SUPPORTING INFORMATION

Interrogation of Electrochemical Aptamer-Based Sensors via Peak-to-Peak Separation in Cyclic Voltammetry Improves Temporal Stability and Batch-To-Batch Variability in Biological Fluids

Miguel Aller Pellitero[α], Samuel D. Curtis[α] and Netzahualcóyotl Arroyo-Currás[α,β,γ,*]

Affiliations

[α] Department of Pharmacology and Molecular Sciences, Johns Hopkins University School of Medicine, Baltimore, MD 21202.

[β] Department of Chemical and Biomolecular Engineering, Whiting School of Engineering, Johns Hopkins University, Baltimore, MD 21218.

[γ] Institute for Nanobiotechnology, Johns Hopkins University, Baltimore, MD 21218.

* **Correspondence to:**

Netz Arroyo, Ph.D.

Johns Hopkins University School of Medicine

316 Hunterian Building

725 North Wolfe Street

Baltimore, MD 21205

netzarroyo@jhmi.edu

443-287-4798

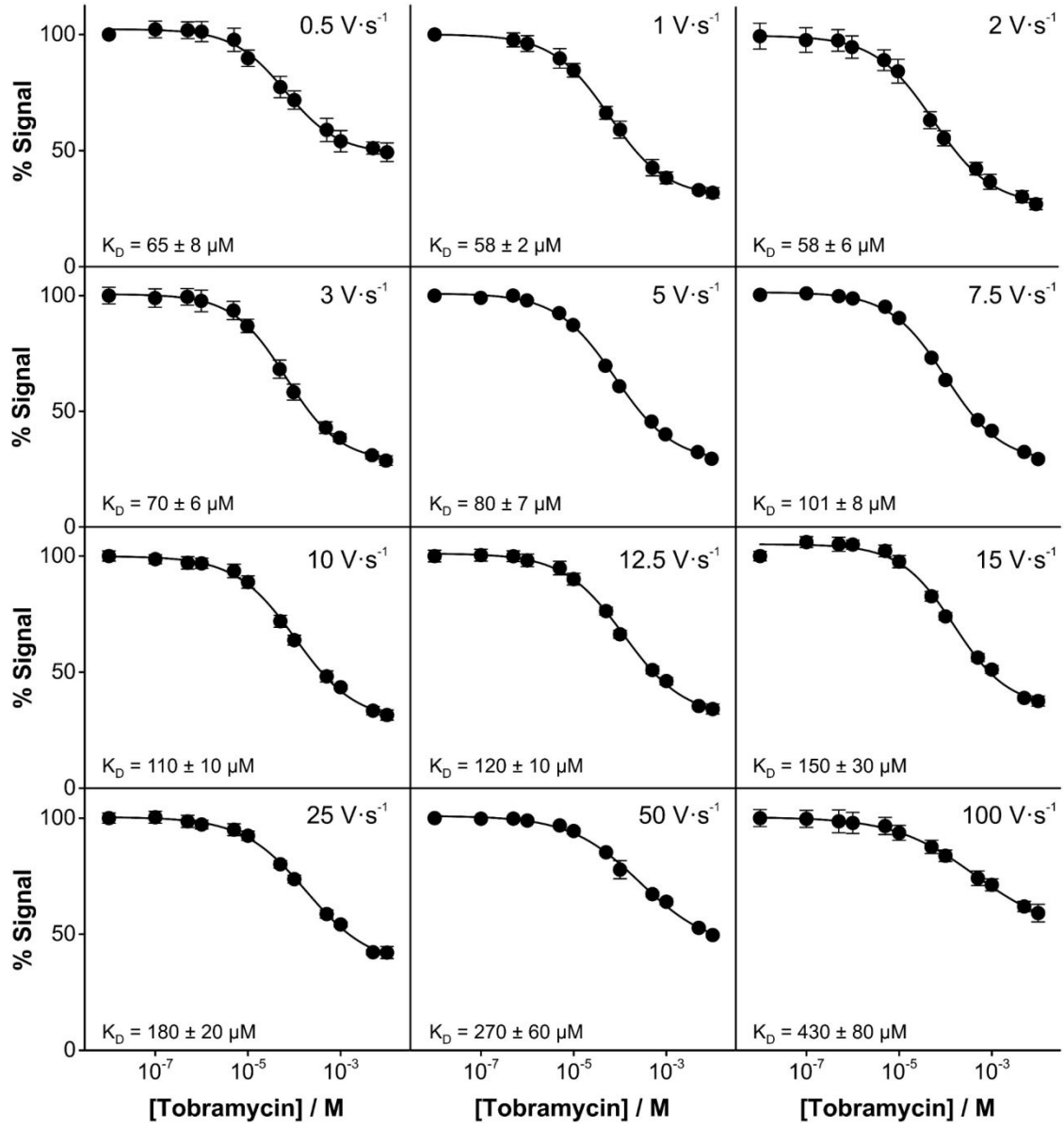

Figure S1. Effect of CV scan rate on the shape of E-AB sensor dose-response curves. Here we illustrate this effect using tobramycin-binding E-AB sensors in phosphate-buffered saline, interrogated at voltage scanning rates ranging from 0.5 to 100 $V \cdot s^{-1}$. Solid lines correspond to non-linear regression analyses using the Hill isotherm (at fixed n = 1). Error bars represent the standard deviation between 5 sensors.

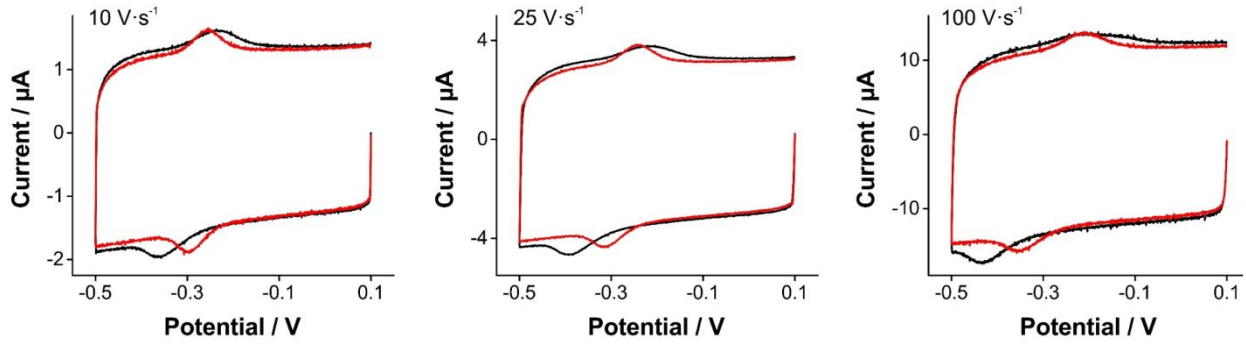

Figure S2. The peak broadening observed at voltage scanning rates faster than 10 V·s can hinder the accurate readout of peak potential.

