An Attack on the ECDSA-variant of BBS#'s Secure Element Protocol

Eysa Lee

August 6, 2024

1 Introduction

A variant of BBS signatures known as BBS# [bbs] has recently been brought up in conversations around the EU Digital Identity Wallet. One of the desired properties of the EU Digital Identity Wallet is to prevent users from sharing credentials with others, and the current desired mitigation strategy is to bind credential presentations with hardware. BBS# proposes techniques for efficiently integrating signatures issued by secure elements with credential presentations. Unfortunately, their protocol using ECDSA do not prevent a user from sharing their credential in a way that prevents others from presenting the credential without possession of the user's hardware.

2 The Attack

We recall the relevant protocol of [bbs] which uses ECDSA. Note that in order to prevent linkability, this protocol ultimately produce signatures under a version of the hardware's public key that is *blinded* by a multiplicative factor.

Let pk be the public key of the hardware with corresponding secret key sk (i.e., $\mathsf{pk} = g^{\mathsf{sk}}$), m be any message (e.g., a nonce sent by the verifier), and H be the hash function used by ECDSA. Their protocol works as follows.

Protocol [bbs, Figure 7]:

1. User first samples $r \leftarrow \mathbb{Z}_p^*$.
2. User computes $M = \frac{1}{r} \cdot H(m)$ and requests a signature on M from the secure element.
3. The secure element produces a signature on M as follows:
   (a) Samples $k \leftarrow \mathbb{Z}_p^*$
   (b) Compute $K = g^k = (x, y)$
   (c) Compute $\sigma' = \frac{1}{k} \cdot (M + \mathsf{sk} \cdot x)$
   (d) Output the signature on M as (σ', x)
4. User computes the blinded public key as $\mathsf{pk}_{\mathsf{blind}} = \mathsf{pk}^r$
5. User computes $\sigma = r \cdot \sigma'$
6. User outputs the blinded public key $\mathsf{pk}_{\mathsf{blind}}$ and corresponding signature on m as (σ, x)

Notice that with (σ, x),

$$\begin{aligned}\sigma &= r \cdot \frac{1}{k} \cdot (M + \mathsf{sk} \cdot x) \\ &= r \cdot \frac{1}{k} \cdot \left(\frac{1}{r} \cdot H(m) + \mathsf{sk} \cdot x\right) \\ &= \frac{1}{k} \cdot (H(m) + r \cdot \mathsf{sk} \cdot x)\end{aligned}$$

which makes it a valid signature under the blinded public key $\mathsf{pk}_{\mathsf{blind}} = \mathsf{pk}^r = g^{\mathsf{sk} \cdot r}$.

Producing multiple signatures from one. Unfortunately, once a user receives (σ', x) for any m, they can produce any number of additional signatures without requiring interaction with the secure element. The attack works as follows.

Let $m_2 \neq m$ and $r_2 = \frac{r}{H(m)} \cdot H(m_2)$. Compute

$$\begin{aligned}\sigma_2 &= r_2 \cdot \sigma' \\ &= \frac{r}{H(m)} \cdot H(m_2) \cdot \frac{1}{k} \cdot \left(\frac{1}{r} \cdot H(m) + \mathsf{sk} \cdot x\right) \\ &= \frac{1}{k} \cdot \left(H(m_2) + \frac{r}{H(m)} \cdot H(m_2) \cdot \mathsf{sk} \cdot x\right)\end{aligned}$$

Notice (σ_2, x) is a valid signature on m_2 under pk^{r_2}. Therefore, a user who wishes to share their credential only needs to share these few extra components in order for an unauthorized second user to produce additional signatures that will verify under blinded versions of the original user's hardware's public key. This unauthorized second user does not require access to the user's device, rendering this protocol moot.

References

[bbs] The BBS# protocol. URL: `https://github.com/user-attachments/files/15905230/BBS_Sharp_Short_TR.pdf`, Accessed 2024/06/24.