15 Apr 2020

https://eprint.iacr.org/2020/278.pdf
https://eprint.iacr.org/2020/274.pdf
https://eprint.iacr.org/2020/223.pdf
https://eprint.iacr.org/2020/096.pdf

19 Mar 2020

- https://github.com/openenclave/openenclave - microsoft
- https://github.com/Open-TEE - old
- https://github.com/OP-TEE/ - lenaro
- https://source.android.com/security/trusty - samsun
- Samsung - Tigris
- No sensitive
    - Every key is sensitive (only exportable if encrypted)
- No exporting keys in our interface
    - Out of scope for now
- Just an API for handling crypto only
    - Not an API for interacting with TEE
    - Just cryptographic functions
- Capabilities are predefined configurations

12 Mar 2020

- Secure Enclave interface
    - Query for supported algorithms
    - Access controls
        - Partition keys and users
    - Query for capabilities
    - Fix insecure protocols and algorithms
    - Standard for encapsulating new enclaves
    - Standard for referencing the information from the enclave that is out of scope
    - Design principles
    - Describe security shortcomings
        - Not perfect
        - Require scalable attacks
        - No honey pots
        - Value attacked vs value gained
        - Risk profiles

            - Security Guarantees
            - Introspection
            - Roots of trust
- PKCS11/KMIP shortcomings
    - Algorithm limitations
        - No anonymous credential keys
        - No Curve25519, Curve448
        - No BLS
        - No threshold crypto
        - Attestations only by signatures - no ZKPs
            - HW only
            - SW only
            - Combination of the two
            - ECDSA - DCAP
            - Registration, with whom
            - Attestations Privacy
            - Revocation/Rotation
    - Logical partitioning
        - Key A belongs in domain ‘ABC’ and ‘XYZ’
        - User A belongs to ‘XYZ’ and therefore can access Key A
    - Capability limitations
        - Can sign
        - Can verify
        - No commitments
    - Protocol limitations
        - HTTP
        - SSH
        - Decentralized Attestation
    - Broken standards
        - Can wrap
        - Can decrypt
        - Allowed export
- SGX - RA TLS
    - Attestations
    - https://software.intel.com/en-us/blogs/2019/05/21/intel-sgx-datacenter-attestation-primitives

20 Feb 2020

- The goals for the Secure Enclave interface
    - Plug-and-play
    - What is currently missing
- Secure architectures

    - Key bag
    - VC based authentication

3 Feb 2020

- https://eprint.iacr.org/2020/016 - Better Coconut from Jan Camenisch
- https://eprint.iacr.org/2020/025 - Secret leader election
- https://eprint.iacr.org/2020/043 - Original Alternative to Coconut finally published
- https://eprint.iacr.org/2020/059 - Key wrapping with Chacha
- https://eprint.iacr.org/2020/067 - Chacha SIV like construction

27 Jan 2020

- Revocation
    - What is the optimal structure
    - Sparse merkle tree implementations
        - Signed root on ledger
        - 1 bit per leaf 0=Revoked, 1=Issued
        - Balanced trees
        - References:
            - Efficient Sparse Merkle Trees
            - Compact Sparse Merkle Trees
            - Optimizing Sparse Merkle Trees
            - Sparse Merkle Trees and Bloom Filters
- Revisit AuthZ
    - Is there a better way than a global accumulator

20 Jan 2020

- Spartan vs PLONK
    - Spartan white paper
        - No pairings
        - No required trusted setup
    - PLONK white paper
        - No need to do R1CS
        - Pairings
        - More efficient constructions
            - Encode preprocessing in verification key
    - Microsoft is using Spartan
    - Downside
        - N snarks (N=attributes)
        - 1 proof of knowledge of signature
- Submitting paper to ZKProof 2020

- Double spend addition to Sovrin
    - Txfr credentials
    - Spendable credentials
- Credential protocols
    - https://github.com/privacypass/challenge-bypass-extension
    - Threshold encryption
        - DiSE Slides
        - DiSE white paper
    - Adept Secret Sharing
    - Threshold issuance protocol
    - DKG base Pedersen Keygen
    - Coconut signing
    - Selective disclosure and ZKP presentations
    - 2-RTT issuance
        - Possible without blinded signatures
        - Is it with blinded signatures
        - Requirements:
            - Issuer receives proof of blinded attributes
            - Issuer cannot learn blinded attribute values
            - Prover cannot cheat about blinded values
    - Presentations
        - Issuer provides nonce and presentation request
        - If using ZKPs

13 Jan 2020

- Link secret DID rotation
    - Requirements:
        - Privacy: The secrets look random
        - Correctness: The secrets can be proven to be current and valid
        - Forward secrecy: If the secrets of any previous state become leaked, any updates remain hidden from the attacker
        - Post compromise secrecy: After an update, update secrets become secret again
    - Proposals
        - https://docs.google.com/document/d/148sDef7InHWKuB6P6kQ8PtAFYDBI5KkHThNX8W0zCKs/edit
        - Have a merkle tree on the ledger with leaves in this form "LID:Pubkey" (If using Poseidon or MiMC, have both LID and public key to be separate inputs to hash function.). Whenever the holder changes its key, the leaf data will change.
        - Holder proves knowledge of private key using discrete log scalar multiplication with ZKPs to Verifiers

        - Holder signs transaction to update leaf in tree to new “LID:Pubkey”
- SNARKS
    - Problem statement: Bulletproofs may not be fast enough for revocation checks, but are highly likely not fast enough for AuthZ. zkSNARKs should help here but have the issue with a trusted setup. For revocation, this isn’t a hurdle because trust in the issuer is already required. In this case, Groth16 and bellman can be used. AuthZ has no issuer other than the ledger and thus the majority of the ledger should be involved to great the circuits. This is where PLONK should be used.
    - Marlin

6 Jan 2020

- [Mike] Link secret DID rotation - Needs a write up of the idea
- https://github.com/pornin/curve9767
- RWC2020 Goals
    - Protocols for checking compromised credentials

16 Dec 2019

- Follow up to [Brent Z] Verifier pays issuer: status
- Any updates on PLONK
- ZKSnarks – using bellman to implement revocation checks
- Implementing set membership registries in indy-node
    - Are Ed25519 signatures sufficient for update validations?

25 Nov 2019

- Delegatable credential schemes
    - Cut – A -> B -> C, B is revoked by A, C can’t prove or delegate
      Independent – Continue: A -> B -> C, B is revoked by A but C can continue to prove and delegate
    - Independent – Halted: A -> B -> C, B is revoked by A but C can continue to prove but not delegate
    - Issuance must have a time stamp, revocation has timestamp, and prove revocation happened after delegation
- Rotate link secret
    - Organizations this makes sense where multiple agents have various permissions
    - Similar to AuthZ
    - If using Mirror Curves then the proof could be simplified
        - https://twitter.com/pwuille/status/993572063389605889?lang=en
        - https://twitter.com/christophera/status/993511329146191872

        - Find a curve where BLS381 is an embedded curve (Scalar field is the prime field of BLS381) JubJub, LID must be related to private key
        - https://arxiv.org/pdf/1907.09579.pdf
        - https://gitlab.com/kiliant/zklaim/tree/master/zklaim
    - Have a merkle tree on the ledger with leaves in this form "LID:Pubkey" (If using Poseidon or MiMC, have both LID and public key to be separate inputs to hash function.). Whenever the holder changes its key, the leaf data will change.
    - Holder proves knowledge of private key using hash preimage (public key is hash of secret key) with ZKPs to Verifiers
    - Holder signs transaction to update leaf in tree to new “LID:Pubkey”
    - Dmitry to see how this scheme could work
- Zmix API

18 Nov 2019

- Other methods of anonymizing issuers besides umbrella approach
    - Delegation – root issuer and many delegated issuers
- Txfr credentials
- Apply Poseidon to Bellman
    - Bellman over BN254
- ZK-STARKS in Rust from Matter Labs
- FYI - Algorand impl for BLS
    - https://github.com/algorand/bls_sigs_ref/tree/master/rust-impl
- Anoncreds 2 proofs
    - What is a generic VC interface
    - Selective Disclosure
    - Proof of Signatures
    - Predicates
    - What issuers are accepted

