Multisig: Defeating Drijvers with Bi-Nonce Signing

koe `ukoe@protonmail.com`

October 14, 2021

Drijvers et. al in [3] discuss an attack on some multisignature schemes. They show how Wagner's generalization of the birthday problem [7] can allow signature forgeries in sub-exponential time if a multisig group containing a dishonest signer performs many concurrent signing attempts (at least 9 parallel attempts are required for an efficient attack, according to [1]).

The FROST signature scheme [4] introduced so-called 'bi-nonce signing' to efficiently and effectively defeat the Drijvers attack without increasing communication rounds between signers, compared to naive multisig. Previous schemes, such as MuSig [5], defeated Drijvers with commit-and-reveal patterns that add an extra round of communication to signing (this approach was recommended in MRL-0009 [6]).

In this technical note, I sketch out an intuition for Schnorr multisig, the Drijvers attack, and the two primary mitigations against Drijvers (bi-nonce signing and commit-and-reveal patterns). I only discuss N-of-N multisignatures, but all/most concepts can be extended to M-of-N thresholded multisig (M $\leq$ N).

This document is for EDUCATIONAL PURPOSES ONLY, and should NEVER EVER be used for a production implementation (refer to the cited materials instead).

Plain Schnorr

Here is a plain Schnorr signature scheme, with one signer (Alice).

Signature

Assume Alice has the private/public key pair (k_A, K_A). To unequivocally sign an arbitrary message $\mathfrak{m}$, she could execute the following steps:

1. Generate random number $\alpha \in_R \mathbb{Z}_l$, and compute αG.
2. Calculate the challenge using a cryptographically secure hash function, $c = \mathcal{H}(\mathfrak{m}, [\alpha G])$.
3. Define the response r such that $\alpha = r + c * k_A$. In other words, $r = \alpha - c * k_A$.
4. Publish the signature (c, r).

Verification

Any third party who knows the EC domain parameters (specifying which elliptic curve was used), the signature (c, r), the signing method, $\mathfrak{m}$, the hash function, and K_A can verify the signature:

1. Calculate the challenge: $c' = \mathcal{H}(\mathfrak{m}, [rG + c * K_A])$.
2. If $c = c'$, then the signature passes.

Naive Schnorr multisig

Here is a naive 2-round multisig Schnorr scheme between N signers (N-of-N). For simplicity, we use plain key aggregation to create the group key (the sum of signer keys). In a real implementation, you would use robust key aggregation (from [5]), FROST-style key generation (from [4, 6]), or key-share signing (from `SpeedyMuSig` in [2]).

Signature

Say there are N people who each have a public key in the set $\mathbb{K}^{pre}$, where each person $e \in \{1, ..., N\}$ knows the private key k_e^{pre}. Their N-of-N group public key, which they will use to sign messages, is $K^{grp} = \sum_e k_e^{pre} G$. Suppose they want to jointly sign a message $\mathfrak{m}$. They could collaborate on a basic Schnorr-like signature like this:

1. **Round 1**: Each participant $e \in \{1, ..., N\}$ does the following.
   (a) picks random nonce $\alpha_e \in_R \mathbb{Z}_l$,
   (b) computes $\alpha_e G$ and sends it to the other participants securely.
2. Each participant computes
$$\alpha G = \sum_e \alpha_e G$$
3. **Round 2**: Each participant $e \in \{1, ..., N\}$ does the following.[1]
   (a) computes the challenge $c = \mathcal{H}_n(\mathfrak{m}, [\alpha G])$,
   (b) defines their response component $r_e = \alpha_e - c * k_e^{pre} \pmod{l}$,
   (c) and sends r_e to the other participants securely.
4. Each participant computes
$$r = \sum_e r_e$$
5. Any participant can publish the signature $\sigma(\mathfrak{m}) = (c, r)$.

Note that, semantically, a round 'ends' when participants have collected all messages produced and sent out by other participants during that round.

Verification

Given K^{grp}, $\mathfrak{m}$, and $\sigma(\mathfrak{m}) = (c, r)$:

1. Compute the challenge $c' = \mathcal{H}_n(\mathfrak{m}, [rG + c * K^{grp}])$.
2. If $c = c'$ then the signature is legitimate except with negligible probability.

[1] Note that a universal requirement of multisig schemes is to never reuse α_e for different challenges c.

The Drijvers attack

The naive Schnorr multisig scheme just described is vulnerable to the Drijvers attack. Suppose there are $j \in \{1, ..., T\}$ concurrent signing attempts (for different messages $\mathfrak{m}_j$) by the same multisig group. For the sake of notation, suppose signer $e = N$ is dishonest and executes the Drijvers attack.