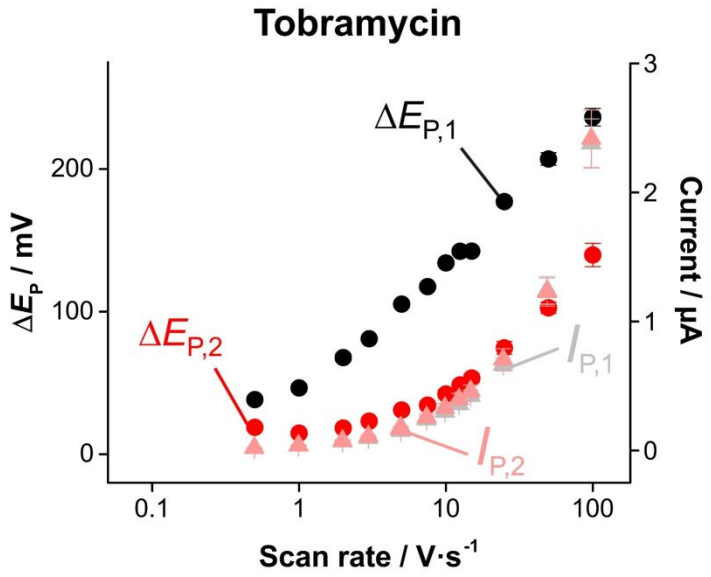

Figure S3. Comparison of signal change using I_P or ΔE_P as analytical parameters. The increase in electron transfer rate between MB and the electrode surface upon tobramycin addition (10 mM) is reflected by a decrease in peak-to-peak separation of cyclic voltammograms ($\Delta E_{P,1}$ and $\Delta E_{P,2}$ for the unbound and bound states, respectively). However, this variation in electron transfer rate is minimally reflected as a change in peak current, observing only a slight increase at high scan rates ($I_{P,1}$ and $I_{P,2}$ for the unbound and bound states, respectively). Error bars represent the standard deviation between 5 sensors.

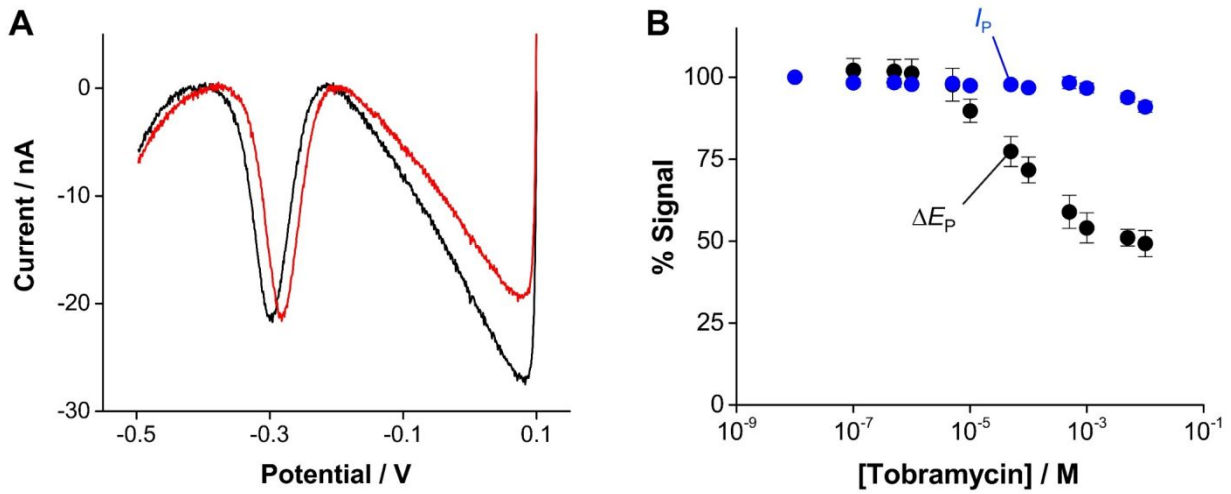

Figure S4. Changes in I_P and ΔE_P at slow rates. (A) The interrogation of tobramycin-binding E-AB sensors at a low scan rate (1 $V \cdot s^{-1}$) leads to no changes in peak current upon addition of a saturating tobramycin concentrations (10mM). (B) Thus, the resulting titration curve is flat when I_P is used as the analytical parameter. In contrast, when we use ΔE_P, a sigmoidal titration curve is obtained, reflecting the suitability of the ΔE_P-based method to interrogate E-AB sensors even at such slow voltage scanning rates. Error bars represent the standard deviation between 5 electrodes.