11 Nov 2019

- Brent Z - report on use cases
    - Credential has value, monetized with each use
    - Verifier is willing to pay holder
    - Tollbooth wants a piece of it
    - Tollbooth doesn’t know who is using it
- Application of DKG base Pedersen key generation
    - New issuer
    - Remove issuer
    - Rotate keys
    - Revocation
        - https://dedis.cs.yale.edu/dissent/papers/witness.pdf

        - http://cs-www.bu.edu/~reyzin/papers/multisig.pdf
- zkInterface for snarks
    - Written in Rust
    - Pick a snark backend
    - Same frontend
- Chained credentials
    - Provenance

4 Nov 2019

- [Brent Z] Verifier pays issuer: status
    - Brent to find out use cases
- [Mike L] Threshold Issuance Signatures
    - Move forward with Coconut and possibly add some recommendations from Manu D if needed
    - Very similar to Coconut is https://eprint.iacr.org/2019/1058
    - Proofs are basically the same as a regular signatures
    - DKG based on Pedersen (https://pdfs.semanticscholar.org/642b/d1bbc86c7750cef9fa770e9e4ba86bd49eb9.pdf, https://www.cs.cornell.edu/courses/cs754/2001fa/129.PDF)
    - Rust implementation of secret sharing (Pedersen Verifiable secret sharing, with and without trusted party)

28 Oct 2019

21 Oct 2019

- Incorporating zmix into Indy
- Indy consume new stuff from ursa and zmix
- Aries consume new stuff from ursa and zmix
- Discuss Optimizing Inner Product
- Review Repudiable ZKPs, implementation in lovesh/ursa for CL
- Threshold signatures

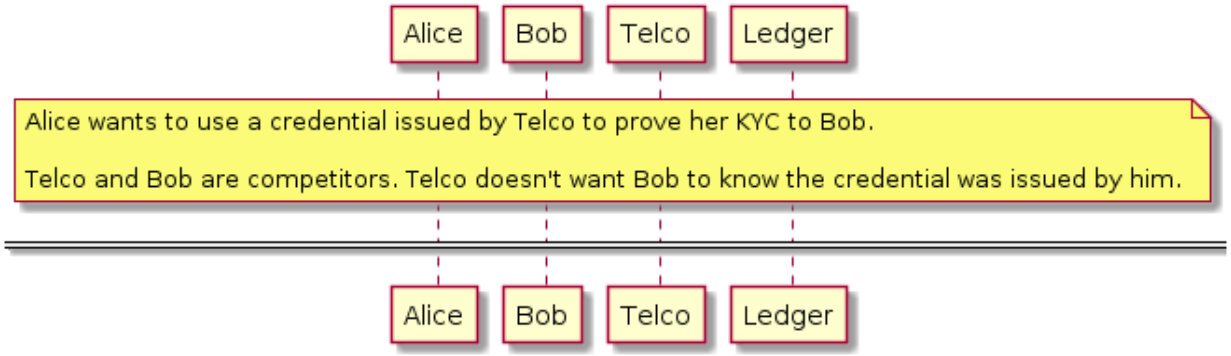
14 Oct 2019

- Review Undeniable signatures
- Threshold signatures - https://www.iacr.org/archive/pkc2003/25670031/25670031.pdf
- Discuss Optimizing Inner Product
- Review Repudiable ZKPs
- Post Quantum possibilities
    - https://pq-crystals.org/

    - https://microsoft.github.io/Picnic/
    - https://bikesuite.org/
    - https://falcon-sign.info/
    - Dave's Caesar cipher
- BLS sigs Rust implementation

30 Sep 2019

- Use cases to solve:
    - Many companies want to charge verifiers for the ability to verify proofs from their credentials like KYC. Handle the case where Credit Union A issues KYC which cost them time and money to do and wants to prevent Credit Union B from accepting it unless they receive some remuneration because Credit Union B did not have to pay for the KYC and got it for free. However, this adds correlation to holder and verifier because issuer sees their interaction. Companies don't want the correlation but want to be paid. One solution, an entity that represents a group establishes a trust framework for generating type of credentials–KYC. All issuers have their own private key in this group. The group has one public key. Verifiers can trust the group if they trust the framework. Companies don't know who signed credential. The Entity can deanonymize signer if necessary. Group signatures but can holder do selective disclosure? PS signatures with 1 of N?
    - Holders and Verifiers do not trust single entity credentials but can trust a group of potential credential signers
    - Credential hides signature, the verifier accepts any signature no particular issuer, payment address hidden in credential, verifier doesn't know who he paid but issuer can see he was paid
    - Can settling occur out of band and be epoch based. Aggregation helps to preserve privacy
    - https://eprint.iacr.org/2019/1058
    - https://www.iacr.org/archive/asiacrypt2002/25010414/25010414.ps
    - Dmitry to investigate issuer anonymity



- M of N issuer signatures
    - Keygen
        - N issuers participate at a given epoch
        - MPC protocol to generate parameters
        - Each issuer creates a private key
        - Collectively create public key
        - Each issuer publishes the group key to the ledger in their cred def
        - Cred def should DIDs of other issuers involved in process
    - Sign
        - Holder looks up DIDs of available issuers
        - Holder sends credential request to N issuers
        - Holder receives partial signatures from M issuers
        - Holder aggregates partial signatures and unblinds to create signature
    - Proof gen
        - Same as now
    - New Issuers
        - A new epoch is run
    - Removals
        - A new epoch is run
    - Questions:
        - Can use PS signatures? If so, Coconut or Threshold Dynamic Group Signatures With Attributes (DGS+A)
        - What MPC protocol should be used for keygen?

9 Sep 2019

- M of N issuer signatures
    - Single Issuer with multiple servers
        - Have issuer implement this on their own
        - Not necessary
    - Updates in threshold schemes are hard
- Verifier doesn't know exact issuers signed
- Mike to write up protocol for multi issuer signature process
- PLONK instead of SONIC
- Just code Bellman for now and know that's probably the best case scenario

26 Aug 2019

- M of N issuer signatures
    - Each issuer computes non-interactively, aggregated by credential holder
        - Desire is this one
    - Secure MPC model
    - Secure MPC key gen

    - Need to do efficient updates
    - Adds and removes
    - Secure MPC API in ursa
        - Threshold signature key generation

19 Aug 2019

- Sonic, Bellman, Poseidon, 4-ary or 8-ary Merkle Tree
- [Dmitry] Report on other methods for generating curves based on other curves that are better than JubJub
- Chained Credentials
    - Requirements
        - Parent is required
        - Proof that the chain is valid
        - Aggregated credentials from multiple parents if needed
        - Linkable Indistinguishable Tag (BLS)
        - What is being signed
    - Proposal
        - Multisign a credential
        - Need a proof where issuer got credential
        - Save a proof from each issuer
        - Same as redundant delegatable credentials
- Delegatable Credentials
    - Two models: children revoked when parent is revoked (Cut), and children continue to be valid even if parent is revoked (Redundant)

12 Aug 2019

- Options for faster set membership merkle proofs
    - Use case-payments
    - Proofs must be less than 1 second including all predicates
    - Use Sonic https://eprint.iacr.org/2019/099
        - Update once a month
        - Publish to ledger
    - Sonic is using bellman
    - Dmitry says Aurora could work also
        - Aurora is stark like
        - Bigger proofs but much faster
        - No trusted setup
        - https://eprint.iacr.org/2018/828
- Chained credentials
    - Provenance
    - Car manufacturing

        - Manufacturer received parts from N suppliers
        - Manufacturer sold to dealer
        - Dealer sold to first owner
        - First owner sold to second owner
    - Receive issued credentials and proof from parent(s)
        - Proofs avoid the need to walk entire chain but don't prevent it
- Rotate link secret review
    - Link secret is a JubJub curve keypair
    - Use JubJub curve for EdDSA signatures
    - Dmitry found other methods for generating curves based on other curves that would be better than JubJub.