1. **Round 1 (all)**: Each honest participant $e \in \{1, ..., N-1\}$ does the following for each concurrent signature j.
   (a) picks random nonce $\alpha_{j,e} \in_R \mathbb{Z}_l$,
   (b) computes $\alpha_{j,e}G$ and sends it to the other participants securely.
2. After collecting $\alpha_{j,e}G$ from all other participants, dishonest participant $e = N$ prepares for his attack with the following.
   (a) He picks a random nonce $\alpha' \in_R \mathbb{Z}_l$ and computes $\alpha' G$.
   (b) He creates $w \in \{1, ..., W\}$ new messages $\mathfrak{m}_w$.
   (c) He creates W new malicious challenges
   $$c_w^{fake} = \mathcal{H}_n(\mathfrak{m}_w, [\sum_{e=1}^{N-1}\sum_{j=1}^{T} \alpha_{e,j}G + \alpha' G])$$
3. Dishonest participant $e = N$ executes the Drijvers attack.
   (a) Create, but do not define, a set of EC points A_j for $j \in \{1, ..., T\}$.
   (b) Use an ROS solver (e.g. Wagner) to find a combination of points A_j such that $\sum_j c_j$ equals one of the fake challenges c_w^{fake}, by iteratively re-defining different A_j values in the following.
   $$c_j = \mathcal{H}_n(\mathfrak{m}_j, [\sum_{e=1}^{N-1} \alpha_{e,j}G + A_j])$$
   (c) Once he finds a successful challenge c_s^{fake}, he sends all A_j to the other participants.
4. Each honest participant computes
   $$\alpha_j G = \sum_{e=1}^{N-1} \alpha_{j,e}G + A_j$$
   Note that honest participants won't be able to distinguish dishonest A_j from honest values $\alpha_{j,e}G$. If the signature scheme requires signers to make a signature on $\alpha_{j,e}G$, then in the Drijvers attack the attacker would iteratively define a_j, compute $a_j G = A_j$ for the c_j computation, then send A_j to other participants (making the attack a bit less efficient, but still effective).
5. **Round 2 (all)**: Each honest participant $e \in \{1, ..., N-1\}$ does the following for each concurrent signature j:

(a) computes the challenge $c_j = \mathcal{H}_n(\mathfrak{m}, [\alpha_j G])$,

(b) defines their response component $r_{j,e} = \alpha_{j,e} - c_j * k_e^{pre} \pmod{l}$,

(c) and sends $r_{j,e}$ to the other participants securely.

6. After collecting $r_{j,e}G$ from all other participants, dishonest participant $e = N$ completes their forgery.

   (a) He computes his response $r' = \alpha' - c_s^{fake} * k_N^{pre} \pmod{l}$.

   (b) He computes the total forged response
$$r^{fake} = \sum_{e=1}^{N-1} \sum_{j=1}^{T} r_{e,j} + r'$$

7. The dishonest participant publishes their forgery $\sigma_{forged}(\mathfrak{m}_s) = (c_s^{fake}, r^{fake})$.

Verification

Given K^{grp}, $\mathfrak{m}_{\mathfrak{s}}$, and $\sigma_{forged}(\mathfrak{m}_s) = (c_s^{fake}, r^{fake})$:

1. Compute the challenge $c' = \mathcal{H}_n(\mathfrak{m}_{\mathfrak{s}}, [r^{fake}G + c_s^{fake} * K^{grp}])$.
2. If $c_s^{fake} = c'$ then the signature is considered valid (even though it is a forgery!).

Why it works

This works because
$$\begin{aligned}
c' &= c_s^{fake} \\
\mathcal{H}_n(\mathfrak{m}_j, [r^{fake}G + c_s^{fake}K^{grp}]) &= \mathcal{H}_n(\mathfrak{m}_w, [\sum_{e=1}^{N-1} \sum_{j=1}^{T} \alpha_{e,j}G + \alpha' G]) \\
&= \mathcal{H}_n(\mathfrak{m}_w, [\sum_{e=1}^{N-1} \sum_{j=1}^{T} (r_{j,e} + c_j k_e^{pre}) * G + (r' + c_s^{fake} k_N^{pre}) * G]) \\
&= \mathcal{H}_n(\mathfrak{m}_w, [(\sum_{e=1}^{N-1} \sum_{j=1}^{T} r_{j,e} + c_s^{fake} * \sum_{e=1}^{N-1} k_e^{pre}) * G + (r' + c_s^{fake} k_N^{pre}) * G]) \\
&= \mathcal{H}_n(\mathfrak{m}_w, [(\sum_{e=1}^{N-1} \sum_{j=1}^{T} r_{j,e} + r') * G + c_s^{fake} K^{grp}]) \\
&= \mathcal{H}_n(\mathfrak{m}_w, [r^{fake}G + c_s^{fake} K^{grp}])
\end{aligned}$$

