Discreet Log Contracts with Schnorr Adaptor Signatures

Jesse Posner

September 11, 2020

This document is a formal and detailed description of the pre-existing concept[1] of combining discreet log contracts[2] and Schnorr adaptor signatures[3].

1 Schnorr Signatures

A Schnorr signature scheme (G, S, V) uses the group of points $\mathbb{G}$ over a finite field $\mathbb{F}_p$. Let g be a generator of $\mathbb{G}$, let q be the prime order of the group $\mathbb{G}$, let $\mathcal{C}$ be the challenge space $\mathcal{C} \subseteq \mathbb{Z}_q$, and let H be a hash function $H : \mathcal{M} \times \mathbb{G} \to \mathcal{C}$. The scheme works as follows:

1. Key generation algorithm G:
   - (a) (private key) $\alpha \overset{\text{R}}{\leftarrow} \mathbb{Z}_q$
   - (b) (public key) $u \leftarrow g^{\alpha}$
2. Signature generation algorithm S:
   - (a) (message space) $m \in \mathcal{M}$
   - (b) (inputs) $S(\alpha, m)$
   - (c) (nonce) $\alpha_n \overset{\text{R}}{\leftarrow} \mathbb{Z}_q$
   - (d) (nonce commitment) $u_n \leftarrow g^{\alpha_n}$
   - (e) (Fiat-Shamir heuristic challenge) $c \leftarrow H(m, u_n)$
   - (f) (proof) $\alpha_z \leftarrow \alpha_n + \alpha c$
   - (g) (output) $\sigma := (u_n, \alpha_z)$
3. Signature verification algorithm V:
   - (a) (inputs) $V(\sigma, m, u)$

[1] `https://lists.launchpad.net/mimblewimble/msg00485.html`
[2] `https://adiabat.github.io/dlc.pdf`
[3] `http://diyhpl.us/wiki/transcripts/layer2-summit/2018/scriptless-scripts/`

(b) (Fiat-Shamir heuristic challenge) $c \leftarrow H(m, u_n)$

(c) (verification) $g^{\alpha_z} \overset{?}{=} u_n \cdot u^c$

$$\begin{aligned} g^{\alpha_z} &= u_n \cdot u^c && \Leftrightarrow \alpha_z = \alpha_n + \alpha c \\ &= g^{\alpha_n} \cdot (g^{\alpha})^c && \Leftrightarrow \\ &= g^{\alpha_n} \cdot g^{\alpha c} && \Leftrightarrow \end{aligned}$$

2 Aggregation

The linearity of Schnorr signatures allows for key and signature aggregation.

1. Key generation:

   (a) (Alice's private key) $\alpha_a \overset{\text{R}}{\leftarrow} \mathbb{Z}_q$

   (b) (Alice's public key) $u_a \leftarrow g^{\alpha_a}$

   (a) (Bob's private key) $\alpha_b \overset{\text{R}}{\leftarrow} \mathbb{Z}_q$

   (b) (Bob's public key) $u_b \leftarrow g^{\alpha_b}$

2. Key aggregation:

   (a) (aggregate public key) $\hat{u} \leftarrow u_a \cdot u_b$

$$\begin{aligned} \hat{u} &= u_a \cdot u_b \\ &= g^{\alpha_a} \cdot g^{\alpha_b} \\ &= g^{(\alpha_a + \alpha_b)} \\ &= g^{\hat{\alpha}} \end{aligned}$$

3. Nonce generation:

   (a) (Alice's nonce) $\alpha_{n_a} \overset{\text{R}}{\leftarrow} \mathbb{Z}_q$

   (b) (Alice's nonce commitment) $u_{n_a} \leftarrow g^{\alpha_{n_a}}$

   (a) (Bob's nonce) $\alpha_{n_b} \overset{\text{R}}{\leftarrow} \mathbb{Z}_q$

   (b) (Bob's nonce commitment) $u_{n_b} \leftarrow g^{\alpha_{n_b}}$

4. Nonce aggregation:

   (a) (aggregate nonce commitment) $\hat{u}_n \leftarrow u_{n_a} \cdot u_{n_b}$

$$\begin{aligned} \hat{u}_n &= u_{n_a} \cdot u_{n_b} \\ &= g^{\alpha_{n_a}} \cdot g^{\alpha_{n_b}} \\ &= g^{(\alpha_{n_a} + \alpha_{n_b})} \\ &= g^{\hat{\alpha}_n} \end{aligned}$$

