PAXOS

Ring Signatures

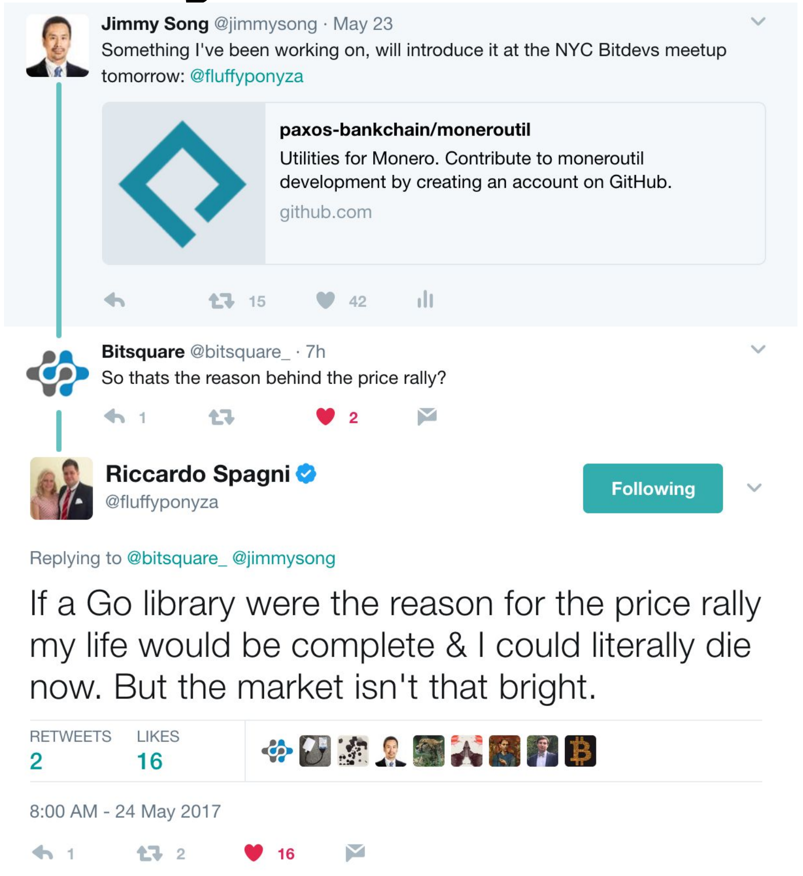
Jimmy Song

- Former VP Engineering Armory
- Core Contributor (13 commits merged)
- btcd Contributor
- Principal Architect at Paxos
- Blogger (http://medium.com/@jimmysong)

Monero Library in Go

- github.com/paxos-bankchain/moneroutil
- Ring Signature creation/verification ✓
- Address utilities work ✓
- Ring Confidential Transactions WIP

Monero Library in Go



Jimmy Song @jimmysong · May 23
Something I've been working on, will introduce it at the NYC Bitdevs meetup tomorrow: @fluffyponyza

paxos-bankchain/moneroutil
Utilities for Monero. Contribute to moneroutil development by creating an account on GitHub.
github.com

15 42

Bitsquare @bitsquare_ · 7h
So thats the reason behind the price rally?

1 2

Riccardo Spagni @fluffyponyza

Replying to @bitsquare_ @jimmysong

If a Go library were the reason for the price rally my life would be complete & I could literally die now. But the market isn't that bright.

RETWEETS	LIKES
2	16

8:00 AM - 24 May 2017

1 2 16

PAXOS

Ring Signature Creation

- Select a total of n public keys P_i, the real P at index P_j
- Your private key is x, and $P_j=xG$
- For $i \neq j$, select a bunch of random scalars c_i and r_i and calculate $L_i=r_iG+c_iP_i$
- Choose random scalar q, $L_j=qG$
- $c=H(m||L_1||L_2||...||L_n)$
- $c_j=c-c_1-c_2-...-c_n$ ($...$ doesn't include j)
- $r_j=q-c_jx$
- $L_j=qG=(r_j+c_jx)G=r_jG+c_jxG=r_jG+c_jP$ or $L_j=r_jG+c_jP_j$
- Signature = $(c_1,c_2,...,c_n,r_1,r_2,...,r_n)$

PAXOS

Ring Signature Verification

- Signature = $(c_1, c_2, ..., c_n, r_1, r_2, ..., r_n)$
- $L_i = r_i G + c_i P_i$
- $c = H(m || L_1 || L_2 || ... || L_n)$
- If $c == c_1 + c_2 + ... + c_n$, signature is valid

Ring Confidential Transactions

- Similar to Ring Signatures
- Adds Adam Back’s Linkable Spontaneous Anonymous Group Signatures (shrinks sigs by about half)
- Requires explicit Transaction Fees
- Amounts blinded using Pedersen Commitments
- Requires Range Proofs for Pedersen Commitments
- Requires Borromean Signatures instead of Schnorr
- Fewer outputs, but longer signatures

Questions?

