Taproot Security Proof

Andrew Poelstra[*]

(None) (commit (None))

Definition 1. *Throughout, we assume a security parameter λ, a cyclic group $\mathcal{G} = \langle G \rangle$ for which no algorithm can solve the discrete logarithm in order $p(\lambda)$ except with probability* $\mathrm{neg}(\lambda)$*, and two hash functions H^{pk}, H^m modeled as random oracles whose output is of length $q(\lambda)$.*

Definition 2. *Let $\beta = (\mathrm{KeyGen}^\beta, \mathrm{Sign}^\beta, \mathrm{Verify}^\beta)$ be a signature scheme. We define the* Taproot *of β, denoted TR^β, by the four algorithms*

- $\mathrm{KeyGen}(b)$ *takes a bit b and outputs a keypair* $(\mathrm{sk}_0, \mathrm{sk}_1, \mathrm{pk})$. *It acts as follows.*
  It first selects $\mathrm{sk}^\alpha \xleftarrow{\$} \mathbb{Z}/p(\lambda)\mathbb{Z}$, and computes $\mathrm{pk}^\alpha \leftarrow \mathrm{sk}^\alpha G(\lambda)$.
  - *If $b = 0$, it simply outputs* $(\mathrm{sk}^\alpha, \perp, \mathrm{pk}^\alpha)$.
  - *If $b = 1$, it obtains* $(\mathrm{sk}^\beta, \mathrm{pk}^\beta) \leftarrow \mathrm{KeyGen}^\beta$ *and computes a "tweak" $\varepsilon \leftarrow H^{\mathrm{pk}}(\mathrm{pk}^\beta \| \mathrm{pk}^\alpha)$. It then outputs $\mathrm{sk}_0 \leftarrow \mathrm{sk}^\alpha + \varepsilon$, $\mathrm{sk}_1 = (\mathrm{sk}^\beta, \mathrm{pk}^\alpha, \mathrm{pk}^\beta)$ and $\mathrm{pk} \leftarrow \mathrm{pk}^\alpha + \varepsilon G$.*
- $\mathrm{Sign}^1(\mathrm{sk}_0, m)$ *takes a secret key* sk_0 *and message m and outputs a signature σ. It acts as follows.*
  - *It computes $k \xleftarrow{\$} \mathbb{Z}/p(\lambda)\mathbb{Z}$, $R \leftarrow kG$, $\mathrm{pk} \leftarrow \mathrm{sk}_0 G$, and $e \leftarrow H^m(\mathrm{pk}\|R\|m)$.*
  - *It outputs $\sigma = (R, k + e\mathrm{sk}_0)$.*
- $\mathrm{Sign}^2(\mathrm{sk}_1 = (\mathrm{sk}^\beta, \mathrm{pk}^\alpha, \mathrm{pk}^\beta), m)$ *takes a secret key* sk_1 *and message m and outputs a signature σ. It acts as follows.*
  - *It computes $\sigma^\beta = \mathrm{Sign}^\beta(\mathrm{sk}^\beta, m)$.*
  - *It outputs $\sigma = (\mathrm{pk}^\alpha, \mathrm{pk}^\beta, \sigma^\beta)$.*
- $\mathrm{Verify}(\mathrm{pk}, \sigma, m)$ *takes a public key* pk*, signature σ and message m. It outputs a bit b. It acts as follows.*
  - *If $\sigma = (R, s)$, it computes $e \leftarrow H^m(\mathrm{pk}\|R\|m)$ and accepts iff $sG = R + e\mathrm{pk}$.*
  - *If $\sigma = (\mathrm{pk}^\alpha, \mathrm{pk}^\beta, \sigma^\beta)$, it checks first that $\mathrm{pk} = \mathrm{pk}^\alpha + H^{\mathrm{pk}}(\mathrm{pk}^\beta \| \mathrm{pk}^\alpha)G$. If so, it accepts iff* $\mathrm{Verify}^\beta(\mathrm{pk}^\beta, \sigma^\beta, m)$ *accepts.*

[*] `taproot@wpsoftware.net`

Theorem 1. *If the discrete logarithm problem is hard in $\mathscr{G}$, and β is (strongly) existentially unforgeable, then TR^{β} is (strongly) existentially unforgeable in the sense of the following game between adversary $\mathscr{A}$ and challenger $\mathscr{C}$.*

