Generic Legacy Signature w/ EC keys

\( \text{private key } x \) → \( X \)
\( \text{public key } x \cdot g \)

Transaction \( tx \) to be signed

\[ X = x \cdot G \]

\[ c \cdot H(tx, r \cdot G) \]

\[ c \cdot H(tx, r \cdot G) \]

Non-interactive (Flattened Shamir) Schnorr

\[ x \cdot G \]

\( \text{random and unique for each signature, otherwise privacy could be leaked from response } r \)

\[ x = \alpha \]

\[ \text{called challenge because it's known to the signer only after knowing } x \text{ (if not, in that case the signer could be asked to prove ownership of a } \]

\[ \text{random } \alpha \text{ and reveal } \alpha \cdot G \text{ (to specify the output of a one-way hash)} \text{, as in interactive Schnorr proof where it's provided by the verifier only after knowing } x \text{ (if not, in that case the signer could be asked to prove ownership of a } \]

\[ \text{random } \alpha \text{ and reveal } \alpha \cdot G \text{ (to specify the output of a one-way hash)} \text{, as in interactive Schnorr proof where it's provided by the verifier only after knowing } x \text{ (if not, in that case the signer could be asked to prove ownership of a } \]

\[ \text{random } \alpha \text{ and reveal } \alpha \cdot G \text{ (to specify the output of a one-way hash)} \text{, as in interactive Schnorr proof where it's provided by the verifier only after knowing } x \text{ (if not, in that case the signer could be asked to prove ownership of a } \]

 MLSAS (Multilayer Linkable SAG)

\[ \alpha \cdot H(tx, r \cdot G, c \cdot H_{\text{Multilayer}}(x)) \]

\[ c \cdot H(tx, r \cdot G, c \cdot H_{\text{Multilayer}}(x)) \]

\[ c \cdot H(tx, r \cdot G, c \cdot H_{\text{Multilayer}}(x)) \]

Rings unleashed notes

\[ \text{SAG (Spontaneous Anonymous Group)} \]

- the index value of actual signer \( x \) is random, otherwise \( X \) could be deduced from the order of parameters provided in signature.
- the challenges \( \alpha \) are fixed for other elements, with dependencies depicted by arrows.
- final verification guarantees the dependencies applying to all other and still apply to \( x \) as well (even if originally calculated from \( x \)). How: \( c \cdot H(tx, r \cdot G, c \cdot H_{\text{Multilayer}}(x)) \) forms a closed chain, a ring: that's why it's enough to provide it in signature (it's the "unnamed preserved" single challenge per multi-signature property).
- \( \text{bLSAG is a SAG extended with a key image } X \) (no prevent double spending while still maintaining anonymity, introducing linkability (of signatures) and modified challenges \( c \) to commit to that key image as well);
- \( \text{HuAS is a carefully chosen function returning a random point in EC basepoint subgroup of prime order } X, \text{ acting as generator point for key image } X, x = H(a) \).

\[ c \cdot H(tx, r \cdot G, c \cdot H_{\text{Multilayer}}(x)) \]

\[ c \cdot H(tx, r \cdot G, c \cdot H_{\text{Multilayer}}(x)) \]

\[ c \cdot H(tx, r \cdot G, c \cdot H_{\text{Multilayer}}(x)) \]

CLSA (Concise Linkable SAG)

\[ \text{the schemes currently used by Bitcoin, it's a bLSAG for "pseudo keys" } x^n \text{ and } W \text{ obtained aggregating keys on MLSAG different levels, it provides back-compatible linkability (meaning usual key image property still holds for } X \) \]

\[ \text{W} \text{ doesn't really prevent double spending by itself but it's built from effective } X \text{ and } W \text{ artifacts (that's why it's not used in verification algorithm)} \]

\[ \text{W} \text{ doesn't really prevent double spending by itself but it's built from effective } X \text{ and } W \text{ artifacts (that's why it's not used in verification algorithm)} \]

Credits

This cheat sheet is deeply inspired by Zero to Learn: The Tantalizingly Incomprehensive SAGs and mentioned sources: the notation is only slightly different and with "multilayer" aggregation. A gradual presentation of "Rings" core properties (e.g., key prefling or schema separation for hashtags).