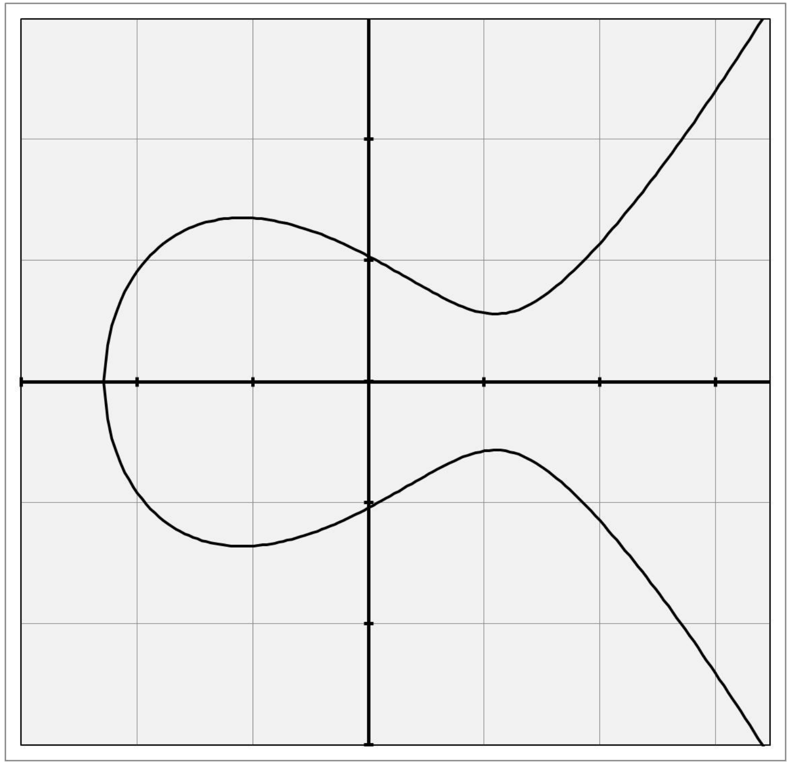
@jimmysong on twitter, github and medium

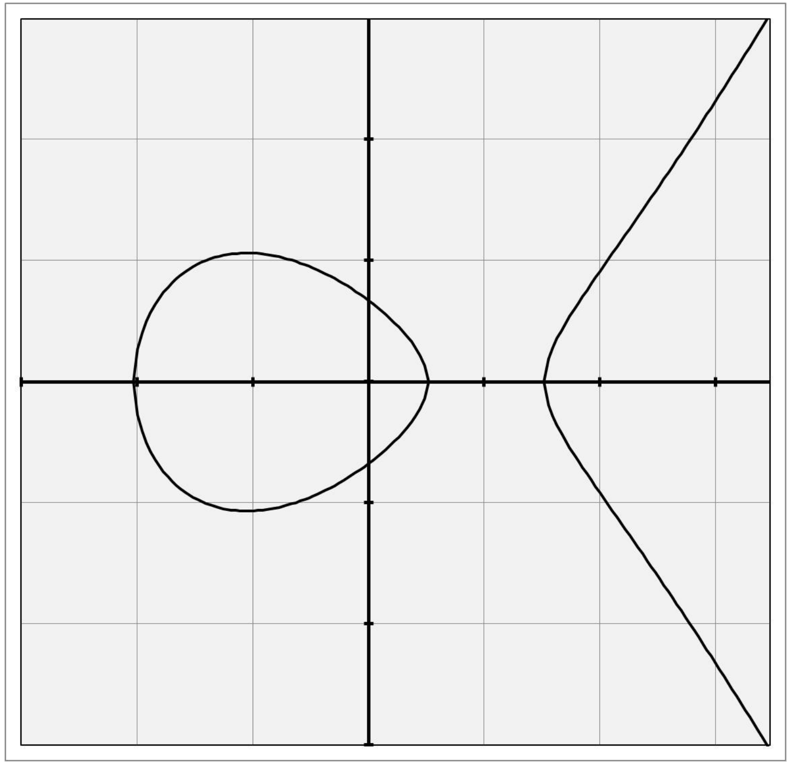
https://github.com/paxos-bankchain/moneroutil