Mitigating the Drijvers attack

The key to executing a Drijvers attack is being able to re-define A_j values many times without affecting any c_w^{fake} challenges. This way, with e.g. Wagner's method, it is possible to find a sum of challenges $\sum_j c_j$ that equals c_w^{fake}. If changing A_j also changes c_w^{fake}, then Wagner's method becomes useless.

Mitigation 1: commit-and-reveal

One way to prevent the flexibility of A_j is to add a commit-and-reveal step to signing.

Signature

Say there are N people who each have a public key in the set $\mathbb{K}^{pre}$, where each person $e \in \{1, ..., N\}$ knows the private key k_e^{pre}. Their N-of-N group public key, which they will use to sign messages, is $K^{grp} = \sum_e k_e^{pre} G$. Suppose they want to jointly sign a message $\mathfrak{m}$. They could collaborate on a basic Schnorr-like signature like this:

1. **Round 1**: Each participant $e \in \{1, ..., N\}$ does the following.
   (a) picks random nonce $\alpha_e \in_R \mathbb{Z}_l$,
   (b) computes $\alpha_e G$
   (c) commits to it with $C_e^\alpha = \mathcal{H}_n(\alpha_e G)$,
   (d) and sends C_e^α to the other participants securely.
2. **Round 2**: Once all commitments C_e^α have been collected, each participant sends their $\alpha_e G$ to the other participants securely. They must verify that $C_e^\alpha \stackrel{?}{=} \mathcal{H}_n(\alpha_e G)$ for all other participants.
3. Each participant computes
$$\alpha G = \sum_e \alpha_e G$$
4. **Round 3**: Each participant $e \in \{1, ..., N\}$ does the following:
   (a) computes the challenge $c = \mathcal{H}_n(\mathfrak{m}, [\alpha G])$,
   (b) defines their response component $r_e = \alpha_e - c * k_e^{pre} \pmod{l}$,
   (c) and sends r_e to the other participants securely.
5. Each participant computes
$$r = \sum_e r_e$$

6. Any participant can publish the signature $\sigma(\mathfrak{m}) = (c, r)$.

Now, if an attacker tried to execute a Drijvers attack, they have a problem. They can't learn $\alpha_{j,e}$ until after sending $C^{\alpha}_{j,N} = \mathcal{H}_n(A_j)$ to other participants. To re-define A_j, they would need to restart signing from the beginning. However, that would entail new $\alpha_{j,e}$ values from all participants, which would also entail new c^{fake}_w challenges. Therefore the Drijvers attack is mitigated.

Mitigation 2: bi-nonce signing

Bi-nonce signing has a similar effect to the commit-and-reveal pattern, but requires only two rounds thanks to a neat trick with the random oracle model.

Signature

Say there are N people who each have a public key in the set $\mathbb{K}^{pre}$, where each person $e \in \{1, ..., N\}$ knows the private key k^{pre}_e. Their N-of-N group public key, which they will use to sign messages, is $K^{grp} = \sum_e k^{pre}_e G$. Suppose they want to jointly sign a message $\mathfrak{m}$. They could collaborate on a basic Schnorr-like signature like this:

1. **Round 1**: Each participant $e \in \{1, ..., N\}$ does the following.
   (a) picks random nonces $\alpha^a_e, \alpha^b_e \in_R \mathbb{Z}_l$,
   (b) computes $\alpha^a_e G, \alpha^b_e G$ and sends them to the other participants securely.
2. Each participant
   (a) Computes nonce coefficients n_e for $e \in \{1, ..., N\}$
   $$n_e = \mathcal{H}_n(e, \mathfrak{m}, [\alpha^a_1 G], [\alpha^b_1 G], ..., [\alpha^a_N G], [\alpha^b_N G])$$
   (b) Computes
   $$\alpha G = \sum_e [\alpha^a_e G + n_e * \alpha^b_e G]$$
3. **Round 2**: Each participant $e \in \{1, ..., N\}$ does the following.
   (a) computes the challenge $c = \mathcal{H}_n(\mathfrak{m}, [\alpha G])$,
   (b) defines their response component $r_e = (\alpha^a_e + n_e \alpha^b_e) - c * k^{pre}_e \pmod{l}$,
   (c) and sends r_e to the other participants securely.
4. Each participant computes
   $$r = \sum_e r_e$$
5. Any participant can publish the signature $\sigma(\mathfrak{m}) = (c, r)$.

