + c^{(k)} \cdot a_1^{(k)}(x_1 + t^{(f(k))})$ which allows the adversary to compute

$$s_1^{*'} = \sum_{k=1}^{k_{\max}} s_1^{(k)} \tag{2}$$

$$= \sum_{k=1}^{k_{\max}} r_{1,1}^{(k)} br_{1,2}^{(k)} + \left(\sum_{k=1}^{k_{\max}} c^{(k)} a_1^{(k)} \right) \cdot x_1 + \sum_{k=1}^{k_{\max}} c^{(k)} a_1^{(k)} t^{(f(k))} \tag{3}$$

$$= \log_g(R^*) + c^* x_1 + \sum_{k=1}^{k_{\max}} c^{(k)} a_1^{(k)} t^{(f(k))} \tag{4}$$

where the last equality follows from Equation (1). Then the adversary can subtract the unwanted term as follows

$$s_1^* = s_1^{*'} - \sum_{k=1}^{k_{\max}} c^{(k)} a_1^{(k)} t^{(f(k))}$$

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

4 Where the security proof of MuSig2 fails against MuSig2NaiveTweak

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

5 Tweaking in MuSig2 without

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. Here, adversary can have full control over t.

6 Conclusion

- Other multisigs and fix? - MuSig1, PoP, FROST - looses freedom for the dev

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

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

$\mathsf{KeyGen}()$

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

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

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

$\mathsf{KeyAgg}(L)$

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

$\mathsf{Sign}(sk_1, \mathbf{t})$

// Local signer has index 1.
for $j := 1 \ldots \nu$ **do**
$\quad r_{1,j} \leftarrow\!\!\$\ \mathbb{Z}_p\ ;\ R_{1,j} := g^{r_{1,j}}$
$out_1 := (R_{1,1}, \ldots, R_{1,\nu})$
$state_1 := (r_{1,1}, \ldots, r_{1,\nu}, sk_1 + \mathbf{t} \bmod \mathbf{p})$
return $(out_1, state_1)$

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

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 $out := (R_1, \ldots, R_\nu)$

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

// Sign' must be called at most once per $state_1$.
$(r_{1,1}, \ldots, r_{1,\nu}, \mathbf{x_1}) := state_1$
$X_1 := g^{x_1}$
$(X_2, \ldots, X_n) := (pk_2, \ldots, pk_n)$
$L := \{X_1, \ldots, X_n\}$
$a_1 := \mathsf{KeyAggCoef}(L, X_1)$
$\widetilde{X} := \mathsf{KeyAgg}(L)$
$(R_1, \ldots, R_\nu) := out$
$b := \mathsf{H}_{\mathrm{non}}(\widetilde{X}, (R_1, \ldots, R_\nu), m)$
$R := \prod_{j=1}^\nu R_j^{b^{j-1}}$
$c := \mathsf{H}_{\mathrm{sig}}(\widetilde{X}, R, m)$
$s_1 := ca_1x_1 + \sum_{i=1}^\nu r_{1,j} b^{j-1} \bmod p$
$state_1' := R\ ;\ out_1' := s_1$
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. 2. The multi-signature scheme $\mathsf{MuSig2Tweak}[\mathsf{GrGen}, \nu]$. The differences to $\mathsf{MuSig2}[\mathsf{GrGen}, \nu]$ are displayed in **red**.