Appendix

Twisted Edwards Curves

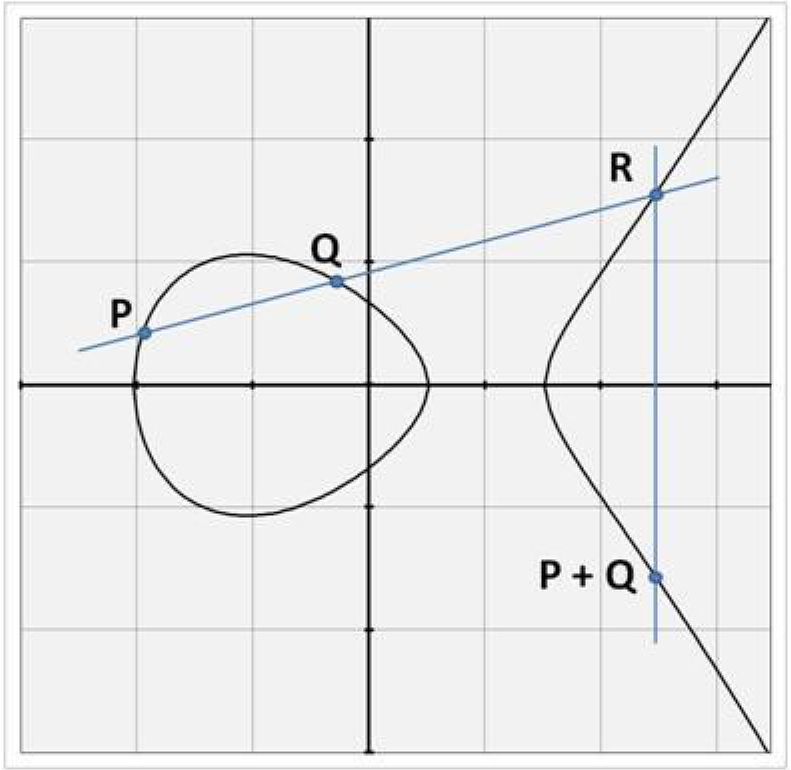
Elliptical: ($y^2 = x^3 + bx + c$)





PAXOS

Group Law for Elliptical Curves



PAXOS

Group Law (two different points)