In this case, if a Drijvers attacker redefines A^a_j or A^b_j, then all the nonces $n_{j,e}$ will change, thereby changing the c^{fake}_w challenges. Therefore the Drijvers attack is mitigated.

Why two nonces?

It may seems like setting $\alpha_e^a = 0$ would be acceptable, only transmitting $\alpha_e^b G$ to other signers. However, doing so would allow the Drijvers attacker to 'cancel' out the nonce coefficients of honest signers.

Suppose there are $j \in \{1, ..., T\}$ concurrent signing attempts (for different messages $\mathfrak{m}_j$) by the same multisig group. For the sake of notation, suppose signer $e = N$ is dishonest and executes the Drijvers attack. Also suppose $N = 2$ (or, equivalently, assume the dishonest participant controls $N - 1$ of the key shares). It isn't clear to me if the problem I demonstrate below is also a problem if the dishonest signer controls only $N - 2$ key shares, but since honest signers must assume $N - 1$ co-signers are malicious, this problem is sufficient to debunk the $\alpha_e^a = 0$ simplification.

I don't show computations of nonce coefficients n_e, which are straightforward.

1. **Round 1 (all)**: The honest participant does the following for each concurrent signature j.
   (a) picks random nonces $\alpha_{j,1}^b \in_R \mathbb{Z}_l$,
   (b) computes $\alpha_{j,1}^b G$ and sends them to the dishonest participant.
2. After collecting $\alpha_{j,1}^b G$, the dishonest participant prepares for his attack with the following.
   (a) He picks a random nonce $\alpha' \in_R \mathbb{Z}_l$ and computes $\alpha' G$.
   (b) He creates $w \in \{1, ..., W\}$ new messages $\mathfrak{m}_w$.
   (c) He creates W new malicious challenges
   $$c_w^{fake} = \mathcal{H}_n(\mathfrak{m}_w, [\alpha_{j,1}^b G + \alpha' G])$$
3. The dishonest participant executes the Drijvers attack.
   (a) Create, but do not define, a set of EC points A_j for $j \in \{1, ..., T\}$.
   (b) Use an ROS solver (e.g. Wagner) to find a combination of points A_j such that $\sum_j (1/n_{j,1}) * c_j$ equals one of the fake challenges c_w^{fake}, by iteratively re-defining different A_j values in the following.
   $$c_j = \mathcal{H}_n(\mathfrak{m}_j, [n_{j,1} * \alpha_{j,1}^b G + n_{j,2} * A_j])$$
   (c) Once he finds a successful challenge c_s^{fake}, he sends all A_j to the honest participant.
4. The honest participant computes
   $$\alpha_j G = n_{j,1} * \alpha_{j,1} G + n_{j,2} * A_j$$
5. **Round 2 (all)**: The honest participant does the following for each concurrent signature j:
   (a) computes the challenge $c_j = \mathcal{H}_n(\mathfrak{m}, [\alpha_j G])$,
   (b) defines their response component $r_{j,1} = n_{j,1} * \alpha_{j,1} - c_j * k_1^{pre} \pmod{l}$,
   (c) and sends $r_{j,1}$ to the dishonest participant securely.

6. After collecting $r_{j,1}G$ from the honest participant, the dishonest participant completes their forgery.
   (a) He computes his response $r' = \alpha' - c_s^{fake} * k_N^{pre} \pmod{l}$.
   (b) He computes the total forged response
   $$r^{fake} = (1/n_{j,1}) * r_{j,1} + r'$$
7. The dishonest participant publishes their forgery $\sigma_{forged}(\mathfrak{m}_s) = (c_s^{fake}, r^{fake})$.

