1 参考材料

- `https://zkp.science/` (白皮书比较有价值)
- `https://qed-it.com/2017-12-20-the-incredible-machine/` (浅显的例子)
- `https://www.jianshu.com/p/7b772e5cdaef` (中文综述)

2 文摘

2.1 ZKProof Community Reference

`https://zkproof.org/`

1. ZKP systems involve at least two parties: a prover and a verifier. The goal of the prover is to convince the verifier that a statement is true, without revealing any additional information.
2. A ZKP system can be described with three components: setup, prove, verify. The setup, which can be implemented with various techniques, determines the initial state of the prover and the verifier, including private and common elements. The prove and verify components are the algorithms followed by the prover and verifier, respectively, possibly in an interactive manner. These algorithms are defined so as to ensure three main security requirements: completeness, soundness, and zero-knowledge.
3. Completeness requires that if both prove and verify are correct, and if the statement is true, then at the end of the interaction the prover is convinced of this fact. Soundness requires that not even a malicious prover can convince the verifier of a false statement. Zero knowledge requires that even a malicious verifier cannot extract any information beyond the truthfulness of the given statement
4. One important aspect to consider upfront is the representation of statements. An actual conversion suitable for ZKPs, namely for more complex statements, can pose an implementation challenge. There are nonetheless various techniques that enable converting a statement into a mathematical object, such as a circuit. This document gives special attention to representations based on a Rank-1 constraint system (R1CS) and quadratic arithmetic programs (QAP), which are adopted by several ZKP solutions in use today.

5. Some of these claims (commonly known by the prover and verifier, and here described as informal statements) require a substrate (called instance, also commonly known by the prover and verifier) to support an association with the confidential information (called witness, known by the prover and to not be leaked during the proof process).

6. The theory of ZKPs distinguishes between proofs and arguments, as related to the computational power of the prover and verifier. Proofs need to be sound even against computationally unbounded provers, whereas arguments only need to preserve soundness against computationally bounded provers (often defined as probabilistic polynomial time algorithms). For simplicity, "proof" is used hereafter to designate both proofs and arguments, although there are theoretical circumstances where the distinction can be relevant.

7. A statement is either a membership claim of the form x L, or a knowledge claim of the form "In the scope of relation R, I know a witness for instance x". there are scenarios where a statement of knowledge cannot be converted into a statement of membership, and vice-versa.

8. The relation R can for instance be specified as a program (e.g. in C or Java), which given inputs x and w decides to accept, meaning (x,w) R, or reject, meaning w is not a witness to x L. In the academic literature, relations are often specified either as random access memory (RAM) programs or through Boolean and arithmetic circuits, described below.

9. A rank-1 constraint system (R1CS) is a system of equations represented by a list of triplets ($\vec{a}$,$\vec{b}$,$\vec{c}$) of vectors of elements of some field. Each triplet defines a constraint as an equation of the form (A) · (B) - (C) = 0. Each of the three elements (A), (B), (C) in such equation is a linear combination (e.g., (C) = c1 · s1 + c2 · s2 + ...) of variables si of the so called solution $\vec{s}$ vector.

10. A R1CS does not produce an output from an input (as for example a circuit does), but can be used to verify the correctness of a computation (e.g., performed by circuits with logic and/or arithmetic gates). The R1CS checks that the output variables (commonly known by both prover and verifier) are consistent with all other variables (possibly known only by the prover) in the solution vector. R1CS is only an intermediate representation, since the actual use in a ZKP system requires subsequent formulations (e.g., into a QAP) to enable verification without revealing the secret variables.

11. In some cases, a trusted third party runs an algorithm to generate the setup. In other cases, Setup may be a multi-party computation offering resilience against a subset of corrupt and dishonest parties (and the auxiliary output may represent side-information the adversarial parties learn from the MPC protocol). Yet, another possibility is to work in the plain model, where the setup does nothing but copy a security parameter, e.g., setupP = setupV = k.

12. In the traditional notion of zero-knowledge, a ZKP system prevents the verifier from even being able to convincingly advertise having interacted in a legitimate proof execution. In other words, the verifier cannot transfer onto others the confidence gained about the proven statement. This property is sometimes called deniability or non-transferability, since a prover that has interacted as a legitimate prover in a proof is later able to plausibly deny having done so, even if the original verifier releases the transcript publicly. Transferability means that the verifier in a legitimate proof execution becomes able to convince an external party that the corresponding statement is true.