Curve: $y^2=x^3+ax+b$

$P_1=(x_1,y_1)$ $P_2=(x_2,y_2)$

$P_1+P_2=(x_3,y_3)$

$s=(y_2-y_1)/(x_2-x_1)$

$x_3=s^2-x_1-x_2$

$y_3=s(x_1-x_3)-y_1$

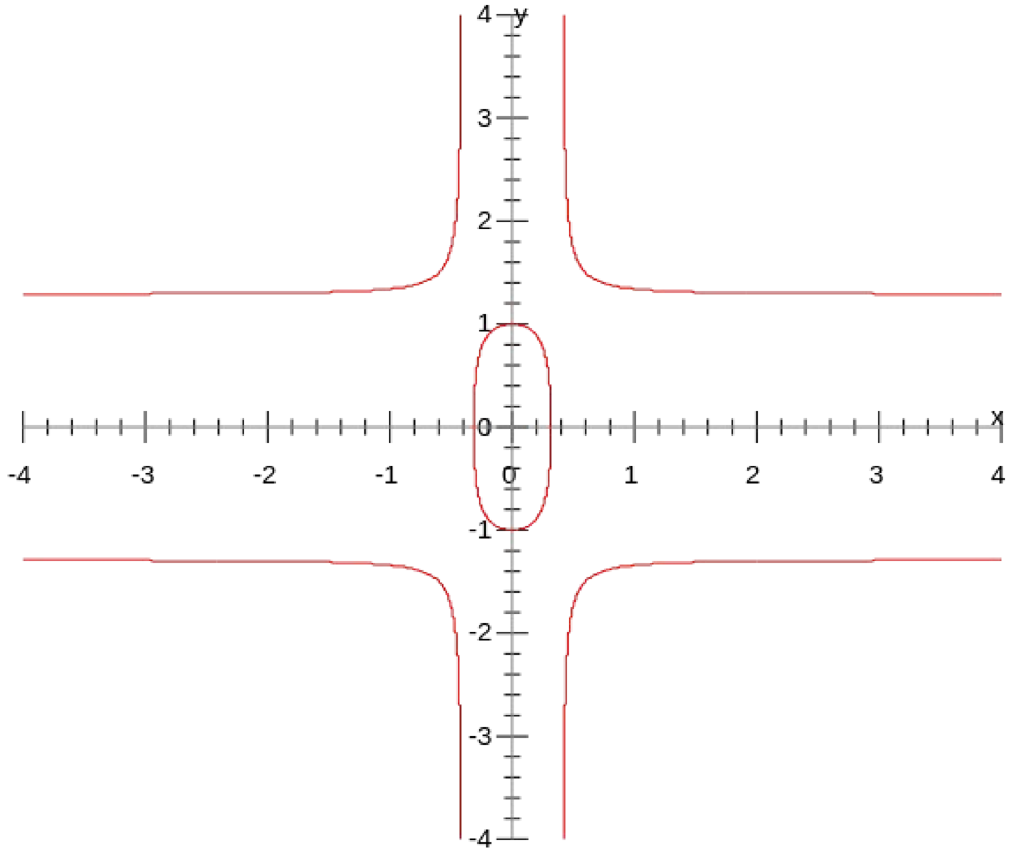
PAXOS

secp256k1

- Curve: $y^2=x^3+7$
- Prime Field (p) = $2^{256}-2^{32}-977$
- Generator Point (G) =
  (79BE667EF9DCBBAC55A06295CE870B07029BFCDB2DCE28D959F2815B16F81798,
  483ADA7726A3C4655DA4FBFC0E1108A8FD17B448A68554199C47D08FFB10D4B8)
- Group Order (n) = 2^{256}-0x14551231950b75fc4402da1732fc9bebf
- Identity = point at infinity
- P=sG=G+G+…+G (s times)
- Scalar multiplication basis for public key cryptography

PAXOS

Twisted Edwards ($ax^2+y^2=1+dx^2y^2$)



PAXOS

Group Law for Twisted Edwards Curve

Curve: $ax^2+y^2=1+dx^2y^2$

$P_1=(x_1,y_1)$ $P_2=(x_2,y_2)$

$P_1+P_2=(x_3,y_3)$

$x_3=(x_1y_2+y_1x_2)/(1+dx_1x_2y_1y_2)$

$y_3=(y_1y_2-ax_1x_2)/(1-dx_1x_2y_1y_2)$

PAXOS

Ed25519

- Curve: $-x^2+y^2=1-(121665/121666)x^2y^2$
- Prime Field $p = 2^{255}-19$
- Generator Point (G) $=(x, 4/5)$ where x is positive
- Group Order (n) = 2^{252}+0x14def9dea2f79cd65812631a5cf5d3ed
- Identity = $(0,1)$
- $P=sG=G+G+\ldots+G$ (s times)
- Scalar multiplication basis for public key cryptography

PAXOS

Cryptonote Protocol

Used by Monero, Bytecoin, etc

Advantages of Cryptonote

- Privacy
- Faster confirmation
- Continuously declining emission schedule

Disadvantages of Cryptonote

- Much larger signatures
- Version 1 requires outputs of same amounts to get privacy
- Scalability is hard
- Tricky bugs
- Hard to do light clients

Schnorr Signature

- Pick a random scalar r with $R=rG$
- Let x be the private key of the signer, $P=xG$ is the public key
- $c=H(m||Encode(R))$
- $s=r-cx \mod n$
- Signature = (c,s)
- $R_v=sG+cP=sG+cxG=(s+cx)G=(r-cx+cx)G=rG$
- $c_v=H(m||Encode(R_v))$
- If $c==c_v$, signature is valid.

