How would you explain the trusted setup to a friend?
A computational social contract in which a groups of people agree about an initial set of conditions without the need to trust each other.
A fun ceremony, where multi-party computation gets executed to calculate parameters for executing the zkSNARKS protocol.
A good friend ceremony
A group of people get together online to generate the parameters used to verify and construct private transactions in a zkSNARK enabled cryptocurrency. These participants are relied upon to delete the data they create after the fact as if they don't and all collude, they can print new coins without detection.
a pre-phase where people have to agree on a set of data to make the proof/verification protocol later non-interactive. This setup has to be trusted so no one can fake proofs.
A procedure that waranties that a random seed is generated, used to calculate some keys and the deleted. The security of the zero knowlage system is based in that now body knows the seed (some times called toxic value) that generated the keys.
A rather complicated preparation phase that has to be done at each major update of the network and that has important security implication for the whole network. In practice it is carried out in a very transparent and safe way.
A set of encrypted data, securely computed, that allows to verify the later calculations of the zkSNARKs proofs generation and verification in a non interactive way.
A trusted setup is a set of some random numbers with special properties. They are used to create proofs and verify them so they are shared to the verifier and the prover. During the setup creation, some intermediate values are used to generate the output numbers. The setup is called trusted because one must trust the setup provider that he deleted the intermediate values used to generate this setup. Indeed, with these intermediate values, which are called toxic wastes, it is possible to generate a fake proof.
a trusted setup is like god, something you need to believe in when there isnt another alternative
A trusted setup is where a system requires some trusted parties to set up a set of initial parameters but we have to be able to trust that every party will destroy this set of parameters, so that no one, including themselves, would have access to the parameters after the setup.
A way to generate a key (foundational piece to particular tech) that reduces the risks of what you consider a negative outcome.
A zk-SNARK algo requires a ceremony to generate key parameters for one-time setup of a cryptographic proof (SNARK)
After explaining the Find Waldo analogy for zkSNARKs, I would describe the trusted setup as constructing the board with the hole. If I were to cheat during the setup and tape a picture of waldo behind the hole, and thus always be able to find Waldo.
An efficient solution to privacy concerns we are all confronted with today.
Anonymous secure server activity attestation with TLS and Zeroknowledge
Assuming a non-techie friend, I would say that the trusted setup is a necessary step to make zk-SNARKS to work. It is a way to decide on the initial parameters among a group of participants and we say it is "trusted" because we have to trust that at least 1 participant is honest and will do as expected.
At the very start of a special type of a system that doesn't leak any information, some special artefacts are created that, if not properly destroyed, can compromise the secrets, at any point in the future.
Bootstrapping an engine manually by choosing some of its key parameters that will help it function
Cooperate with someone to create a judge that cannot be fooled.
Create a shared private key, keep public, throw private provably away
Crypto party for ice breaking :)
cryptomagic that allows peoples to verify stuff without revealing what this stuff is about
Depending on the friend: I would first explain what a zk proof is and then explain that in some systems you'd need to do a computation before you can do the zk proofs. It is called trusted setup because whoever has the full result of the computation has the power to create proofs for fake statements.
Dunno what you mean by "trusted setup" but I assume it's the Zcash trusted setup: For Zcash we need a trusted setup phase first, where in theory everyone could participate. We need to generate a secret that must not be revealed at any point in time (otherwise Zcash is worthless). To achieve this everyone adds a snippet to this secret and destroys his/her secret afterwards. (It's way more trick in reality but you know, you're not the brightest light in the harbor. ;)). The process of creation is designed in a way that it is not possible to restore the secret at all, but even if you think there might be a small chance that the process doesn't work you can still participate and make sure you delete your "toxic waste".
Firstly as a secret that is not known to the participants of the protocol. Then I would talk very fast about multi-party computation while waving my hands a lot.
Generally, a trusted setup is a mathematical environment, created by a trusted source, from which participants of a cryptographic process receive some common knowledge necessary to actually perform the process. For zkSNARKs specifically, a trusted setup creates the proving and verification key, allowing the proof to be non-interactive.
Generating a bunch of randomness to create the shared params that privacy preserving currencies use to shield the identities of Alice and Bob in Alice pays Bob scenarios.