29 Jul 2019

- Detecting MITM with Credentials
    - Alice and Bob establish connection
    - Mallory inserts between Alice and Bob so Alice and Bob has Mallory's public key
    - Alice prepares a BBS+ proof as stated in Anoncreds 2.0
    - Alice adds her public key to the challenge hash in the proof
    - Alice sends the proof to Bob.
    - Bob verifies
        - If proof w/o her public key succeeds and proof w/public fails, potential MITM
        - If proof w/public key succeeds and proof w/o public key fails then ??? (redo?)
        - If both checks succeed, No MITM or Mallory knows Alice's credential signature and attribute values to mimic the proof
        - If both checks fail, Alice does not know the attributes or the signature
    - Expand public key to be the context that Alice and Bob operate in
        - Public keys from both
        - Channel details
        - History of communication
- Rotate link secret
    - Organizations this makes sense where multiple agents have various permissions
    - Similar to AuthZ
    - If using Mirror Curves then the proof could be simplified
        - https://twitter.com/pwuille/status/993572063389605889?lang=en
        - https://twitter.com/christophera/status/993511329146191872
        - Find a curve where BLS381 is an embedded curve (Scalar field is the prime field of BLS381) JubJub, LID must be related to private key
        - https://arxiv.org/pdf/1907.09579.pdf
        - https://gitlab.com/kiliant/zklaim/tree/master/zklaim

    - Have a merkle tree on the ledger with leaves in this form "LID:Pubkey" (If using Poseidon or MiMC, have both LID and public key to be separate inputs to hash function.). Whenever the holder changes its key, the leaf data will change.
    - Holder proves knowledge of private key using discrete log scalar multiplication with ZKPs to Verifiers
    - Holder signs transaction to update leaf in tree to new “LID:Pubkey”
- Verifying self attested attributes
    - Add in attribute data to challenge hash just like MITM
    - Add to Anoncreds 2.0 paper
- Verifiable Encryption alternative
- Delegatable Credentials
- Repudiable proofs

22 Jul 2019

- Aggregatable signatures that are pairing friendly
- Rotate link secret in cred. Replace a link secret with Link secret identifier (LID) in credential

  Here is an approach:

  1. Have a merkle tree with leaves in this form "LID:Verkey" (If using Poseidon or MiMC, have both LID and Verkey to be separate inputs to hash function.). Whenever the holder changes its key, the leaf data will change.
  2. Credentials have LID in-place of link secret.
  3. Verifier sends a nonce as part of proof request.
  4. Holder proves
     a. the LID across credentials used in proofs is same
     b. creates a EdDSA signature over the nonce
     c. signature over nonce can be verified using a public key
     d. he knows a leaf in the merkle tree containing LID from step a and public key from step c. LID and verkey are not revealed. The signature is not revealed to the verifier. It is created outside the circuit and passes as private input.

  However this approach makes us do EC operations in circuit (sig verification) so it will be expensive. Not impractical though.

15 Jul 2019

- Hash to Curve reference
    - https://github.com/cfrg/draft-irtf-cfrg-hash-to-curve
    - From BLS call this morning, Riad Wahby recommends HKDF for Hash to Curve instead of SHA256 to get to a Field Element

    - For performance the BLS WG recommends https://github.com/relic-toolkit/relic
- Anoncreds 2.1
    - Threshold PS signatures - section 4
    - Spendable credentials - Transferable, Limited use
    - Delegatable credentials
        - Need redundancy to allow delegates to not be revoked
    - Repudiable proofs
    - Multiple attributes for range proofs
        - A1 + A2 < Term
        - C1.A1 - C2.A2 >= Term

08 Jul 2019

- [Mike L] ZKLang code draft and data model
- [Mike L] Lovesh's Verifiable Encryption use case
    - Can we use ElGamal encryption like here instead?
- [Mike L] Manu D said his coauthors want to revise the paper (Threshold PS), won't be released for a while
- Hart M and Dmitry K - report on recommended methods for hashing to BLS12-381
    - https://github.com/kwantam/bls_sigs_ref
- Dmitry and Lovesh
    - Optimizing hash S boxes
- Transferable credentials
    - Portion
    - Whole
        - Similar to Authz with some changes

01 Jul 2019

- [Mike L] Armando Faz from Cloudflare and Justin Drake from Ethereum Foundation are working on an optimized BLS12-381 implementation and want to collaborate with Me on creating an implementation in Ursa.
- [Lovesh] Optimizing Poseidon Hash for our use case
    - Next steps
- [Manu D] Report on getting Threshold PS signatures in eprint
- Authz protocol in depth
    - Hash to curve point
    - Recommendation depends on the curve
    - Hart M and Dmitry to find recommended method for hashing to BLS12-381
        - Possibly https://eprint.iacr.org/2019/403.pdf
        - Source code https://github.com/kwantam/bls12-381_hash
- Delegatable credentials
    - First draft

    - Detailed discussion here
    - Next steps
- Transferable credentials

24 Jun 2019

- AMCL
    - Are there faster options
    - https://github.com/herumi/mcl
        - TL;DR - Uses Openssl or GMP for curve operations
        - Has BN254 and BLS12-381
        - API similarities to AMCL
        - Same maintainer problem as AMCL, 1
- If we know the depth and width
    - Then we can optimize the hash S boxes
    - Reduce the number of partial rounds
    - Reduce the linear combinations
- AuthZ protocol in depth
- Repudiable proofs
- Transferable credentials
- Delegatable credentials

18 Jun 2019

- [Mike L] ZKLang meeting schedule - when should this be? Need feedback from Franz-Stephan
- [Mike L] Anoncreds 2.0 paper in ursa-docs
- [Mike L] Review Lovesh 4-ary tree implementation
    - 30s Proving time is too slow. What depth allows a proving to be 3s or less?
    - Would snarks be faster? What's the best way to test this?
        - Trade-off: setup vs efficiency (see blog post)
    - Average case is guestimated to be half
- [Mike L] AuthZ protocol in depth
- Perhaps the issue is AMCL
    - Is there a faster more optimized library?
- Repudiable proofs
- Transferable credentials
- Delegatable credentials

11 Jun 2019

- Lovesh report on Poseidon benchmark
    - Reasonable default size 4-ary tree for 10 billion is enough = 4^17 = 16 Billion

    - Dalek (Ristretto)
        - For 4-ary tree of height 16 (has 2^32 leaves) and Poseidon rounds 148, no of multipliers is 9328 and linear constraints is 21666
        - Proving time is 4.87s
        - Verification time is 1.66s

      The numbers above are when the proof size is maximum
      Poseidon, 4-ary merkle tree
    - AMCL (BLS12-381)
        - For 4-ary tree of height 16 (has 2^32 leaves) and Poseidon rounds 148, no of multipliers is 9328 and constraints is 18658
            - Proving time is 30.59s
            - Verification time is 4.41s
        - For 4-ary tree of height 12 (has 2^24 leaves) and Poseidon rounds 65, no of multipliers is 4008 and constraints is 8018
            - Proving time is 9.29s
            - Verification time is 959.86ms
- [Dan M] Opinions on utility of Yao's Garbled Circuit?
- Repudiable proofs
- [Mike L] Overleaf -> ursa-docs
    - Set to the status to be: In Progress, Proposed, Accepted, Adopted
- Transferable credentials
- Delegatable credentials

4 Jun 2019

- Anoncreds 2.0 list
    - BBS+ signatures - BLS381
    - Bulletproofs for predicates
    - Sparse Merkle tree revocation - Discuss ZKSnark w/o trusted setup vs w/trusted setup - see paper
    - Poseidon-252 (actually 253)
        - For binary tree of height 253 and Poseidon rounds 8, no of multipliers is 37444 and constraints is 87033
            - Proving time is 14.359508008s
            - Verification time is 2.176287639s
        - For 4-ary tree of height 128 and Poseidon rounds 8, no of multipliers is 20864 and constraints is 47874
            - Proving time is 7.560476903s
            - Verification time is 1.163940276s
    - AuthZ
        - ZKSnark - MPC (Sonic?) by ledger
        - Some sonic impls
            - https://github.com/adjoint-io/sonic

        - https://github.com/LayerXcom/lx-sonic
- Anoncreds 2.1 list
    - Multi issuer - PS signatures - BLS381
    - Proof of some function of attributes
        - Sum of two or more attributes ≥ *N*
    - Transferrable credentials
        - Whole
        - Portion
- Transferrable Credentials Protocol
- ZKLang spec and ZKL-JSON.
- Repudiable proofs using Bulletproofs.