PAXOS

Keys in CryptoNote

- All private keys are two Ed25519 scalars (a,b)
- All public keys are two Ed25519 points (A,B)
- `A=aG` and `B=bG`
- a is called the viewing key
- b is called the signing key
- Addresses are encoded in a base58-like encoding of the public key

Encoding and Hashing

- Any point P can be encoded in 32 bytes (255 bits for the y coordinate and 1 bit for the sign of x), call these `t = Encode(P), P = Decode(t)`
- Scalars are encoded to bytes in little-endian format `E(s)`
- Let H be a hash function (Monero uses Keccak256)
- You can hash both scalars and points!
- H_s`=H(E(s))`
- H_P`=8*Decode(`H_s`(Encode(Point)))`
- $H_{P->s}$`=H(`H_P`(8*Encode(Point))||output_index)`

One Time Public Key

- All outputs are to Ed25519 Points
- Choose a random scalar r, with corresponding $R=rG$
- R gets published as part of a tx (transaction public key)
- Create a one-time public key $P=H_{P->s}(rA)G+B$
- One-time private key is $x=H_{P->s}(aR)+b$ because $B=bG$ and $aR=arG=raG=rA$. Note $P=xG$
- Note if you have viewing key a, you can derive P as well
- Only person with private signing key b can derive x
- P is published as the outpoint key (receiver of funds)
- r, a, b, x never transmitted, kept private

PAXOS

Ring Signature

- Based on Schnorr Signatures
- Signer knows one-time private key: $x=H_{P->s}(aR)+b$
- $P=xG$ is the outpoint key being spent
- Must publish $K=xH_p(P)$ as part of the spend. This is how double-spending is prevented. Spending the same output twice would publish the same K.
- To add privacy, other outpoint keys are added to a mix ring.
- Mix ring has z outpoint keys one of which is P

PAXOS

Ring Signature continued...

- Select a bunch of other public keys P_i, the real P at index j
- For $i \neq j$, select a bunch of random scalars c_i and r_i
- For $i \neq j$, calculate $L_i = r_i G + c_i P_i$ and $R_i = r_i H_P(P_i) + c_i K$
- Choose random scalar q, $L_j = qG$, $R_j = qH_P(P_j)$
- $c = H(m || L_1 || R_1 || L_2 || R_2 || \ldots || L_z || R_z)$
- $c_j = c - c_1 - c_2 - \ldots - c_z \mod n$ (... doesn't include j)
- $r_j = q - c_j x \mod n$
- $L_j = qG = (r_j + c_j x)G = r_j G + c_j xG = r_j G + c_j P_j$
- $R_j = qH_P(P_j) = (r_j + c_j x)H_P(P_j) = r_j H_P(P_j) + c_j K$
- Signature = $(K, c_1, c_2, \ldots, c_z, r_1, r_2, \ldots, r_z)$