This works because

$$c' = c_s^{fake}$$

$$\mathcal{H}_n(\mathfrak{m}_j, [r^{fake}G + c_s^{fake}K^{grp}]) = \mathcal{H}_n(\mathfrak{m}_w, [\sum_{j=1}^{T} \alpha_{j,1}^b G + \alpha' G])$$

$$\mathcal{H}_n(\mathfrak{m}_j, [(\sum_{j=1}^{T}(1/n_{j,1}) * r_{j,1} + r')G + c_s^{fake}K^{grp}]) = \mathcal{H}_n(\mathfrak{m}_w, [\sum_{j=1}^{T} \alpha_{j,1}^b G + \alpha' G])$$

$$\mathcal{H}_n(\mathfrak{m}_j, [(\sum_{j=1}^{T}(1/n_{j,1}) * (n_{j,1} * \alpha_{j,1}^b - c_j * k_1^{pre}) + \alpha' - c_s^{fake} * k_N^{pre})G + c_s^{fake}K^{grp}]) = \mathcal{H}_n(\mathfrak{m}_w, [\sum_{j=1}^{T} \alpha_{j,1}^b G + \alpha' G])$$

$$\mathcal{H}_n(\mathfrak{m}_j, [(\sum_{j=1}^{T}(\alpha_{j,1}^b - (1/n_{j,1}) * c_j * k_1^{pre}) + \alpha' - c_s^{fake} * k_N^{pre})G + c_s^{fake}K^{grp}]) = \mathcal{H}_n(\mathfrak{m}_w, [\sum_{j=1}^{T} \alpha_{j,1}^b G + \alpha' G])$$

$$\mathcal{H}_n(\mathfrak{m}_j, [(\sum_{j=1}^{T} \alpha_{j,1}^b + \alpha' - (\sum_{j=1}^{T}(1/n_{j,1}) * c_j * k_1^{pre} + c_s^{fake} * k_N^{pre}))G + c_s^{fake}K^{grp}]) = \mathcal{H}_n(\mathfrak{m}_w, [\sum_{j=1}^{T} \alpha_{j,1}^b G + \alpha' G])$$

$$\mathcal{H}_n(\mathfrak{m}_j, [(\sum_{j=1}^{T} \alpha_{j,1}^b + \alpha' - (c_s^{fake} * k_1^{pre} + c_s^{fake} * k_N^{pre}))G + c_s^{fake}K^{grp}]) = \mathcal{H}_n(\mathfrak{m}_w, [\sum_{j=1}^{T} \alpha_{j,1}^b G + \alpha' G])$$

$$\mathcal{H}_n(\mathfrak{m}_w, [\sum_{j=1}^{T} \alpha_{j,1}^b G + \alpha' G]) = \mathcal{H}_n(\mathfrak{m}_w, [\sum_{j=1}^{T} \alpha_{j,1}^b G + \alpha' G])$$

With two nonces, $r^{fake} = (1/n_{j,1}) * r_{j,1} + r' = (1/n_{j,1}) * \alpha_{j,1}^a + \alpha_{j,1}^b - (1/n_{j,1}) * c_j * k_1^{pre} + r'$. Since the nonce coefficient is still prefixed on $\alpha_{j,1}^a$, it is implicitly present in c_w^{fake}, hence c_w^{fake} is dependent on A_j (A_j^a and A_j^b in the case of two nonces), so the Drijvers attack is mitigated.

Bibliography

[1] Fabrice Benhamouda, Tancrède Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova. On the (in)security of ros. Cryptology ePrint Archive, Report 2020/945, 2020. https://ia.cr/2020/945 [Online; accessed 10/14/2021].

[2] Elizabeth Crites, Chelsea Komlo, and Mary Maller. How to prove schnorr assuming schnorr: Security of multi- and threshold signatures. Cryptology ePrint Archive, Report 2021/1375, 2021. https://ia.cr/2021/1375 [Online; accessed 10/14/2021].

[3] Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igors Stepanovs. On the Security of Two-Round Multi-Signatures. Cryptology ePrint Archive, Report 2018/417, 2018. https://eprint.iacr.org/2018/417 [Online; accessed 10/14/2021].

[4] Chelsea Komlo and Ian Goldberg. Frost: Flexible round-optimized schnorr threshold signatures. Cryptology ePrint Archive, Report 2020/852, 2020. https://ia.cr/2020/852 [Online; accessed 10/14/2021].

[5] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple Schnorr Multi-Signatures with Applications to Bitcoin. May 2018. https://eprint.iacr.org/2018/068.pdf [Online; accessed 03/01/2020].

[6] Shen Noether and Sarang Noether. Thring Signatures and their Applications to Spender-Ambiguous Digital Currencies, MRL-0009, November 2018. https://web.getmonero.org/resources/research-lab/pubs/MRL-0009.pdf [Online; accessed 01/15/2020].

[7] David Wagner. A Generalized Birthday Problem. In Moti Yung, editor, *Advances in Cryptology — CRYPTO 2002*, pages 288–304, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. https://link.springer.com/content/pdf/10.1007%2F3-540-45708-9_19.pdf [Online; accessed 02/07/2020].