Taproot in the Generic Group Model

Lloyd Fournier (lloyd.fourn@gmail.com)

Introduction

BIP-340 (Schnorr) and BIP-341 (Taproot) are proposed upgrades to the Bitcoin network that create a new type of public key output which can be spent by (i) a Schnorr signature under that public key or (ii) revealing a hidden commitment to a script *inside* the public key and satisfying the conditions of the script. Framed as a hybrid commitment scheme:

$\mathsf{TapCom}(G, m)$
$x \leftarrow\!\!\$\ \mathbb{Z}_q; X \leftarrow xG$
$y \leftarrow H(f(X)\|m); Y \leftarrow yG$
$\mathsf{com}_{pk} \leftarrow X + Y$
$\mathsf{open} := (X, m)$
$sk \leftarrow x + y$
return $(sk, (\mathsf{com}_{pk}, \mathsf{open}))$

$\mathsf{TapOpen}(G, \mathsf{com}_{pk}, \mathsf{open})$
$(X, m) := \mathsf{open}$
if $X + H(f(X)\|m)G = \mathsf{com}_{pk}$
  return m
else return $\perp$

If the hash function H is idealised as a random oracle then the scheme is secure[1]. Taking inspiration from [2], we instead idealise the elliptic curve group in the *Generic Group Model* to isolate what properties the hash function requires for Taproot to be secure. To compute new group elements the adversary is allowed up to q_G queries to the oracle $\mathcal{G}$ with two elements it already knows (G_1, G_2). The oracle returns a new group element G_3 representing $G_1 - G_2$.

The main hash function properties we consider are:

- Random-Prefix Preimage Resistance (RPP): Strictly weaker assumption than collision resistance. Already required for Schnorr[2].
- Chosen Offset Prefix Collision Resistance (COPC): New assumption for Taproot's binding as commitment scheme. Breaking seems unrelated to collision resistance.

RPP
$(\mathsf{st}, h) \leftarrow\!\!\$\ \mathcal{A}$
$P \leftarrow\!\!\$\ \mathcal{P}$
$m^* \leftarrow\!\!\$\ \mathcal{A}(\mathsf{st}, P)$
return $H(P\|m^*) = h$

COPC
$P_1 \leftarrow\!\!\$\ \mathcal{P}$
$(\mathsf{st}, \delta) \leftarrow\!\!\$\ \mathcal{A}(P_1)$
$P_2 \leftarrow\!\!\$\ \mathcal{P}$
$(m_1, m_2) \leftarrow\!\!\$\ \mathcal{A}(\mathsf{st}, P_2)$
return $H(P_1\|m_1) - H(P_2\|m_2) = \delta$

Forging an Opening

Can an adversary forge a fake opening on someone else's coins? Call this the *Taproot Forge* problem (TF). RPP is necessary for TF to be hard:

TF
$(\mathsf{st}, m_1) \leftarrow\!\!\$\ \mathcal{A}$
$G \leftarrow\!\!\$\ \mathbb{G}$
$(\cdot, (\mathsf{com}_{pk}, \mathsf{open})) \leftarrow\!\!\$\ \mathsf{TapCom}(G, m)$
$(X^*, m_2) \leftarrow\!\!\$\ \mathcal{A}(st, G, \mathsf{com}_{pk}, \mathsf{open})$
return $X^* + H(f(X^*)\|m_2)G = \mathsf{com}_{pk}$
  $\wedge\, m_2 \neq m_1$

$\mathcal{R} : \mathsf{TF} \rightarrow \mathsf{RPP}$
[Challenger: $\rightarrow m_1$; $\leftarrow G, \mathsf{com}_{pk}, \mathsf{open}$]
$(h, \mathsf{st}) \leftarrow\!\!\$\ \mathcal{A}_{\mathsf{RPP}}; T := \mathsf{com}_{pk}$
$C \leftarrow T - hG; P \leftarrow f(C)$
$m_2 \leftarrow\!\!\$\ \mathcal{A}_{\mathsf{RPP}}(\mathsf{st}, P)$
return (C, m_2)

To show RPP is sufficient, $\mathcal{R}$ guesses which query to $\mathcal{G}$ will be used for the malicious *Taproot internal key*, C.