PAXOS

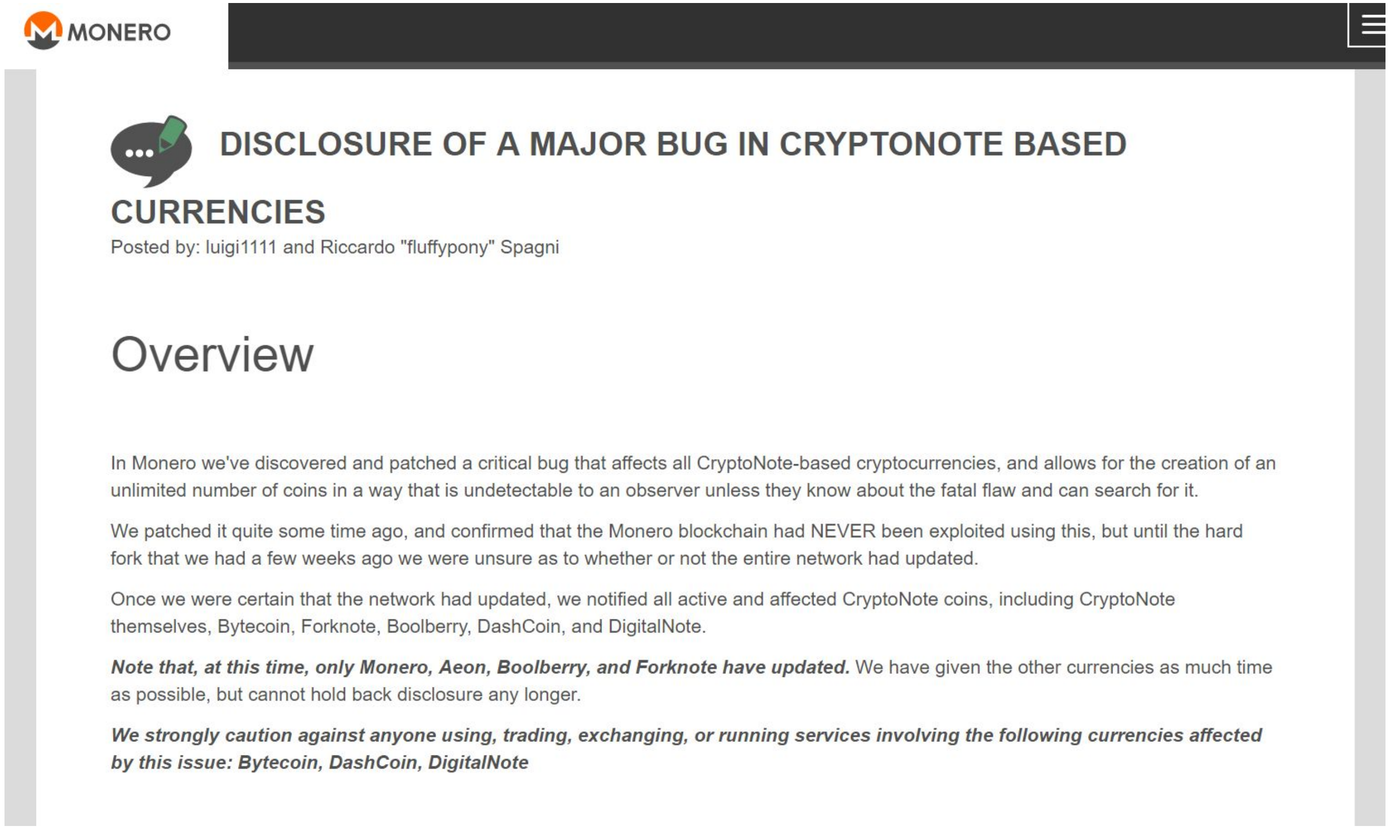
Ring Signature Verification

- Signature = $(K, c_1, c_2, ..., c_z, r_1, r_2, ..., r_z)$
- $L_i = r_i G + c_i P_i$
- $R_i = r_i H_P(P_i) + c_i K$
- $c = H(m || L_1 || R_1 || L_2 || R_2 || ... || L_z || R_z)$
- If $c == c_1 + c_2 + ... + c_z$ mod n, signature is valid

PAXOS

Monero Bug

Bug disclosed on 2017-05-18



DISCLOSURE OF A MAJOR BUG IN CRYPTONOTE BASED CURRENCIES

Posted by: luigi1111 and Riccardo "fluffypony" Spagni

Overview

In Monero we've discovered and patched a critical bug that affects all CryptoNote-based cryptocurrencies, and allows for the creation of an unlimited number of coins in a way that is undetectable to an observer unless they know about the fatal flaw and can search for it.

We patched it quite some time ago, and confirmed that the Monero blockchain had NEVER been exploited using this, but until the hard fork that we had a few weeks ago we were unsure as to whether or not the entire network had updated.

Once we were certain that the network had updated, we notified all active and affected CryptoNote coins, including CryptoNote themselves, Bytecoin, Forknote, Boolberry, DashCoin, and DigitalNote.

Note that, at this time, only Monero, Aeon, Boolberry, and Forknote have updated. We have given the other currencies as much time as possible, but cannot hold back disclosure any longer.

We strongly caution against anyone using, trading, exchanging, or running services involving the following currencies affected by this issue: Bytecoin, DashCoin, DigitalNote

PAXOS

Ring Signature

- Signer knows one-time private key: $x=H_{P->s}(aR)+b$
- $P=xG$ is the outpoint key being spent
- Must publish $K=xH_p(P)$ as part of the spend. This is how double-spending is prevented. This is known only to signer. Spending the same output twice would publish the same K
- To add privacy, other outpoint keys are added to a mix ring.
- Mix ring has $z-1$ other outpoint keys (P_i) and P for a total of z keys

PAXOS

Vulnerability

- Must publish $K=xH_P(P)$ as part of the spend. This is how double-spending is prevented. This is known only to signer. Spending the same output twice would publish the same K
- K must be a part of the group. If not, you can create a different K^* that's not a part of the group but where all the other equations go through.

Mitigation

- Check that K is part of the group by checking that the order is the same: nK=`(0,1)`
- Identity is (0,1) or 1 for the group law.

PAXOS