28 May 2019

- Proofs for attributes across credentials like
    - Att1 + Att2 < Threshold (Same credential)
    - Att2 + Att7 > Threshold (Two different credentials)
        - Verifier will have to create
- ZKLang
    - Formalizing so cryptographers and engineers are in agreement
    - Proof description
    - Older ZKLang spec and ZKL-JSON.
- Wire message format review

21 May 2019[Mike L] Interactive proof - What stops Bob from replaying the interaction with Alice to show Alice proved something to him?

    - How do we make SNARKS interactive? It can be do
    - ne. Break into parts and add inputs from both parties
    - Designated verifier proofs paper
    - Non-Interactive Keyed-Verification Anonymous Credentials
- [Mike L] Implementing Poseidon hash function in libzmix or ursa?
    - Standalone or as part of the ZKP directly as specific implementations
    - Reference notes but for BLS12-381 or BN254.
    - Can we copy and modify Dusk-Network
        - This implementation is not complete
    - We have 2 implementations of Poseidon with Bulletproofs
        - On Ristretto
        - On BLS12-381 using Milagro
- [Kyle DH] Wire message format review and improvements

    - Look at moving to an MLS style messaging system as well as other end to end encryption options to build a p2p and n-wise (group) encrypted messaging protocol
    - Message Layer security protocol draft document
    - Message Layer Security architecture draft spec
    - List of MLS implementations based on current protocol draft

14 May 2019

- Anonymous Revocation
    - Sparse merkle tree
        - Every member of a set added to the leaves of merkle tree
        - Leaves populated by hash index?
        - Allows precalculation of merkle tree
        - When revoked index set to null
        - Built using specific hash function
    - Proofs
        - Depends on the hash function used to build Tree
            - should select a hash function that is friendly to the method.
        - create a bulletproof for 30 repeated invocations of the hash function.
        - for AuthZ, the value of the leaf is commitment to the address of the policy, so that
        - circuit
- Interactive zero knowledge proofs for repudiable proofs in Anoncreds
    - Should we refactor Ursa to support both interactive as well as out non-interactive zero knowledge proofs
    - If a proof is repudiable, the verifier cannot prove that the holder proved it to them.
    - derive the next challenge from the verifier's signature on the first nonce?
        - The Prover needs to be unable to construct the proof herself.
    - suppose Alice proves something to Bob.
        - Bob should not be able to repeat the proof
        - It may also be nice if Bob was unable to prove that the proof happened.
    - Two or three weeks of dev time?
- Clarification on state proofs, BLS signatures, aggregation already there?
    - Implementation of aggregated BLS signatures is part of the state proofs now.
    - High level flow
        - Request some value from the ledger, the value comes with the aggregated signature of the verifiers as a state proof.
    - BLS Multisig code in Ursa

7 May 2019

- Deep dive into Anoncreds 2.0 Proof math

- Verifiable Encryption
    - Camenisch and Shoup scheme from 2003
        - Do we know a more recent scheme or better scheme than this
    - What if we decided to implement Elgamal encryption (since the message is going to be small, a prime field element) with a circuit (Bulletproofs) and then prove properties about the encrypted value.

30 April 2019

- Review math in Anon Creds 2.0
- Credential index can be the hash of credential
    - Issuance would have an additional step
- Revocation Registry on double spend ledger
    - Benefits are similar to transferable credentials
- Poseiden published in the next few days
- [Dmitry K] Tradeoff verification time for proof size
    - Extension to Bulletproofs
    - Prover's time not necessarily significant
- Convert points between curves is heavy
    - Commit to the same value on two different curves is a range proof
        - Let G1, G2 be two groups, C1 and C2 are some commitments prove that you know X such that X<min(ord(G1),ord(G2)) and C1 and C2 are commitments to X in G1 and G2, respectively.
    - 2 group element range proof

23 April 2019

- Authz Overview and Review
- Authz New architecture
- Pointcheval Saunders Threshold Signature
    - $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$
    - Groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ have prime order p.
    - Must be Type-3 pairing
    - Secret key $x, y_1, \ldots, y_{k+1}$
    - Public key $X = g_2^x, Y = g_2^{y_j}$ for $j \in [k+1]$
    - Signature
        - Message $(m_1, \ldots, m_k) \in \mathbb{Z}_p^k$
        - Random $m' \in \mathbb{Z}_p$
        - Random $\sigma_1 \in \mathbb{G}*$
        - Compute $\sigma_2 \leftarrow {\sigma_1}^{x+\sum_{j=1}^{k} y_j m_j}$

        - Output is (m', σ_1, σ_2)

16 April 2019

- [Mike L] Report on librustzcash vs Apache Milagro
- [Mike L] Report on ZKProof
- Dmitry report on research for BBS+, PS weaknesses
    - No weaknesses found in the last years.
    - PS is based on less natural assumptions.
    - Performance, signature size, etc. are comparable.
- [Maria D] Report on Forward Secure Multisignatures
- [Lovesh] Should Bulletproofs codebase using BLS12-381 curve be part of Ursa? How do we plan to review it?
- [Lovesh] Do we need (Zk-Snarks) circuits using Bellman for Ursa?
- [Mike L] AnonCred 2.0 implementation details and updates
    - BBS+/PS
    - AuthZ
    - Bulletproofs
    - zkSNARKS
    - SHARKMIMC
        - Poseiden
        - Stark
- [Mike L] Anoncreds 2.0 components
    - Ursa
        - Crypto components
    - Indy-sdk
        - Schema 2.0
    - Indy-node
        - Merkle Trees
        - Authz policies
- Dmitry: Poseidon hash function, suitable for Bulletproofs, is in third party evaluation.
- Dmitry to modify Authz paper.
    - New version

9 April 2019

- Delegatable credential feedback
- Proofs to groups (≥2 verifiers)
    - Prove to one verifier that can pass proof to auditor
        - As long as its possible to prove the nonce was generated by the verifier and fresh then the proof is fresh as well
    - Proof to group as a whole (blockchain)

        - Might need additional binding to prove the Prover's identity
        - Prevent proof presentation reuse
        - Link presentation with additional secret
            - Link to a key that hold certain number of coins
            - Proof generator must hold the key
            - Creates linkability of Prover to non fungible asset not across presentations
- Multi issuer credentials
    - Threshold unlinkable signatures
    - Hart thinks Pairing solution (PS) sigs will scale better for signature generation but ECDSA with snarks will be faster for smaller groups
    - Dmitry - https://eprint.iacr.org/2016/013.pdf - signature is very inefficient (curve size ^ 8)
    - Hart would be surprised if this is more efficient signing than pairings
    - Pairings appears better at present
    - Mike to test implementing PS vs BBS+
        - Single case – performance, memory,
    - Dmitry to research any weaknesses or vulnerabilities with either signature and report
- Librustzcash vs Apache Milagro
    - Affine vs Projection vs Jacobian coordinates
    - No real difference between them
- (un)linkable pseudonyms see papers here and here, or slides