Generation of keys to be able to generate snark proofs
Good one ... Think of zk-snarks as a myriad of uniquely matched keys and multi-keyhole locks. In order to create all these keys and locks, we start with a set of master keys, one for each keyhole for the origin/genesis lock. From these masters, we can derive a myriad copies with slight changes so that there is only one set of keys to each keyhole for each corresponding lock making for unique and secure key-lock combos. However, the changes to the masters are so small, that the (combined set of) master keys can open any one of the derived locks. so to ensure security, the master keys must be destroyed and we must trust the entities originally crafting those masters to do that.
Group of people need to compute an unbiased random number with special properties such that all of them would have to collude to bias it.
Hey J., 'trusted setup' is just the name for a implementation not revealing the details to the public, remember when you told you nephew the story with the robber's den with the locked door in the tunnel - just to tell him why you liked zcash for using zkSNARKs, or when you calculated speedy O(n*) runtime for zkSTARKS fibonacci proofs or this crazy TOPOS everyone ist now so excited about.
I can prove something without revealing the solution itseld
I don't know what that is (I am still a cs student, wanting to learn more and go deeper into the field)
I just wish it was gone and I did not have to
I promise I will think of something funny later
I wouldn't. I would say use PRE or FHE. But really... I know there is a ceremony involved and its used in MPC and TEEs but that's about it.
I'm assuming that I'm talking to a reasonably technical friend who already has a rough idea of what a zk proof is. I do not consider myself to have a strong knowledge of the internals of zk-SNARKS, some of this may be wrong. In interactive zk proofs the verifier asks the prover a series of questions about the answer that reveal partial knowledge of the solution until the verifier is satisfied that the prover knows the full answer. In zk-SNARKS we preselect the questions and then encrypt them. The prover uses homomorphic encryption to compute the encrypted answers to the preselected questions and returns them as the proof. The trusted setup is the phase of selecting the questions & encrypting them. It's trusted because if the participants do not throw away the encrypted questions they can easily fake proofs.
If at least one of us is honest - we both are ok
If you trust this one thing, you don't have to ever trust anything else

How would you explain the trusted setup to a friend?
Imagine that you had to make a time capsule that you don't want tampered with by anyone in your generation. You would make the box, fill it with valuable time capsule goodies, and put a lock on it. But what do you do with the key? The key has a lot of information (one could, perhaps, reverse engineer the lock?). What if you want to prove to everyone else that you won't go back on your promise and dig up the time capsule and get the goodies. A trusted setup is basically a ceremony where you create the lock/key pair in a public manner with only 'trusted' users allowed to place the input on the initial key. These users each contribute a little to the key design (I get one nook, you get 3 nooks, she gets 5 nooks), but none of them know enough (hopefully) to recreate the key, reverse engineer the lock, and steam the goodies. The key, by the way, is the 'parameter' here, in parameter selection. [I got the idea for this description from Craig Gentry's PhD thesis on FHE]
Imagine your friend films how you are carrying out cake at a birthday party. And then you hilariously fall and cake is all over the place. You ask your friend to delete this video,you even watch how he does it,however you can't be totally sure it's not already backed up somewhere and the only way you're figure that it is Some time later when you become YouTube star.
In a trusted setup everyone has to trust anyone. In case the private keys are not properly destroyed, the system can be compromised at a later point.
in a very abstract way: Crypto schemes have different phases (setup, keygen, etc...). A scheme could now have a trusted setup phase or an untrusted one. A trusted setup phase here means that you need to trust something to make the system work technically. This is a disadvantage. Schemas that support an untrusted setup are basically more robust, since no trust relation to anything or someone is needed to set up and use the system. Further good crypto reads at https://eprint.iacr.org
In order to setup a zero knowledge proof system for public usage one needs to generate a random number. This random number is only needed once for the setup procedure. It needs to be deleted afterwards because it would allow people that can access it to tamper with the proofs. The setup is called trusted, because you must trust the founders to delete the number.
In some systems like ZCash or POANetwork it is necessary to have a one-time setup of some common parameters of a cryptographic protocol. Trusted setup guarantees that if at least one part of multi-party computation behave honest, than the system is not compromised
In the CRS model you need data both to generate and verify proofs. This data can be generated by the verifier as a challenge to the prover, who has to construct its proof using this data and the witness data from the computation it ran. In a trusted setup this data won't be generated as a challenge by a verifier, but once and for all for everyone in a big MPC. Since knowledge of the paramters that went into the CRS is enough to craft false proofs, you want to distribute this computation to many parties in such a way that a single honest party is enough to ensure the CRS is incorruptible.