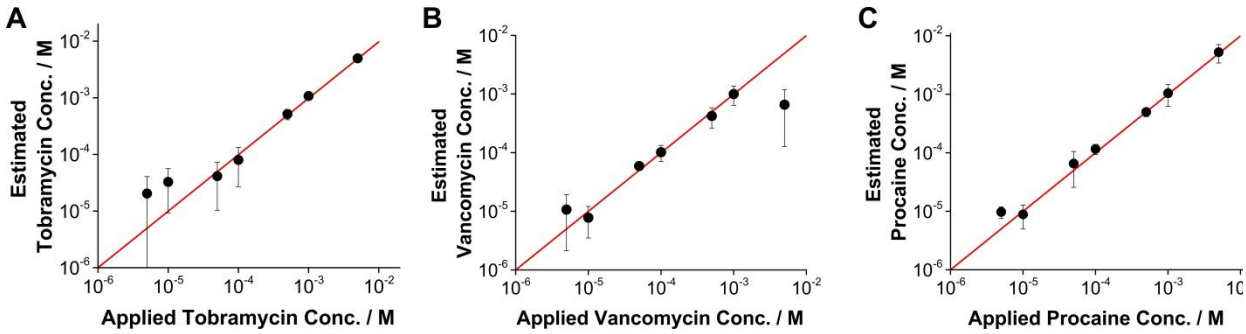

Figure S5. We evaluated the precision of the ΔE_P-based interrogation method by comparing estimated vs added target concentration from the calibration curves in Figure 4. The asymmetric error bars are due to the log-log plots used.

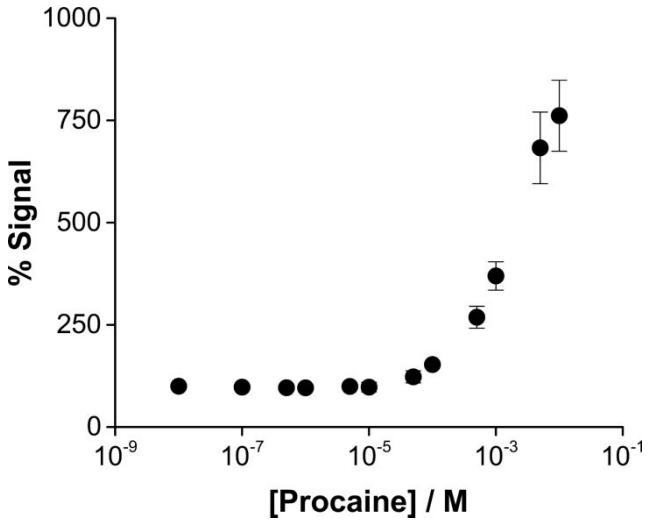

Figure S6. Calibration curve for procaine-binding E-AB sensors obtained by square wave voltammetry. The Hill's coefficient and dissociation constant obtained were 0.97 ± 0.05 and 1.9 ± 0.2 mM, respectively. Each point represents the average of 5· measurements at a frequency of 100 Hz, amplitude of 50 mV, and step potential of 1 mV. The estimated LOD for the SWV-based interrogation is 60 ± 10 µM vs 90 ± 10 µM for CV-based interrogation.

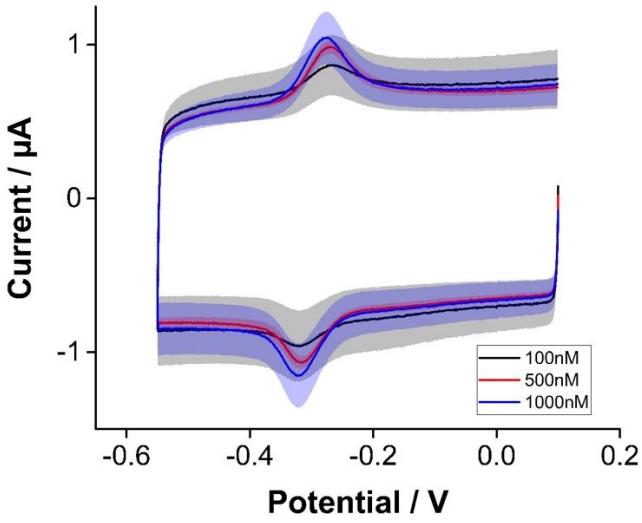

Figure S7. Cyclic voltammograms measured on tobramycin sensors functionalized from different aptamer deposition concentrations. Each voltammogram represents the average of 4 electrodes at a scan rate of 5 $V \cdot s^{-1}$, in the presence of 1mM tobramycin. Shaded areas represent the standard deviation between 4 sensors of each batch.