13. There are several proposals to reduce the trust in the setup such as using secure multi-party computation to generate a CRS, using a multi-string model where there are many CRSs and security only relies on a majority being honestly generated, and subversion resistant CRS where zero-knowledge holds even against a maliciously generated CRS.

2.2 zk-SNARK **ATTACH**

- zero knowledge Succinct Non-interactive ARgument of Knowledge
- ZCash 是最早广泛应用 zk-SNARK 的数字货币。
- ZCash 继承了比特币的交易模型，只不过 UTXO 被衍生出的新概念 “note” 所代替，后者是 ZCash 的基本交易单元。不过，翻译成 “支票” 更贴切，因为每张 note 上都标注了只有谁才能兑现它（即所有者）。一个交易的输入和输出都是若干 note。为描述方便起见，将 note 记为 “note=(PK, v, r)”，其中，PK 是所有者的公钥（地址），v 是金额，而 r 是可以唯一区分该 note 的序列号。
- ZCash 交易分为两类：透明地址和隐藏地址。透明地址交易的输入、输出直接是可见的 note 信息（除了货币单位外，和比特币交易一模一样）。对于隐藏地址交易，输入和/或输出的地址和金额是隐藏的。透明地址和隐藏地址还可以混用。

- 在隐藏地址的交易中，输入、输出不再是明文的 note，而分别是 note 的废止通知和签发通知。
  - 签发通知（note commitment）：作为交易的输出，表示一张新 note 被签发。一个有效的 commitment 是一张 note 存在的证明，然而从它包含的信息中并不知道是哪张 note，也就无法知道所有者是谁，金额多少。为满足这一点，最简单的方法是对 note 的描述信息取哈希，因此 note 对应的 commitment 可以简单描述为 "HASH(note)"；
  - 废止通知（note nullifier）：作为交易的输入，表示一张老支票将作废（因为马上要被兑现、花掉了）。同比特币一样，一个交易的输入一定是另一个交易的输出，因此 nullifier 对应唯一一个 commitment（结合 commitment 的定义，也就唯一对应一张 note），但从它包含的信息并不能推导出是哪个 commitment（如果可以的话，ZCash 交易便可被追踪，因而丧失隐私性了）。为构造满足要求的 nullifier，取哈希依然是个好办法，因此序号为 r 的 note，对应的 nullifier 可描述为 "HASH(r)"。
- 对于 NP 问题，验证它的解是否正确是 "可行的"；而对于 P 问题，更进一步，求出它的解也是可行的。验证和求解的不对称性是密码学应用的基础。
- the rules for determining a valid transaction get transformed into equations that can then be evaluated on a candidate solution without revealing any sensitive information to the parties verifying the equations.
- Computation -> Arithmetic Circuit -> R1CS -> QAP -> zk-SNARK
- a Rank 1 Constraint System, or R1CS, to check that the values are ¡°traveling correctly¡±.
- In this R1CS representation, the verifier has to check many constraints ¡ª one for almost every wire of the circuit. (For technical reasons, it turns out we only have a constraint for wires coming out of multiplication gates.) In a 2012 paper on the topic, Gennaro, Gentry, Parno and Raykova presented a nice way to ¡°bundle all these constraints into one¡±. This method uses a representation of the circuit called a Quadratic Arithmetic Program (QAP). The single constraint that needs to be checked is now between polynomials rather than between numbers. The polynomials can be quite large, but this is alright because when an identity does not hold between polynomials, it will fail to hold at most points. Therefore, you only have to check that the two

polynomials match at one randomly chosen point in order to correctly verify the proof with high probability.

- If the prover knew in advance which point the verifier would choose to check, they might be able to craft polynomials that are invalid, but still satisfy the identity at that point. With zk-SNARKs, sophisticated mathematical techniques such as homomorphic encryption and pairings of elliptic curves are used to evaluate polynomials ¡°blindly¡± ¨C i.e. without knowing which point is being evaluated. The public parameters described above are used to determine which point will be checked, but in encrypted form so that neither the prover nor the verifier know what it is.