In zk-snarks to verify the private transactions there need the initial trusted setup phase initiated with the private key which is then trusted to be destroyed.
increasing confidence in performance of legal obligations is what Lexon is about
Interoperable networking layer for distributed resources and services
It is an one-time setup where the seeding parameters for zero knowledge proof generation/validation are created. It has to be done in such a way that it is impossible to retrieve all the pieces (keys) later on - otherwise it would be possible for someone to read/decypher/compromise the zero knowledge proofs later on.
It is like having a special piggy bank with a lock. During the setup, the founders lock the bank with the key and destroy it, so no one will be able to steal any money from the bank without actually breaking the jar (being detected). If the key is not destroyed , someone could potentially steal money from the bank by opening the lock and closing it back without being detected...
It is the initial parameter generation for a ZKSNARK setup. During this process, one must generate a parameter that links the keys used to create and verify proofs, that if leaked, will allow those in possession of it to trick the verifier. For this reason this initial setup is considered "trusted", as in you must trust the participant(s) to make this parameter irrecoverable. To do so, we've invented several nifty ways to involve many individuals to participate in trusted setup ceremonies in such a way that they must all collude in order for the parameter to be recoverable. STARKS do not require a trusted setup :)
It is time window when people could exchange secrets before they exchanging any zero-knowledge based transactions.
It was necessary to make ZCash happen :)
It's a ceremony where a group of people found a cult promising privacy or something similar.
It's a crypto setup process when a system needs some value B to operate and this B needs to be computed from A in a specific way. But if anyone knows A - the system is broken. That's why they use MPC to compute B out of pieces of A. That way if at least one party is honest and actually destroys their part of A - A never existed in full. Unless someone screwed up and logs were too verbose ;)
It's a one-way process to generate a piece of information that acts as a sort of shortcut to checking a math problem. The process has an information by-product that is actually a bit more 'powerful', and would provide easy solutions to the problem. Because we want to make sure that people actually have to solve the problem, the information is generated in a way where everyone involved would have to be cheating in order for the by-product to leak.
It's a process by a group of people, as long as one person destroy his/her toxic waste. The system is trustable
It's a simplifying assumption, where you assume all participants have access to a "string of random bits in the sky". In traditional computer science, this kind of assumption is usually realistic and easy to make (because of the wide availability of practical pseudorandomness) and thus many cryptographic solutions use this assumption as well. However, because cryptography is actually adversarial, those pseudorandom solutions tend to be much trickier to obtain in practice in crypto solutions, so this assumption turned out to be a huge pain in the ass (even immortalized in a Radiolab episode featuring Zooko.) So right now many researchers are trying to get rid of this assumption. (My friends tend to be pretty savvy about algorithms :))
It's like a seed planting
It's like putting money into a trusted bank and you expect the bank will calculate the interests for you correctly
It's the most dangerous phase of the process, when information that is needed for the system to work is derived from a common secret, that should be protected lest it becomes possible to eavesdrop/forge further communications.
It's so complicated not even zcash got it right the first time ;)
just listen to the radiolab podcast episode haha
Let's try with colors (improvising here :)) A community needs to agree on a color to build a publicly-available-but-impossible-to-reproduce color-filter. Thus, this color needs to be created to build the filter but known to none. Ppl trusted to build this filter will each commit a drop of a paint of the color of their choice in a black box, the mix of all paints is used to build the filter. If only one of the person is honest and trash out his original and unique paint, it will be impossible to create the same color and reproduce the filter.
Like a hazing experience we must go through to get connected but forget about and make sure no one (not even ourselves) must know about.