5. Signature generation:
   (a) (Fiat-Shamir heuristic challenge) $\hat{c} \leftarrow H(m, \hat{u}_n)$
   (b) (Alice's proof) $\alpha_{z_a} \leftarrow \alpha_{n_a} + \alpha_a \hat{c}$
   (c) (Bob's proof) $\alpha_{z_b} \leftarrow \alpha_{n_b} + \alpha_b \hat{c}$
   (d) (aggregate proof) $\hat{\alpha}_z \leftarrow \alpha_{z_a} + \alpha_{z_b}$

$$\begin{aligned}
\hat{\alpha}_z &= \alpha_{z_a} + \alpha_{z_b} \\
&= (\alpha_{n_a} + \alpha_a \hat{c}) + (\alpha_{n_b} + \alpha_b \hat{c}) \\
&= (\alpha_{n_a} + \alpha_{n_b}) + (\alpha_a \hat{c} + \alpha_b \hat{c}) \\
&= \hat{\alpha}_n + (\alpha_a \hat{c} + \alpha_b \hat{c}) \\
&= \hat{\alpha}_n + \hat{c}(\alpha_a + \alpha_b) \\
&= \hat{\alpha}_n + \hat{\alpha}\hat{c}
\end{aligned}$$

   (e) (output) $\hat{\sigma} := (\hat{u}_n, \hat{\alpha_z})$

It should be noted that the above key aggregation protocol is insecure because it is vulnerable to a rogue key attack. MuSig[4] or MuSig-DN[5] should be used to defend against such an attack.

3 Discreet Log Contract

Suppose Alice and Bob want to bet on the outcome of the 2020 presidential election. They both trust an oracle Olivia to produce a signature with the message $m_{jb} \leftarrow sha256(\text{'}biden\text{'})$ or $m_{dt} \leftarrow sha256(\text{'}trump\text{'})$ depending on the winner, and to not produce a signature at all if there is no clear winner.

They will each lock up 5 BTC pending the outcome, and the winner will receive 10 BTC, and if there is no clear winner by December 31 they both will be refunded.

Olivia is not aware of Alice and Bob's bet, or anyone else's bet using her signature, but she knows that a signature could be useful for parties desiring to make such a bet. Olivia has spent considerable time and money building up a reputation as a reliable oracle, so she is strongly incentivized to act honestly.

Well in advance of the election, Olivia publishes her public key u_o and funds the address associated with the key with a considerable amount of BTC. She also publishes a nonce commitment u_{n_o} to be used only for bets on the election.

Alice and Bob can seperately compute the two possible Fiat-Shamir heuristic challenges $c_{o_{jb}} \leftarrow H(m_{jb}, u_{n_o})$ and $c_{o_{dt}} \leftarrow H(m_{dt}, u_{n_o})$ that Olivia will use in her signature. They can also seperately compute the commitments to the two possible signatures of Olivia: $u_{z_{o_{jb}}} \leftarrow u_{n_o} \cdot {u_o}^{c_{o_{jb}}}$ and $u_{z_{o_{dt}}} \leftarrow u_{n_o} \cdot {u_o}^{c_{o_{dt}}}$. Alice and Bob will be using these commitments to construct adaptor signatures.

[4]`https://eprint.iacr.org/2018/068.pdf`
[5]`https://eprint.iacr.org/2020/1057.pdf`

Next, Alice and Bob will construct a refund transaction that sends them each 5 BTC from the address associated with their aggregate key $\hat{u}$ with a timelock targeted for December 31, and sign the transaction using the process described in section 2. They can then proceed to fund the address with 5 BTC each.

Alice and Bob will construct a transaction that sends Alice 10 BTC if Biden wins and Bob 10 BTC if Trump wins and produce the messages $m_{tx_{jb}} \leftarrow sha256(tx_{jb})$ and $m_{tx_{dt}} \leftarrow sha256(tx_{dt})$. They will then each generate two nonces and their commitments $(u_{n_{a_{jb}}}, u_{n_{a_{dt}}})$ and $(u_{n_{b_{jb}}}, u_{n_{b_{dt}}})$, and carefully exchange the nonce commitments, as described in MuSig or MuSig-DN.

Alice and Bob will aggregate their nonce commitments, but they will also include the commitment to Olivia's corresponding signature: $\hat{u}_{n_{jb}} \leftarrow u_{n_{a_{jb}}} \cdot u_{n_{b_{jb}}} \cdot u_{z_{o_{jb}}}$ and $\hat{u}_{n_{dt}} \leftarrow u_{n_{a_{dt}}} \cdot u_{n_{b_{dt}}} \cdot u_{z_{o_{dt}}}$.

They can then seperately compute the two possible Fiat-Shamir heuristic challenges $c_{tx_{jb}} \leftarrow H(m_{tx_{jb}}, \hat{u}_{n_{jb}})$ and $c_{tx_{dt}} \leftarrow H(m_{tx_{dt}}, \hat{u}_{n_{dt}})$ that they will use in their adaptor signatures.