2 April 2019

- Delegatable credentials: first approach
    - An issuer can issue credentials.
    - An issuer can also delegate the right to issuance provided he has the right to delegate.
    - The first issuer, also called the root issuer can both issue and delegate right to issue a credential.
    - For any credential who issuance rights have been delegated to one or more other issuers, the ordered list of issuers forms a chain, called "chain of command". eg. If issuer I0 issued a delegation credential to I1 who issued a delegation credential to I2 who then issued a credential to I3. Now when I3 presents a proof, the chain of command per that proof (and verifier) is I0, I1, I2, I3. Note that I3 does not need to be an issuer.
    - Each issuer has a private key used to sign credentials and public key to verify credentials.
    - Anyone (issuer or holder) in the chain of command needs to know public keys of all issuers. The public keys need not be on a ledger but some database hosted by the root issuer. A root issuer like a bank's headquarter might host an internal

database accessible only to bank employees. Various bank employees can then be the delegatees.
    - The verifier of a proof from a delegatable credential does not need to know the public key of anyone except the root issuer, i.e first member of chain of command.
    - Delegation of issuance is done by transferring the received credential (that gave right to delegation) in plaintext to the delegatee. In above example of I0, I1, etc. I0 issued credential C1 to I1, I1 issued credential C2 to I2 but I2 also gave C1 to I2, I2 issued C3 to I3 but I2 also gave C1 and C2 to I3.
    - Credentials can only be revoked by the issuer who issued the credential not by someone before the issuer in the chain of command.
    - Revocation is done by issuer maintaining a merkle tree of revoked credential ids as leaves. Absence of a credential id in the tree is proved by having null at the leaf denoting the credential id (Similar approach as revocation in Anoncreds 2.0). When issuers (root or others) revoke credentials, they prove to the system hosting the tree (ledger or root issuer's hosted publicly accessible database) that they issued that credential.
    - It requires the use of generic proving systems like SNARKs/Bulletproofs but this is acceptable since Bulletproofs is utilized in Anoncreds 2.0
    - Note that since these credentials are not compatible with regular anonymous credentials (Anoncreds 1.0 or 2.0), the method to prove linkage between regular anonymous credentials and delegated anonymous credentials is different from the current method

- Proofs to groups (≥2 verifiers)
    - Honest prover
        - Randomness beacon (merkle root)
        - All honest verifier
        - ≥1 honest verifier
        - All malicious verifiers
        - Do they collectively agree the proof is valid or individually know its valid
            - [Mike L] use case which one, what does the ideal protocol look like
- Interesting signature for consensus here and here
- Multi issuer credentials

26 March 2019

- Transferrable credentials
    - Mike should write this up (requirements):
        - Stock certificate use case is useful here
        - current holder can prove they possess the credential
        - only the current holder can prove they possess the credential
        - the former holder can prove they once possessed the credential

        - current holder cannot prove former holder once possessed the credential
        - the transfer transaction does nothing to create correlation between current and next holder
        - the issuer is not involved in transferring the credential from holder to another holder
        - the ability of the holder to prove they no longer possess the credential is not in scope.
        - This is looking a lot like zcash, is there anything zcash has that we don't need?
            - no need to subdivide the credentials
            - no founders reward :)
            - do we still want viewing keys, or just spending keys?
                - history of the credential can be passed along as a separate credential, if this is necessary.
        - Is there anything zcash doesn't provide that we do need?
            - probably not at this time
            - in the future, possibly traffic analysis resistance.
    - Issuer creates a transferable credential
        - gets a blinded secret from a holder
        - combines that blinded secret into a commitment with the credential ID
        - puts the commitment into a registry
- Delegatable credentials
    - Dmitry: first approach
    - Question about revocation: revocation is based on the index, could revocation also be done according to the holder's public key? This would allow for revocation of all the credentials a holder has been issued at one time.
    - Question: Could DIDs be used instead of public keys? difficult to prove control of a DID anonymously.
    - Comment: Alternatively context can be bit-vector with bits at different indices which are set to specify possession of corresponding privileges.
- unrelated thing: please check out the ursa encryption and zmix APIs.
    - https://github.com/hyperledger/ursa-rfcs/pull/7

19 March 2019

- Dead Drops
- Transferable credentials
    - How do they differ from cyber coins?
    - Important to whom it belongs not its origin
    - Ledger is just a registry of current owner
    - Similar to AuthZ
    - Spending means transfer to nobody or to verifier

    - Owners and transfers should still be anonymous
    - Still do proof of ownership
        - Even selective disclosure could still work
    - Owner and cred ID in registry – Commitment to both in the registry
    - Issuer mints the registry signs updates
    - Owners submit proof of knowledge of value in registry and commitment and proof of correctness that commitment has special form
    - All or nothing vs partial ownership changes
- Delegatable credentials
- Rotating a link secret w/o revoking the existing credentials
    - Can't prove linking in credentials anymore just ownership
    - Not part of the signatures
- Proofs to groups
- ZKPs for Key Agreement

- DPKI for TEE and HSM attestations (Jon and Mike)
    - Almost nobody uses DAA
    - Multi issuer setup
        - Prevent system lock
        - Prevent vendor lock
    - What should the credential look like
        - Att1 - Overall can I establish an enclave?
        - Att2 - Which hardware targets are trustable
            - Serial numbers
            - Who can authorize these targets
            - Don't want multiple enclaves running at the same time, Mux
        - Att3 - Data signed in cross paranoid way
            - Signed by at least two parties
            - What data will trust what engines–could be sensitive
                - Not going into an emulator or VM
    - Who is authorized to execute the code
    - Who is authorized to sign the code
- Trustworthy repo for trusted targets–cloud service for enhanced security capabilities
    - Hardware agnostic
    - Updateable by Trust Framework–open to additions, but trustable, world accessible
    - MAC Sec, IPSec, Deep Quotes
    - Revocation can be controlled by versioning
        - Vulnerabilities found = bump to newer version
- Remote attestations
    - Locally is simple and trust is easier

    - Geographic boundaries require more complex proofs
    - Version
- Alice wants to register Secure Enclave
    - Sends her enclave information to the decentralized authority and receives a credential
    - Key can only be used by that enclave
    - Challenge-response
- Alice wants to prove her data has only been handled by her enclave
    - (Ncipher certificate chain example)
        - Enclave capabilities are signed by HSM's attestation key
            - Signing and Encryption
            - Signing only
            - Other crypto primitives
        - Common Criteria and FIPS; key never leaves the box, only used for creating certificates, no emulator, no virtual environment
    - Sign all output data with key that was registered previously
    - Sends proof of enclave
- HEX5 TEE

12 March 2019

- Use Case DPKI for TEE and HSM attestations
    - Mike L and Jon G to discuss deeper and diagram protocols and proposals
- Deaddrops
    - Establish a relationship based on previous interactions
    - Use case – when both endpoints change locations at the same time
- Double spend proof ledger
    - Structure preserving signatures with Schnorr proofs
    - To spend, proof of knowledge of signature. Multi issuer credentials
- Delegatable credentials
- Transferable credentials
- Rotating a link secret w/o revoking the existing credentials
- Proofs to groups

4 March 2019

- Comments on release candidate draft AnonCreds 2.0
- AnonCreds 2.1
    - Policies
    - Multi issuer credentials
    - Delegatable credentials
    - Transferable credentials
- Rotating a link secret w/o revoking the existing credentials

26 Feb 2019

- Sparse merkle tree implementations
    - Signed root on ledger
    - 1 bit per leaf 0=Revoked, 1=Issued
    - Balanced trees
    - References:
        - Efficient Sparse Merkle Trees
        - Compact Sparse Merkle Trees
        - Optimizing Sparse Merkle Trees
        - Sparse Merkle Trees and Bloom Filters
- https://github.com/QED-it/gadget_standard
    - Creates an intermediate step for linking zkp libraries to gadgets
- What's needed for the ledger
    - Arbitrary Merkle trees
    - Policy Addresses
        - Could this serve for revocation for attestations
        - Authz
- Auditability for proofs
    - Generate both repudiable and non-repudiable proofs
    - Allow honest verifier to prove verifiability of received proofs to auditor
    - Limit damage from colluding verifiers
    - Current implementation
        - $c \leftarrow H(T_{set}||n_0)$
        - n_0 generated by Verifier
    - Can we improve it by adding entropy from Prover, Verifier, and Ledger?
- Proving to a group of people
- Delegatable credentials