Magic which creates secure randomness
Many people come together to create a secret that will be revealed to none of the induviduals. The only way it can be compromised is that all of the people collaborate.
mpc for starters
necessary procedure for zk SNARKs to set the initial parameters. "Trusted" since the keys used to do so need to be destroyed for the network to be trustless
No idea
Non-interactive zero-knowledge proofs rely on a set of public parameters in order to allow anyone to compute as well as verify proofs of private transactions. Those public parameters need to be established using some random numbers, which need to be deleted afterwards (otherwise they would allow for a back-door for fraudulently proving wrong facts/transactions). Thus there is trust to be placed in the fact that setup phase was conducted correctly and random numbers have in deed been deleted. Therefore multi-party computations, which allow for the random number generation to be derived from many different random numbers (from multiple parties), while the process has integrety as long as at least 1 party involved can be trusted to properly delete their random numbers after the ceremony.
nuclear waste that makes you glow
Number of independent people run piece of code to generate the secure key. The generation requires each party to come up with a secret number and use that number to generate piece of secure key. Caveat is that these secret numbers if put together will result in ability to counterfeit the encryptions we use the secure key. This is why we need to trust that at least one person will destroy their secret.
One time generation of protocol parameters which involves knowledge of or learning of some secrets that can be used to break the guarantees of the protocol

How would you explain the trusted setup to a friend?
Part of the setup creates some "toxic waste" distributed across a number of people that causes the entire system to fall apart if someone gets their hand on all of the pieces. However, as long as one of the participants was honest, and deleted their section, we're safe!
probably by suggesting the radiolab about it - http://www.radiolab.org/story/ceremony/
Probably one of the craziest security ceremonies to ever go down, and an endless well of tweetable conspiracy theory content.
Provers and verifiers need to use a set of keys to generate and verify solutions for a given constraint system - which is generated as part of the common reference string. This refers to choosing a source of entropy that would enable the prover to create points of evaluation of the linear combination of polynomials making up the constraint system without actual giving up the underlying values, and for the verifier to utilise a elliptic curve (a cryptographic primitive) to provide the appropriate points in the trusted setup where the polynomials should be evaluated, without giving up the actual secret (which could enable the creation of fraud proofs).
proving a statement usually goes in a interactive challenge-response fashion. To make it non-interactive you can refer to a common shared information but this information should be encrypted. It has to be set up in a trusted way.
Say you want to be able to prove to me that you know the solution to a "where's wally" puzzle without giving away the solution. You will be able to do that by generating and sending me a Zero Knowledge Proof which I will be able to verify without learning any new information about your solution. Before you can build such a proof, we will need to setup the system so as to generate a few parameters that will be used for you to generate the proof ("proving key") and for me to verify it ("verification key"). The challenge is that when we run this setup, we will also generate some data (sometimes called "toxic waste") which, if you happen to store this data, will let you generate fake proofs. For this reason, it's important that we run this setup in such a way that everyone is confident that the toxic waste data got deleted. One way to achieve that is to run a multi-party computation wherein if at least one participant acts honestly (i.e. deletes its part of the data), then you are guaranteed that the resulting "toxic waste" cannot be recovered, making the setup "trusted:
Secure and decentralized ; only you hold and know your keys
So I told you about all the cool things you can do with SNARKs, but there's a catch. At the very beginning, there is something called trusted setup. It's some computation that only needs to be do once, but whoever does it will be able to break the system - without anyone noticing. Currently, we try to alleviate that by distributing it to multiple parties. This way we strengthen trust in the system, because now we have multiple parties that are dishonest and collude, instead of just one dishonest one.
Some cryptographic protocols require shared knowledge to exist between participants of the protocol. Generating this preliminary shared knowledge is referred to as a setup. The derivation of this shared knowledge leads to generation of information that needs to be forgotten afterwards to guarantee the protocol's security. Since "forgetting" cannot be proven, trust in the fact that the party deriving the shared knowledge forgot the information is required.
Some of the cryptographic schemas need to have one time setup (ZK-SNARKS for example). It is called trusted because you have to trust the person or group of people who are involved in the setup. Typically during the setup they will have access to parameters of the system which would allow producing false proves in the future. The recommended way for dealing with these parameters is destroying them but it is very difficult to prove that someone didn't make a copy. In the state of the art cryptographic systems we use schema called multiparty computation. In this schema no one knows value of the parameters for the setup, the parameters are created jointly by all participants. It is enough if at least one participant is honest to guarantee security of the system.
Some protocols require numbers with special properties but our algorithms that create those properties also create auxiliary information that makes the protocol insecure. A trusted setup is an algorithm that makes this auxiliary information less dangerous, normally by making it safer to destroy. As an example we sometimes need a number that is a product of two prime numbers but where nobody knows the primes.