$\mathcal{R} : \mathsf{RPP} \rightarrow \mathsf{TF}$
$(\mathsf{st}, m_1) \leftarrow\!\!\$\ \mathcal{A}_{\mathsf{TF}}$
$(G, X, T) \leftarrow\!\!\$\ \mathbb{G}^3$
$x \leftarrow\!\!\$\ \mathbb{Z}_q$
$y \leftarrow H(f(X)\|m_1)$
$t \leftarrow x + y$
$\mathcal{L} := \{(G, 1, 0), (X, 0, 1), (T, y, 1)\}$
$i_0 \leftarrow\!\!\$\ \{1, 2, \ldots, q_G\}; i \leftarrow 1$
$(X^*, m_2) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{G}}_{\mathsf{TF}}(\mathsf{st}, G, T, (X, m_1))$
if $X^* = \tilde{X}$
  // $\implies X^* + H(P\|m_2)G = T$
  // $\implies H(P\|m_2) = h$
  return m^*
else return $\perp$

Simulate $\mathcal{G}(G_1, G_2)$
$(a_1, b_1) \leftarrow \mathcal{L}[G_1]; (a_1, b_1) \leftarrow \mathcal{L}[G_2]$
$(a_3, b_3) \leftarrow (a_1 - a_2, b_1 - b_2)$
if $\exists(\cdot, a_i, b_i) \in \mathcal{L} \mid a_i + b_i x = a_3 + b_3 x$
  abort
else if $i_0 = i$
  $h \leftarrow t - (a_3 + b_3 x)$
  [Challenger: $\rightarrow h$; $\leftarrow P$]
  $\tilde{X} \leftarrow f^{-1}(P); G_3 := \tilde{X}$
else $G_3 \leftarrow\!\!\$\ \mathbb{G}$
$\mathcal{L} := \mathcal{L} \cup \{(G_3, a_3, b_3)\}$
$i \leftarrow i + 1$
return G_3

MuSig with Covert Taproot

Can an adversary come up with a covert Taproot spend by choosing their MuSig public key maliciously? Call this the *MuSig Covert Taproot* (MCT) problem.

MCT
$X_1 \leftarrow\!\!\$\ \mathbb{G}$
$(X_2, (C, m)) \leftarrow\!\!\$\ \mathcal{A}(X_1)$
$X \leftarrow \mathsf{MuSig}(X_1, X_2)$
return $X = C + H(f(C)\|m)G$

$\mathsf{MuSig}(X_1, X_2)$
$L := (X_1, X_2)$
$c_1 \leftarrow H_{\mathsf{agg}}(L, X_1)$
$c_2 \leftarrow H_{\mathsf{agg}}(L, X_2)$
return $c_1 X_1 + c_2 X_2$

RPP is sufficient to ensure MCT is hard if X_2 is queried before C. If the reduction guesses correctly which queries will be used for X_2 and C it solves RPP. This approach only works for 2-party MuSig.

$\mathcal{R} : \mathsf{RPP} \rightarrow \mathsf{MCT}$
$x_1 \leftarrow\!\!\$\ \mathbb{Z}_q; (G, X_1) \leftarrow\!\!\$\ \mathbb{G}^2$
$(i_0, i_1) \leftarrow\!\!\$\ \{1, 2, \ldots, q_G\}$ **s.t.** $i_0 < i_1$
$\mathcal{L} := \{(G, 1, 0), (X_1, 0, 1)\}$
$(X_2, (C, m)) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{G}}_{\mathsf{RPP}}(G, X_1)$
if $X_2 = \tilde{X}_2 \wedge C = \tilde{C}$
  // $\implies \mathsf{MuSig}(X_1, X_2) = C + H(P\|m)G$
  // $\implies H(P\|m) = h$
  return m
else
  return $\perp$

Simulate $\mathcal{G}(G_1, G_2)$
$(a_1, b_1) \leftarrow \mathcal{L}[G_1]; (a_2, b_2) \leftarrow \mathcal{L}[G_2]$
$(a_3, b_3) \leftarrow (a_1 - a_2, b_1 - b_3)$
if $\exists(\cdot, a_i, b_i) \in \mathcal{L} \mid a_i + b_i x_1 = a_3 + b_3 x_1$
  abort
else if $i = i_0$
  $\tilde{X}_2 \leftarrow\!\!\$\ \mathbb{G}; \tilde{x}_2 \leftarrow a_3 + b_3 x_1; G_3 := \tilde{X}_2$
else if $i = i_1$
  $L := (X_1, \tilde{X}_2)$
  $x \leftarrow H_{\mathsf{agg}}(L, X_1)x_1 + H_{\mathsf{agg}}(L, \tilde{X}_2)\tilde{x}_2$
  $h \leftarrow x - (a_3 + b_3 x_1)$
  [Challenger: $\rightarrow h$; $\leftarrow P$]
  $\tilde{C} \leftarrow f^{-1}(P); G_3 := \tilde{C}$
else $G_3 \leftarrow\!\!\$\ \mathbb{G}$
$\mathcal{L} := \mathcal{L} \cup \{(G_3, a_3, b_3)\}$
$i \leftarrow i + 1$
return G_3

If C is queried before X_2, or $C = X_2$ then (I think) $\mathcal{A}$ can be used to break Preimage Resistance.