Alice will then proceed to produce an adaptor signature for each transaction: $\alpha'_{z_{a_{jb}}} \leftarrow \alpha_{n_{a_{jb}}} + c_{tx_{jb}}\alpha_a$ and $\alpha'_{z_{a_{dt}}} \leftarrow \alpha_{n_{a_{dt}}} + c_{tx_{dt}}\alpha_a$. And Bob will do the same: $\alpha'_{z_{b_{jb}}} \leftarrow \alpha_{n_{b_{jb}}} + c_{tx_{jb}}\alpha_b$ and $\alpha'_{z_{b_{dt}}} \leftarrow \alpha_{n_{b_{dt}}} + c_{tx_{dt}}\alpha_b$. They will then exchange adaptor signatures and aggregate them: $\alpha'_{z_{jb}} \leftarrow \alpha'_{z_{a_{jb}}} + \alpha'_{z_{b_{jb}}}$ and $\alpha'_{z_{dt}} \leftarrow \alpha'_{z_{a_{dt}}} + \alpha'_{z_{b_{dt}}}$.

Notice that these are not valid signatures because the challenges use nonce commitments that include Olivia's corresponding signature commitment, whereas the signatures themselves do not include Olivia's corresponding signature (which is serving as the missing component of the nonces in the adaptor signatures). If we were to add Olivia's corresponding signature, it would convert the adaptor signature into a valid signature by completing the nonce.

Lastly, Alice and Bob will fund the address, and wait for the election. If there is a clear outcome to the election, Olivia will produce either the signature $\alpha_{z_{o_{jb}}} \leftarrow \alpha_{n_o} + c_{o_{jb}}\alpha_o$ or $\alpha_{z_{o_{dt}}} \leftarrow \alpha_{n_o} + c_{o_{dt}}\alpha_o$. If $\alpha_{z_{o_{jb}}}$ is produced, then Alice will be able to collect her 10 BTC by adding it to the corresponding adaptor signature to produce a valid signature for the transaction tx_{jb} that sends her 10 BTC: $\alpha_{z_{jb}} \leftarrow \alpha'_{z_{jb}} + \alpha_{z_{o_{jb}}}$.

$$\begin{aligned}
\alpha_{z_{jb}} &= \alpha'_{z_{jb}} + \alpha_{z_{o_{jb}}} \\
&= (\alpha_{n_{a_{jb}}} + \alpha_{n_{b_{jb}}} + \hat{\alpha}c_{tx_{jb}}) + \alpha_{z_{o_{jb}}} \\
&= (\alpha_{n_{a_{jb}}} + \alpha_{n_{b_{jb}}} + \alpha_{z_{o_{jb}}}) + \hat{\alpha}c_{tx_{jb}} \\
&= \hat{\alpha}_{n_{jb}} + \hat{\alpha}c_{tx_{jb}}
\end{aligned}$$

On the other hand, if $\alpha_{z_{o_{dt}}}$ is produced, then Bob will be able to collect his 10 BTC by adding it to the corresponding adaptor signature to produce a valid signature for the transaction tx_{dt} that sends him 10 BTC: $\alpha_{z_{dt}} \leftarrow \alpha'_{z_{dt}} + \alpha_{z_{o_{dt}}}$.

$$
\begin{aligned}
\alpha_{z_{dt}} &= \alpha'_{z_{dt}} + \alpha_{z_{o_{dt}}} \\
&= (\alpha_{n_{a_{dt}}} + \alpha_{n_{b_{dt}}} + \hat{\alpha} c_{tx_{dt}}) + \alpha_{z_{o_{dt}}} \\
&= (\alpha_{n_{a_{dt}}} + \alpha_{n_{b_{dt}}} + \alpha_{z_{o_{dt}}}) + \hat{\alpha} c_{tx_{dt}} \\
&= \hat{\alpha}_{n_{dt}} + \hat{\alpha} c_{tx_{dt}}
\end{aligned}
$$

If Olivia decides to produce both signatures $\alpha_{z_{o_{jb}}}$ and $\alpha_{z_{o_{dt}}}$, then she will reveal her private key, lose her reputation, and risks losing access to the BTC associated with the corresponding address:

$$
\begin{aligned}
\alpha_{z_{o_{jb}}} - \alpha_{z_{o_{dt}}} &= (\alpha_{n_o} + \alpha_o c_{o_{jb}}) - (\alpha_{n_o} + \alpha_o c_{o_{dt}}) \\
&= \alpha_{n_o} + \alpha_o c_{o_{jb}} - \alpha_{n_o} - \alpha_o c_{o_{dt}} \\
&= \alpha_{n_o} - \alpha_{n_o} + \alpha_o c_{o_{jb}} - \alpha_o c_{o_{dt}} \\
&= \alpha_o c_{o_{jb}} - \alpha_o c_{o_{dt}} \\
&= \alpha_o (c_{o_{jb}} - c_{o_{dt}}) \\
\frac{(\alpha_{z_{o_{jb}}} - \alpha_{z_{o_{dt}}})}{(c_{o_{jb}} - c_{o_{dt}})} &= \alpha_o
\end{aligned}
$$