1. *$\mathscr{A}$ chooses a bit b and sends to $\mathscr{C}$. $\mathscr{C}$ replies with a public key pk generated (in the adversary's view) by $\mathrm{KeyGen}(b)$.*

2. *$\mathscr{A}$ then submits signing queries (b_i, m_i) for $i \in \{1, 2, \ldots, q\}$ where q is bounded by some polynomial in λ. The challenger must respond with a valid signature σ_i of the form generated by Sign^b. (Except that if $b = 0$ a then the challenger may respond $\perp$ whenever $b_i = 1$.)*

3. *$\mathscr{A}$ finally resonds with a signature (σ_*, m_*) of either form, such that $\mathrm{Verify}(\mathrm{pk}, \sigma_*, m_*)$ accepts.*

   If β is strongly existentially unforgeable, we require $(\sigma_*, m_*) \neq (\sigma_i, m_i)$ for all i; otherwise we require further that $m_* \neq m_i$ for all i.

The proof consists of two lemmas which consider adversaries who produce the two different forms of signatures. The intuition is essentially that forging in the form of Sign^2 is essentially just forging on β, while forging in the form of Sign^1 is essentially just forging a Schnorr signature.

Lemma 1. *If such an adversary $\mathscr{A}$ outputs a (strong) forgery in the form of Sign^2 with probability ε, it can be used to produce a (strong) forgery for β with probability $\varepsilon - \mathrm{neg}(\lambda, q)$.*

Proof. Suppose such an adversary $\mathscr{A}$ exists, and consider the following challenger $\mathscr{C}$, with access to a β-challenger. First, $\mathscr{C}$ replies to all random oracle queries uniformly at random.

1. $\mathscr{C}$ chooses a uniformly random keypair $(\mathrm{sk}^{\alpha}, \mathrm{pk}^{\alpha})$.

   - If $b = 0$, he gives pk^{α} to $\mathscr{A}$ and stores $\mathrm{sk} = \mathrm{sk}^{\alpha}$.
   - If $b = 1$, he requests a public key pk^{β} from his β-challenger, computes $\mathrm{pk} = \mathrm{pk}^{\alpha} + H^{\mathrm{pk}}(\mathrm{pk}^{\beta} \| \mathrm{pk}^{\alpha})G$, and gives this to $\mathscr{A}$. He stores $\mathrm{sk} = \mathrm{sk}^{\alpha} + H^{\mathrm{pk}}(\mathrm{pk}^{\beta} \| \mathrm{pk}^{\alpha})$.

2. Given a signing query (b_i, m_i), $\mathscr{A}$ acts as follows. If $b_i = 0$, he executes $\mathrm{Sign}^1(\mathrm{sk}, m_i)$ and replies with the result σ_i. If $b_i = 1$ and $b = 0$, he returns $\perp$.

   If $b_i = 1$ and $b = 1$, he first obtains σ_I^{β} by querying the β-challenger on m_i. He replies with $\sigma_i = (\mathrm{pk}^{\alpha}, \mathrm{pk}^{\beta}, \sigma^{\beta})$.

3. Finally, $\mathscr{A}$ responds with a signature (σ_*, m_*) in the form of Sign^2; that is, $\sigma_* = (\mathrm{pk}_*^{\alpha}, \mathrm{pk}_*^{\beta}, \sigma_*^{\beta})$. Further, σ_*^{β} is a valid signature on m_* with key pk_*^{β}, and $\mathrm{pk} = \mathrm{pk}_*^{\alpha} + H^{\mathrm{pk}}(\mathrm{pk}_*^{\beta} \| \mathrm{pk}_*^{\alpha})$.

   First, $(\mathrm{pk}_*^{\alpha}, \mathrm{pk}_*^{\beta}) = (\mathrm{pk}^{\alpha}, \mathrm{pk}^{\beta})$ except with negligible probability, since H^{pk} is modelled as a random oracle and $\mathscr{A}$ may make only polynomially many queries to it.

   We therefore observe that $\sigma_* = \sigma_i$ iff $\sigma_*^{\beta} = \sigma_i^{\beta}$, so that (σ_*^{β}, m_*) is a (strong) forgery for β iff (σ, m_*) is a (strong) forgery for TR^{β}.