Some ZKP schemes involve an initial setup step where (for a given circuit) two keys are generated: A "proving key" needed to construct proofs and a "verification key" used to verify those proofs. This process is randomised; it is essential that some of the random input values that go into the setup ("toxic waste") are discarded, otherwise it is possible to construct false proofs that would still verify. Therefore the actor or group of actors need to be trusted to get rid of the "toxic waste".
That awkward thing that STARKs don't need.
The "trusted setup" is the setup phase of zero-knowledge-SNARKs. In the setup, some parameters are created that are needed for the proofs. For example, if a proofer wants provide a proof about something to a verifier, the proof is computed based on these parameters. The setup is called "trusted", because the party creating (or lets say "computing") the parameters is using a secret value to compute them. Knowing the secret value enables to create even more parameters. If these additional parameters are not known to everyone after setup phase, the party knowing them could create proofs that appear to be valid although they are invalid. Therefore, it has to be ensured that no additional parameters can be created after the setup phase. For this, everyone has to trust the party knowing the secret value to not make use of this secret value anymore, or, ideally, to delete it. This sounds like a lot of trust into one party. Therefore, there is exists also a more complex version of it, in which several parties choose a secrete value, and each party knows only its own secret value. To compute some parameters, all secret values have to be used. And if at least one of the parties deletes its secret value after the setup phase, then no additional parameters can be computed anymore.
The "trusted setup" means you have to simply trust that the initial conditions were not compromised to be able to trust the system as a whole. In a more secure system, you can trust the operation of the system without having to trust the initial setup.
The ceremony to launch a chain that requires a small group of people to be honest. But not as scary as it sounds. All members of this small group would have to collude to do any damage. Fortunately, STARKs will make the trusted setup unnecessary.
The locksmith gave me and my friend a lock which I can lock and my friend can unlock and we trust the locksmith to have destroyed the sketches/models of both the keys.
The one that you either must control entirely, or be aware of potential betrayal of various kinds every moment.
The only trusted part required of some trustless ZKp systems during which the secret to be encrypted is unencrypted.
The protocol requires pubilc and private keys similar to signature keys, but nobody must know the private part.
The spell used by sorcerers to create the magical Box of Wisdom
The trusted setup (for ZK-SNARKs) is the initial phase of a protocol in which a verification & proving key are generated from some randomly sampled values. These random values must be kept secret (i.e. disposed) -- otherwise, provers would be able to generate "fake" proofs.
The trusted setup acts like the lookup tables that old 8-bit micros used to cut corners when performing arithmetic, except this time the table contains secrets
The trusted setup creates the shared assumptions (acts as a framework) which allow for cheaper zero-knowledge proofs. It's also possible without trusted setup but much more compute expensive.
The trusted setup generates magic numbers and toxic waste. The magic numbers are linked to statements, usually involving encrypted numbers, via zero-knowledge proofs (e.g. I have an encrypted number that is less than 100). If you know what the toxic waste is, you can generate fake magic numbers and create fake statements (e.g. you can convince people that "I have an encrypted number that is less than 100", when that is not the case). A trusted setup relies on the toxic waste being destroyed by trusted parties
The trusted setup is a multi-party ceremony to generate trusted parameters for certain zero-knowledge constructions like ZK-SNARKS.
The trusted setup is a process for generating public parameters that are later used in the creation of zkSNARK proofs. If such a setup is compromised, i.e., an adversary learns the secret randomness to generate these parameters, fake zkSNARKS proofs can be created. In the case of blockchain (e.g., Zcash), this means that invalid transactions are deemed valid. That is why a trusted setup is usually based on a multi-party computation that is only compromised if the private randomness of every participant is obtained.
The trusted setup is a process in which multiple independent parties contribute a secret share towards the generation of a bunch of numbers, which are subsequently used for the creation and verification of zero-knowledge proofs. Crucially, none of the participants learns anything about the share of other parties and if only a single participant is honest, the trusted setup is considered sound. Thus, with increasing number of participants, even people who have participated in the trusted setup can be increasingly certain that the generation procedure is safe.
The trusted setup is a process to generate a pair of proving and verification keys that are used later for proof for a specific problem.