19 Feb 2019

- [Mike L] Sparse merkle tree implementations
    - Witness updates use SharkMIMC for nodes
    - Signed root on ledger
    - 1 bit per leaf 0=Revoked, 1=Issued
    - Iden3
    - What's needed to use zkinterface?
        - Core written in C++
        - Do generated gadgets need zkinterface dependent code or can they be independent?
- Set memberships for enums

    - Bullet proofs with bitmaps and commitments
    - See here, bulletproofs implementation
    - For each i in bitmap:
        original_set[i]*bitmap[i] = value*bitmap[i]
        Eg original_set=[10, 20, 30, 40], prove that i know a value in this case, 30
        bitmap = [0,0, 1, 0]
        For each i in bitmap:
            i
- Policies on the ledger
    - AuthZ
    - Expand for other policies
- Other types of predicates?
    - Transferable credentials - Token
        - Double Spend protection
- Data Stores (Asset protection)
    - APIs needed, What does this look like? PKCS11 – V3 is better
        - SGX, TrustZone
        - TPM, HSM
        - Mobile secure enclaves
    - Ledger and Issuer key storage
    - Key wrapping
    - Proof of nonce generation from TEE or Enclave
    - Proof of signature from TEE or Enclave
    - Borrow some cryptree concepts
    - Kademlia for syncing and decentralization
    - Scuttlebutt for handshaking or possibly
        - Or DAKEZ
    - [JonG] Biggest threat – checking against central authorities, attestations require online. I'm working on taking out the central authority and making it distributed.
        - Use PKCS11 V3 for API
        - Root is non standard
- Ledger 2.0
    - VRF similar to Dfinity consensus for randomly selecting Primary
    - Need non-deterministic BLS

12 Feb 2019

- What issues for Merkle Tree revocation
    - Correlation risks
        - Update witness, know modified nodes
        - Need to hide merkle tree access pattern
        - Reconstruct the rest of tree
        - 1 bit per leaf node

        - Should it be stored on the ledger or have a reference somewhere else
    - Publishing changes
- Zmix – using zkInterface to generate constraint system, create specific gadgets.
- Threshold BLS signatures vs PS signatures
    - Ledger issued credentials
        - State Proofs
        - Group credential work flows
    - Threshold BLS no for anoncreds
- Verifiable Encryption
    - Camenisch-Shoup
- Wallets and Enclaves
    - APIs needed, What does this look like?
        - SGX, TrustZone
        - TPM, HSM
        - Mobile secure enclaves
    - Ledger and Issuer key storage
    - Key wrapping
    - Proof of nonce generation from TEE or Enclave
    - Proof of signature from TEE or Enclave
    - Can we do Structured Encryption (Predicate Encryption) with enclaves?
        - https://eprint.iacr.org/2011/010.pdf
        - https://eprint.iacr.org/2018/551.pdf
        - Similar to Clusion
- ORAM for ledger reads
    - What's required?
    - An example

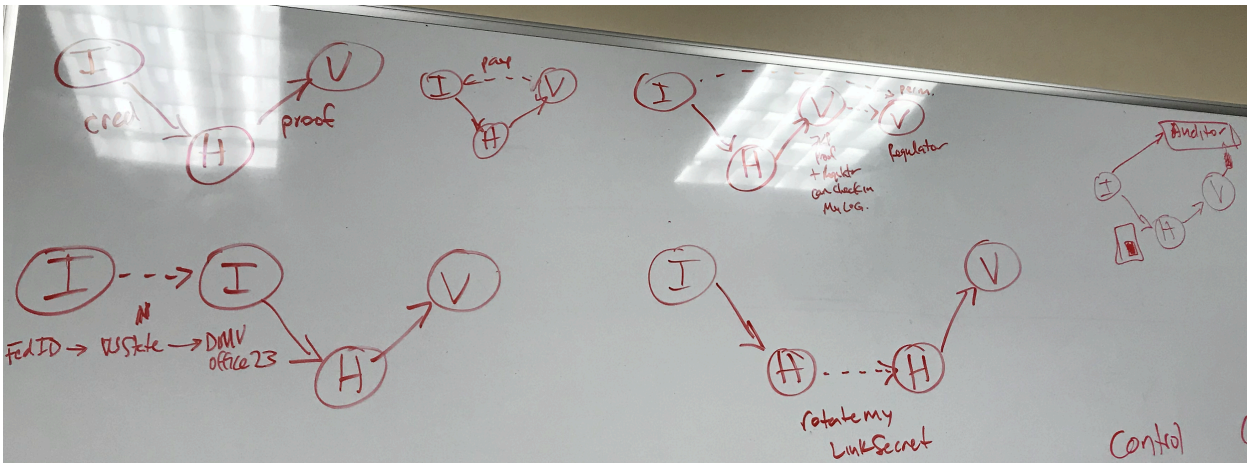
5 Feb 2019

- Credentials 2.0
    - Single signature instead of two
    - BBS+ signatures – BLS12-381
    - Verifiable Encryption
        - Camenisch-Shoup (Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. CRYPTO 2003),
        - Cramer-Shoup (R. Cramer and V. Shoup. Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. EUROCRYPT 2002)
    - Add Pointcheval Sanders later for threshold signatures
        - Maybe bulletproofs and collect multiple signatures instead
        - Signature verification is more expensive
        - Need to verify against all public keys to prevent correlation
        - https://github.com/poanetwork/threshold_crypto

        - Bulletproofs – range proofs
        - R1CS – Enum/static set memberships
        - Snarks w/Merkle Trees – Dynamic set memberships
            - Revocation
            - What are the potential correlation issues?
- [HartM] - Anonymizing ledger writes
    - [Lovesh] - https://arxiv.org/pdf/1701.04439.pdf
    - [Dmitry] - https://eprint.iacr.org/2018/851.pdf and https://eprint.iacr.org/2018/005.pdf
        - Maybe combine this with Riposte
- Zmix Schemas
    - Proof Request – Generated by Verifier, can be on ledger, list of accepted issuers and predicates
    - Proof Offer – Counter proposal from Prover
    - Proof Resolution – simplified version of a Proof Request that helps the Verifier know how to confirm the Proof is valid. Satisfies the non-cryptographic part of the Proof Request
    - Proof Spec – Generated deterministically from a Proof Resolution and because ledger holds the public key (including attribute generators) for Credential Definitions, and reference to top-level Schema.
    - Witness – Prover private input
    - Proof – Crypto token
- Relayed proofs
- Group Credentials
- Issuer delegation
- Link secret rotation

29 Jan 2019

- Credentials 2.0
    - 100% Circuit approach not feasible
    - Moving forward with
        - BBS+ signatures - First draft
        - Circuit based proofs for arbitrary statements
            - Start with bulletproofs
            - Can we use ristretto curves?
    - Need to decide to use single signature or two
        - Two:
            - Pro: Don't need to change revocation in place
                - [Dmitry] Still have a lot work to do anyway
            - Con: Existing system is expensive for big sets with lots of changes
        - One:
            - Pro: One signature to manage

        - Con: Implement set memberships
        - Con: Revocation registry needs changes to Merkle trees
        - Benchmark R1CS vs Native ranges with bulletproofs
    - Group issued credentials:
        - [Pointcheval-Saunders Signatures - Threshold signatures]
        - Other signatures?
        - Shouldn't affect the circuits and PS should be a drop in for BBS+
    - Other credential uses
    - 

    - Relayed proof: A verifier can prove (to the auditor) that he indeed got this proof and he did not intercept this proof between any other parties:
        - The verifier can prove to an auditor that only he could correctly generate the nonce. This is essentially the principle of VRF (Verifiable Random Functions). The basic idea is that verifier has a keypair (not a EC/DDSA or CL sig keypair) and the auditor knows the the public key of the verifier. The verifier always uses his private key to generate the nonce and at the time of audit he proves that only he could have generated the nonce.
        - Another DIY construction that does not use VRF but assumes Verifier has an EC/dDSA keypair and public key is known to auditor:
            - 1. Assume that we work with 32 byte nonce
            - 2. Verifier generates a 32 byte nonce N, signs the nonce to get a signature S.
            - 3. Concatenates the nonce and signature, N||S and hashes the concatenation to get 32 byte hash H.
            - 4. Verifier stores the mapping H=>N.
            - 5. The hash (result of 3) is the nonce he sends to prover.