MuSig Second Covert Taproot

Can an adversary create a second malicious Taproot spend in addition to an agreed upon on one by choosing their parameters maliciously? Call this the *MuSig Second Covert Taproot* (MSCT) problem. COPC is necessary for MSCT to be hard:

MSCT
$X_1 \leftarrow\!\!\$\ \mathbb{G}$
$(X_2, m_1, (C, m_2)) \leftarrow\!\!\$\ \mathcal{A}(X_1)$
$X \leftarrow \mathsf{MuSig}(X_1, X_2)$
$\mathsf{com}_{pk} \leftarrow X + H(f(X)\|m_2)$
return $\mathsf{com}_{pk} = C + H(f(C)\|m_2)$
  $\wedge\, m_2 \neq m_1$

$\mathcal{R}(X_1) : \mathsf{MSCT} \rightarrow \mathsf{COPC}$
$X_2 \leftarrow\!\!\$\ \mathbb{G}$
$X \leftarrow \mathsf{MuSig}(X_1, X_2); P_1 \leftarrow f(X)$
$(\mathsf{st}, \delta) \leftarrow\!\!\$\ \mathcal{A}(P_1)$
$C \leftarrow X - \delta G; P_2 \leftarrow f(C)$
$(m_1, m_2) \leftarrow \mathcal{A}(\mathsf{st}, P_2)$
return $(X_1, m_1, (C, m_2))$

COPC is sufficient to make MSCT hard where the Taproot internal keys for are not the same i.e $X \neq C$. If the reduction guesses which queries will be used for X and C correctly (in any order) it solves COPC.

$\mathcal{R}(P_1) : \mathsf{COPC} \rightarrow \mathsf{MSCT}$
$D_1 \leftarrow f^{-1}(P_1)$
$x_1 \leftarrow\!\!\$\ \mathbb{Z}_q; (G, X_1) \leftarrow\!\!\$\ \mathbb{G}^2$
$(i_0, i_1) \leftarrow\!\!\$\ \{1, 2, \ldots, q_G\}$ **s.t.** $i_0 < i_1$
$i \leftarrow 1$
$\mathcal{L} := \{(G, 1, 0), (X_1, 0, 1)\}$
$(X_2, m_1, (C, m_2)) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{G}}_{\mathsf{MSCT}}(G, X_1)$
$X \leftarrow \mathsf{MuSig}(X_1, X_2)$
if $X = D_1 \wedge C = D_2$
  // $X + H(P_1\|m_1)G = C + H(P_2\|m_2)G$
  return (m_1, m_2)
else if $X = D_2 \wedge C = D_1$
  // $X + H(P_2\|m_1)G = C + H(P_1\|m_2)G$
  return (m_2, m_1)
else return $\perp$

Simulate $\mathcal{G}(G_1, G_2)$
$(a_1, b_1) \leftarrow \mathcal{L}[G_1]; (a_2, b_2) \leftarrow \mathcal{L}[G_2]$
$(a_3, b_3) \leftarrow (a_1 - a_2, b_1 - b_2)$
if $\exists(\cdot, a_i, b_i) \in \mathcal{L} \mid a_i + b_i x_1 = a_3 + b_3 x_1$
  abort
else if $i = i_0$
  $d_1 \leftarrow a_3 + b_3 x_1$
  $G_3 := D_1$
else if $i = i_1$
  $d_2 \leftarrow a_3 + b_3 x_1$
  $\delta \leftarrow d_1 - d_2$
  [Challenger: $\rightarrow \delta$; $\leftarrow P_2$]
  $D_2 \leftarrow f^{-1}(P_2); G_3 := D_2$
else $G_3 \leftarrow\!\!\$\ \mathbb{G}$
$\mathcal{L} := \mathcal{L} \cup \{(G_3, a_3)\}$
$i \leftarrow i + 1$
return G_3

If $X = C$, then $\mathcal{A}$ clearly breaks collision resistance.

Remarks

- These reductions are incomplete – they do not account for $\mathcal{A}$ choosing G or X_1 etc as one of the elements they return. They can be modified to fix this.
- To actually steal coins, the malicious Taproot openings have to be valid Merkle Root (m can't be arbitrary).
- If coin tossing is used to generate joint key instead of MuSig then security in all scenarios follows from RPP.

[1] A. Poelstra, "Taproot Security Proof." `https://github.com/apoelstra/taproot`, 2018.
[2] G. Neven, N. P. Smart, and B. Warinschi, "Hash function requirements for schnorr signatures," *Journal of Mathematical Cryptology*, vol. 3, no. 1, pp. 69–87, 2009.