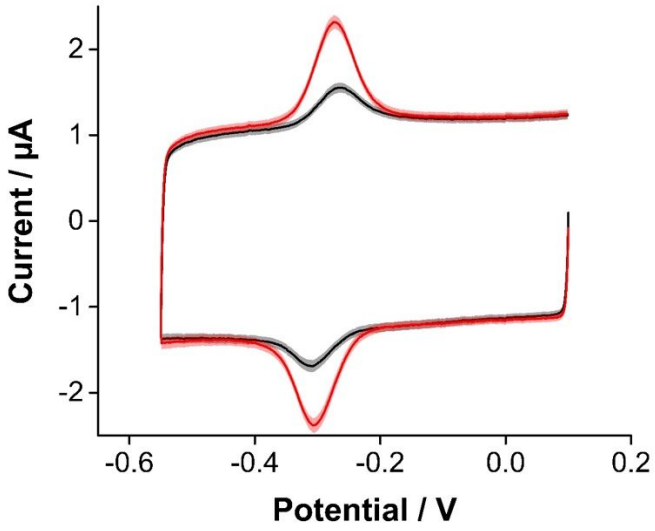

Figure S8. Cyclic voltammograms obtained using a co-immobilization (black line) or a backfilling protocol (red line). Each voltammogram represents the average of 4 electrodes at a scan rate of 5 $V \cdot s^{-1}$, in the presence of 1mM tobramycin. Shaded areas represent the standard deviation between 4 sensors of each batch.

Table S1. Electrochemical parameters calculated from each sensor set in Figure S7.

[Aptamer] / nM	Surface Coverage / pmol cm^{-2}	ΔE_P No Target / mV	ΔE_P Target / mV	I_P (CV) / µA
100	0.5 ± 0.1	118 ± 3	52 ± 1	0.16 ± 0.02
500	0.9 ± 0.1	112 ± 1	47 ± 3	0.33 ± 0.03
1000	1 ± 0.1	108 ± 1	46 ± 1	0.38 ± 0.06

Table S2. Electrochemical parameters calculated from each sensor set in Figure S8.

	Surface Coverage / pmol cm^{-2}	ΔE_P / mV	% Initial signal at 1mM tobramycin
Backfilling	2.5 ± 0.1	33 ± 1	48 ± 1
Co-immobilization	0.9 ± 0.1	47 ± 2	43 ± 1

Outline of the SACMES Peak-to-Peak Separation Algorithm

```
PeakPotentials = {}
for segment in ['forward','reverse']:
    # Read the Cyclic Voltammogram
    currents, potentials = ReadData(current_file)
    # Smooth the data using savitzky-golay smoothing
    smooth_currents = savgol_filter(currents)
    # Correlate currents with their respective potential
    data_dictionary = dict(zip(smooth_currents,potentials))
    # Create a linear baseline under the peak
    # baseline_dictionary = {baseline_potential;baseline_current}
    baseline_dictionary = create_baseline(data_dictionary)
    # Determine the PeakHeight of the segment depending on its orientation
    orientation = get_orientation(segment)
    if orientation == 'Cathodic':
        PeakCurrent = max(smooth_currents)
    elif orientation == 'Anodic':
        PeakCurrent = min(smooth_currents)
    # Match the peak current with its potential
    PeakPotential = data_dictionary[PeakCurrent]
    # Save the peak potential for peak separation calculation
    PeakPotentials[segment] = PeakPotential
    # Calculate the corresponding baseline current for signal gain calculation
    BaselineCurrent =  baseline_dictionary[PeakPotential]
    # Calculate the signal gain
    Signal = abs(PeakCurrent - BaselineCurrent)
# Calculate the separation between the two peak potentials
PeakSeparation = abs(PeakPotentials['forward'] - PeakPotentials['reverse'])
```

Outline of SACMES recursive analysis loop

```
# Iterate through each file
For file in file_limit:
    # Recursive loop searching for the current file
    while True:
        # retrieve the size of the file
        mydata_bytes = os.path.getsize(file)
        # if the size is greater than the threshold for analysis, analyze the file
        if mydata_bytes > args.threshold
            # Get the data from a generator class
            framedata = self.generator(file)
            # Visualize the data
            self._draw_next_frame(framedata)
```

```
# If the data hasn't been generated by the instrument, keep waiting
else:
        pass
# Check to see if the user has changed parameters for data analysis
self.check_input()
```