            - On audit,
              1. The verifier provides the auditor N and S (by signing N).
              2. Auditor verifier the signature S on N.
              3. Auditor hashes the concatenation of N||S and verifies the result is equal to H, which was the nonce in the proof.

22 Jan 2019

- Next steps for Credentials 2.0 Roadmap
    - Security implications of SNARK based protocols
        - Sapling security proof
        - Applications Track proceeding from ZKProof.org (note: may be interesting to see the Identity Framework section if moving to circuit based provers)
    - Potential SNARKs
        - For comprehensive list, check zkp.science
        - Aurora: Transparent Succinct arguments for R1CS
            - Code base to be found
        - Doubly-efficient zkSNARKs without trusted setup - Hyrax
            - Code base for hyrax
    - Circuit based
        - Revocation circuit based logic with merkle trees
        - Circuit proofs for range proofs
        - Good verifiable encryption: [Maria]
            - Camenisch-Shoup (Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. CRYPTO 2003),
            - Cramer-Shoup (R. Cramer and V. Shoup. Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. EUROCRYPT 2002)
    - Nym - a commitment to a secret - one can create as many as one wants (proof of knowledge of a secret = commitment proof)
    - Scope Nym - hash of the scope raised to the secret, one public key per scope
        - Voting
    - Credential can be a collection of signatures usable by verifiers for various proofs
        - What other signatures would be used?
        - Need range proofs and set memberships
        - Possibly verifiable encryption
    - What does this look like?
        - Issuer key management
    - What changes to the ledger are needed?
        - Changes to revocation registries
        - Still need another signature for this but not separate credential
    - What is needed to support hierarchical credentials?
- ZKPs updates
    - I had a discussion with Daniel Benarroch
    - Working similar solution for R1CS system
    - Interested in working with us on this and how it works

- Group issued credentials
    - What does this look like?
    - What changes to the ledger are needed?
    - Would something like the MLS system for adds/removals work?
- Proofs for arbitrary statements using Bulletproofs
    - Can we use commitments from other groups (curves) other than the curve used to implement bulletproofs.

14 Jan 2019

- Meeting day/time change
    - New meeting day Tuesdays at 8am
- Real World Crypto review
    - [Mike] Met with Trevor Perrin, DJB, Tanja Lange, Matthew Green about Noise Protocol
    - Possibly adding enclave programming to Sovrin code
        - Benefits?
        - Where?
        - What:
            - Hardware: SGX, PSP, Trustzone, RISC-V
            - Sanctum
            - Keystone
- Circuit based logic for SNARKS/STARKS/Bulletproofs
    - Should this go in zmix?
    - [Dmitry] Should be done by hash function authors
    - Bellman is low-level, higher level primitives in sapling-crypto
- Credentials 2.0
    - BBS+, PS signatures
    - Credential can be a collection of signatures usable by verifiers for various proofs
    - What does this look like?
        - Issuer key management
    - What changes to the ledger are needed?
        - Changes to revocation registries
        - Still need another signature for this but not separate credential
    - What is needed to support hierarchical credentials
    - Other additions:
        - Verifiable encryption vs Proxy Reencryption
- Group issued credentials
    - What does this look like?
    - What changes to the ledger are needed?
    - Would something like the MLS system for adds/removals work?
- Zmix updates needed for above
- Proofs for arbitrary statements using Bulletproofs

    - Can we use commitments from other groups (curves) other than the curve used to implement bulletproofs.
- Noise protocol
    - Is it valuable?

7 Jan 2019

- ECC based credentials update
- Snark based credentials update
    - https://github.com/lovesh/bellman-examples/blob/master/src/sharkmimc.rs
    - Libstark
        - Dmitry to talk to Stark authors about license and cooperation with rust implementation with Mike, Lovesh, and Brent
- Ledger MPC for snarks
- Ledger based credentials
    - US wide DL credential issued by Montana because Montana is part of the US DL framework
    - [JanC] Potential bottleneck, prefer to do hierarchical based credentials
    - Sov token as a credential
- Double spend proof anything
    - Prove with snark/starks not delegatable credentials
- RSA accumulators recent development, batching updates to the RSA new paper accumulator

17 Dev 2018

- Snark based Credentials. Attempt 1
    - Not much different for STARKS
    - Snarks require values in prime field
    - Starks require values in binary field
- Ledger MPC for snarks
- Ledger based credentials
    - US wide DL credential issued by Montana because Montana is part of the US DL framework
    - Sov token as a credential
- Double spend proof anything
- Zklang witness spec here
    - Proof spec here

10 Dec 2018

- Does Indy need both CL and CKS sigs?
    - Can CL do set membership proofs? No

    - ECC based creds can be used for both - BBS+
    - Revocation Sets
        - Valid handles: 1-10, sign range [1,10]
        - Revoke rh = 5
        - Sign ranges like [1,4], [6,10]
        - Proofs are bigger
    - Hash based accumulators would be more efficient like Merkle Trees
- Bellman answer: wanted simple library, libsnark is complicated
    - Bellman does use a curve with a bigger prime field (BLS12-381) in comparison to libsnark (BN-254)
    - The Matter is doing a SharkMiMC implementation using Bellman and BN254, we will check the performance when they finish
    - No answer on performance questions
- ZKP Payment scenario
- Variable length credentials
    - BulletProofs with PedersenHash - 1KB
    - Snarks marginally smaller
    - Max depth would be defined in the credential def
- Double spend proof anything
- Ledger MPC for snarks
- BBS+ implementation
    - Jan to contact Rafael about his code and contributing to z-mix
        - Maria: we contacted IBM - waiting for them to push the code to the repo
- Zklang witness spec

2 Dec 2018

- Hierarchical credentials
    - Maria report
    - https://drive.google.com/file/d/12vYeaqGrzbo9aWj-XLi-RaYPK_jjqcYp/view
    - Advise to not use this approach (for compact attributes) – Double discrete log
    - Hierarchical issuance - use a chain similar to PKI
    - Structure preserving signatures – Groth
        - Hierarchical issuance
        - Harder to do threshold
        - Use PS sig at the root
        - Use Groth for subchains
        - P signature schemes
        - Have to reveal all attributes to the root
        - Goal with CCS paper
    - Dmitry – how does performance suffer when you double the curve size
        - You would not need just to double - rather an order of magnitude (~80 times maybe)

    - Good verifiable encryption: [Maria]
        - Camenisch-Shoup (Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. CRYPTO 2003),
        - Cramer-Shoup (R. Cramer and V. Shoup. Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. EUROCRYPT 2002)
        - El-Gamal.
- ZK Online Retail purchase
- Registration
    - In a ZKP way, is this possible?
    - Simple exclusion protocol for RSA accumulators
- Tokens as Credentials – Transferability
    - Different way than revoke existing issue new
- Policies
    - Authz but more broadly
- Vouchers
    - Limited use credentials

26 Nov 2018

Review

1. Hierarchical credentials deep dive
    a. Satisfy requirement to variable length list of attributes. Size of list is unknown at time of key creation.
    b. Jan to have Manu D and Maria review the paper
    c. Maria will attend call next week.
2. Examine Coconut signature scheme, determine needed security proofs. Feel free to comment here.
    a. Based on PS signature scheme + threshold capability
    b. Jan and Manu D submitted paper that has the needed security proofs
        i. Available in a few weeks
    c. Main differential requirement: Non-interactive aggregation
        i. Nice to haves: Constant sized signatures and verification time
3. Bellman vs Libsnark
    a. Why was bellman slower than libsnark
    b. Mike/Lovesh/Dmitry contact bellman library authors to determine what decisions were made with its implementation
4. BBS+ implementation
    a. Jan to contact Rafael about his code and contributing to z-mix