$\square$

Lemma 2. *If such an adversary $\mathscr{A}$ outputs a (strong) forgery in the form of* Sign^1 *with probability ε, it can be used to produce a strong Schnorr signature forgery on $(\mathscr{G}, G, H^m)$ with probability $\varepsilon - \mathrm{neg}(\lambda, q)$.*

Proof. Our challenger $\mathscr{C}$ now acts as follows. He has access to a Schnorr challenger $\mathscr{S}$ who has a random oracle H^S and expects signatures of the form (s, R) with $s = R + eP$ and $e = H^S(P \| R \| m)$.

First, if $b = 0$, $\mathscr{C}$ forwards a public key from $\mathscr{S}$ to $\mathscr{A}$, responds to all signatures with $b_i = 1$ with $\perp$, and otherwise forwards all signatures to $\mathscr{S}$ and H^m queries to H^S, and presents the resulting forgery unmodified as a Schnorr forgery. The result is immediate in this case, so we will assume $b = 1$ for the remainder of the proof.

1. $\mathscr{C}$ requests a public key pk^α from $\mathscr{S}$, chooses $(\mathrm{sk}^\beta, \mathrm{pk}^\beta) \leftarrow \mathrm{KeyGen}^\beta$, and computes $\mathrm{pk} = \mathrm{pk}^\alpha + H^{\mathrm{pk}}(\mathrm{pk}^\beta \| \mathrm{pk}^\alpha) G$. $\mathscr{C}$ sends pk to $\mathscr{A}$.

2. $\mathscr{C}$ responds to H^{pk} queries uniformly randomly. $\mathscr{C}$ responds to H^m queries by first replacing pk with pk^α and vice-versa, then forwarding the query to H^S.

3. $\mathscr{C}$ responds to signature queries (b_i, m_i) as follows. If $b_i = 1$, $\mathscr{C}$ obtains $\sigma_i \leftarrow \mathrm{Sign}^2((\mathrm{sk}^\beta, \mathrm{pk}^\alpha, \mathrm{pk}^\beta), m_i)$ and returns this.

   If $b_i = 0$, $\mathscr{C}$ requests a signature (s_i, R_i) from $\mathscr{S}$ on m_i. Letting $e_i = H^S(\mathrm{pk}^\alpha \| R \| m_i)$, $\mathscr{C}$ adds $e_i H^{\mathrm{pk}}(\mathrm{pk}^\beta \| \mathrm{pk}^\alpha)$ to s_i to obtain s_i', and replies with $\sigma_i = (s_i', R_i)$.

   We observe that this is actually a valid signature, since

   $$\begin{aligned} s_i' G &= s_i G + e_i H^{\mathrm{pk}}(\mathrm{pk}^\beta \| \mathrm{pk}^\alpha) G \\ &= R_i + e_i(\mathrm{pk}^\alpha + H^{\mathrm{pk}}(\mathrm{pk}^\beta \| \mathrm{pk}^\alpha) G) \\ &= R_i + e_i \mathrm{pk} \end{aligned}$$

   But $\mathscr{C}$'s replies to H^m ensure that $e_i = H^S(\mathrm{pk}^\alpha \| R \| m_i) = H^m(\mathrm{pk} \| R \| m_i)$, so this is exactly the verification equation for TR^β.

4. Finally, $\mathscr{A}$ replies with a forgery (σ_*, m_*) in the form of Sign^1 such that $(\sigma_*, m_*) \neq (\sigma_i, m_i)$ for any i and $\mathrm{Verify}(\mathrm{pk}, \sigma_*, m_*)$ accepts.

   That is, $\sigma_* = (s_*, R_*)$, $e_* = H^m(\mathrm{pk} \| R_* \| m_*) = H^S(\mathrm{pk}_\alpha \| R_* \| m_*)$, and

   $$(s_* - e_* H^{\mathrm{pk}}(\mathrm{pk}^\beta \| \mathrm{pk}^\alpha)) G = R_* + e_* \mathrm{pk} - H^{\mathrm{pk}}(\mathrm{pk}^\beta \| \mathrm{pk}^\alpha) G \qquad = R_* + e_* \mathrm{pk}^\alpha$$

   so that $(s_* - e_* H^{\mathrm{pk}}(\mathrm{pk}^\beta \| \mathrm{pk}^\alpha), R_*)$ is a valid Schnorr signature which is not equal to output of any signature query to $\mathscr{S}$.

$\square$