19 Nov 2018

Agenda/Notes

1. [Dmitry K] Hierarchical credentials: attribute `m` is a commitment to some other attributes `m1',m2',...,mt'` so that one can prove knowledge of `mi'` without Issuer knowing the details of these attributes and how many they are. Suggestion: use a curve with >1024-bit prime order `p=lq+1` and commit in `Zp` using generators of order `q`. Question: is it compatible with BBS+ signatures by Camenisch et al.
    a. Security of signature in paper may not be high enough.
    b. Pairing based elliptic curve over a large prime field seems inefficient.
        i. Perhaps a double commitment-scheme
    c. We will read the paper and come back next week with a more detailed analysis.
2. [Lovesh H] Coconut threshold signatures, brief analysis
    a. This paper was presented a while ago at the W3C VCWG. Mike has been looking at it.
    b. Others haven't explored this construction in detail, it seems to be a combination of
        i. Sanders signature
        ii. Signature aggregation
        iii. Shamir secret sharing
    c. How expensive would the threshold key generation be?
    d. Could we use the existing indy revocation?
        i. A pairings-based accumulator created in the threshold way?
    e. [Manu D] The paper itself is pretty sketchy re: security proofs.
        i. It claims PS signatures are secure under LRSW, which is not true.
        ii. Not saying it is insecure, but security proof would have to be done.
        iii. It would be interesting to see the proof of verifiable encryption done during credential issuance
    f. There may be a more full version of the paper floating around somewhere.
    g. Would it be efficient to set up a threshold signature such that the whole ledger issues a credential?
3. It would be beneficial to have an email before the meeting, and a summary email after the meeting. Brent will get with Mike to work this out.
4. If anyone has comments on the crypto-lib/ursa stuff that is happening, please make comments.

Takeaways

1. Read the paper Dmitry posted about hierarchical credentials, be ready for a deep discussion next meeting.

2. Examine Coconut signature scheme, determine needed security proofs. Feel free to comment here.
3. Brent will get with Mike to make sure a prep email goes out before each future meeting.
4. Please add your comments about cryptolib/ursa

12 Nov 2018.

1. [Dmitry K] Presentation from DevCon4 on STARK- and SNARK-friendly hash functions.

5 Nov 2018

1. [Dmitry K] ZKSTARKs enjoyed huge popularity at Ethereum's DevCon4, particularly for side chain integration and scalability. Authors report massive performance increase.
    a. Sharkmimc, libsnark implementation
    b. Use smaller constants to increase performance
    c. Fewer constraints
    d. Linear transformations, 44 rounds
    e. Dmitry's new hash function slides.
2. [Mike L] OpenSSL BN code into libsnark/libstark to help with license issues
    a. Also possibly more secure primality testing
3. [Brent] primality paper https://eprint.iacr.org/2018/749
4. [Lovesh] Shared crypto lib usage in indy-sdk
    a. When crypto-lib officially hits incubation status
5. [Lovesh] Hardware wallet support
    a. Armory
    b. Trezor - Allows extensions
    c. [Dan M] https://blog.bitpay.com/intel-and-bitpay/
    iirc we forked trezor and integrated sgx protection into bitpay. could do a similar pattern with indy clients
6. [Mike L] Could Dmitry's HD Key scheme work for other TE curves
    a. Yes it can, its agnostic as long as the signature scheme is the same

29 Oct 2018

1. [Mike L] Distributed point functions (DPF)
    a. http://www.scs.stanford.edu/~dm/home/papers/corrigan-gibbs:riposte.pdf
    b. Lovesh merge z-mix this week
2. [Mike L] Secure payment key management
    a. HD keys
    b. Syncing keys
3. [Nathan G] COSE and credentials
4. [Mike L] Libsnark

15 Oct 2018

1. [Mike L] Remaining tests for ZKSnarks
2. [Mike L] Range proofs for anonymous payments
    a. Proof of solvency
3. [Mike L] Secure payment key management
    a. HD keys
    b. Syncing keys
4. [Mike L] Distributed point functions (DPF)
    a. http://www.scs.stanford.edu/~dm/home/papers/corrigan-gibbs:riposte.pdf

8 Oct 2018

1. Bank Notes
    a. Need Anonymous and Private Payments
2. [Dmitry K] - Mirror curves
    a. Commitments point multiplications
    b. Proofs of Signatures can be inefficient with bullet proofs
    c. Use another curve that *mirrors* where you can make efficient elliptic curve circuits vs arithmetic circuits
    d. https://github.com/BlockchainCommons/secp256k1/issues/1
    e. https://mobile.twitter.com/pwuille/status/993572063389605889
    f. https://mathoverflow.net/questions/249982/elliptic-curve-related-equivalence-between-fields-of-different-characteristic
3. Range Proofs in ECC
    a. Use BBS+ for all proofs but ranges
    b. Use RSA for ranges (commitments to values)
    c. Include RSA parameters in cred defs
4. https://github.com/tjfoc/gmsm

1 Oct 2018

1. [Mike L] - Bellman Curve
    a. Can we swap curve params? Yes
    b. Just need to adjust hash functions that map to points
    c. Adjust generators too
    d. Circuits could be longer
    e. Libsnark is faster and more predictable
        i. Time to generate proof
        ii. SHA256 is very close to MIMC
2. [Mike L] - Heard back from GMP, low probability for license change
3. [Mike L] - Verifiable Credentials Working Group proposal

    a. Known Weaknesses
        i. Linkability
        ii. No Selective Disclosure
4. [Mike L] - Mirror Symmetry for Elliptic Curves

24 Sept 2018

1. [Mike L] Bank notes
2. [Mike L] Zcash snark library
    a. Using libsnark, benmarked MiMC
        i. MiMC is 7 times faster than SHA-256 but not >20 times as expected.
    b. Potential benchmarks.
3. [Mike L] Set membership proofs
    a. Snarks merkle trees
    b. Accumulators
    c. Public - Managed by the ledger more generic
    d. Private - Managed by entities (revocation registries)
    e. Multiple parties involved
4. [Mike L] Range proofs Elliptic curve options
5. [Brent Z] Elixxir
    a. Secure Communications
    b. Anonymous payments
    c. Consensus every node is doing Secure MPC
    d. Claiming 10K/sec
6. [Mike L] Verifiable Random Functions (VRFs)
    a. The state root hash could do this, but not recommended
7. BigNumber libraries
    a. Extract OpenSSL BigNum to be a standalone

17 Sept 2018

1. [Mike L] Proposal to move all indy-sdk crypto code to crypto lib
    a. Ethereum adopting crypto-lib
    b. [TODO]: Read crypto-lib proposal
2. [Lovesh] Merge Z-mix and crypto-lib
3. [Mike L] Snarks update from Lovesh
4. Banknotes
    a. Brent to do a write up for Mike L and Nathan G to review
5. Agent Authorization Policy
    a. Hart and Dave will review
6. [Lovesh] Sovrin smart contract on EVM (it can be used with Ethereum, RSK, etc)
    a. Credential issuance and verification using Indy
    b. Ethereum would house the keys

    c. DAPP that can access Sovrin
    d. Sovrin Protocol can be used on any blockchain
    e. More info to come next week
7. Mike L to petition GMP authors to have third license: Apache 2.0
    a. We can offer them a logo, always display GMP
    b. Mass adoption with license change
8. Sawtooth is adjusting some items in their code base that Indy could utilize.
    a. Could they eventually merge into one project?
    b. Mike L to ask Nathan George
    c. CC Dave Huseby, Brent, Lovesh
9. Regional based crypto
    a. Chinese