The trusted setup is a set of common parameters we need to established before using some cryptographic schemes. They are trusted because we need to make sure no one can use the way they were made to cheat the system
The trusted setup is the first round in a SNARK. It basically creates a public/private keypair that is used to create and check the proof. Since the proof can be created and checked with only the public part, the private part can be destroyed. Moreover, the proof can be verified by someone who did not take part in the trusted setup (-> blockchain). Since the private part can be used to create fake proofs, the trusted setup is usually done in a multiparty computation setting where you can only create fake proofs if you obtain the private part from all the participants.

How would you explain the trusted setup to a friend?
The Zero-Knowledge Proving System of Zcash required a one-time setup of its new cryptographic proof engine called zk-SNARKs (SNARKS) in its Genesis block. The "Trusted Setup" uses the initial parameters that were generated for SNARKs to create keys (linked with a hidden parameter) and are used to provide it with the ability to create proofs and to verify those proofs associated with private transactions, and then add the transaction to the blockchain without revealing the sender, receiver, or the amount associated with a transaction. It requires trusting whoever generated the keys to destroy the secret keys associated with that transaction after the transaction to prevent them being used to forge future transactions. zk-STARKs (STARKS) has solved the "Trusted Setup" problem
There is a lock that anyone should be able to lock their stuff safely. But the key which can lock(and unlock) has to be designed by somebody. Instead of letting one person designg the whole key, so they can keep the blueprint and potentially unlock everything, we form a trusted set of people, where each of them design only a part of the key. In this way, even if only one person honestly burn the blueprint, the whole key cannnot be remanufactured so the key designers cannot unlock the boxes.
There needs to be a secret number that is used for a calculation and then destroyed. If it isn't destroyed properly, anyone who knows it can make fake proofs. Trusting one person to use and then destroy the secret is too risky, so a team does it and if even one of the team-members destroy's their secret, the ceremony is successful.
There's this parameter used in the system called 'toxic waste' which we need to generate as a group, but nobody can know what the value actually is. If anyone finds out, the system can be easily exploited. For the trusted setup, each of the members of the group contributes a randomly generated number and all the inputs are mixed together with some cryptography to produce the toxic waste. As long as one person doesn't cheat and reveal/check their input, we can be sure that nobody will ever know the toxic waste (so the system will be secure!).
To a non-crypto friend I suppose, who saw The Matrix: it's that problem why Agent Smith tortures Morpheus to get at the keys of Zion. Much better no-one has such important keys but only the machines themselves, but except for Smith. Concretely, a ZK blockchain is a chain of signatures and that needs to start somewhere. The chain needs to be anchored in a starting point, but the keys need to be thrown away, like literally, so no-one could ever unmoor that anchor, which would allow you to sign untrue facts, which would basically allow you to forge checks and cash in the money for them. The Trusted Setup then is all about how to make sure no-one can secretly keep the keys and become even richer than everyone else. In effect, some crypto is used that let's the keys be split and the hope is that most participants in the ritual will be honest and destroy their part of the keys as promised.
To a non-technical friend: in a zero-knowledge proof, the prover may need to use some values to generate the proof. Those values need to be created, sent encrypted and destroyed. All this process is what a trusted setup does. We call it trusted because we have to trust that the setup does all three steps properly. Then the prover works with encrypted values to generate the proof, he does not need the original values (him and no one must know about them!). A bit technical friend: super roughly speaking, the interactive zk proofs, we get rid of the interaction by leting a trusted setup provide the challanges.
To do zk-SNARKS we need a public key noone knows the private key of, but you need to do some calculations with the private key. The trusted setup makes sure the private key gets deleted by distributing the task to many people. If at least one of these people was honest the setup is safe.
To gain trust in a friendship and be able to exchange secrets, both have to go at some point through a trust-building process. The outcome should, though, never be revealed to anybody outside of the friendship. Why share party details with the public?
To use SNARKs, you need to get a bunch of people to mix their individual secret randomness so that you end up with some randomness that no one fully has the secret information for
Trust and use it. Nothing more.
Trusted setup is like a card deck printing machine. If the deck was printed correctly and the printer is discarded post-printing, then you're good, but if someone kept the printer, they could print new cards in the deck whenever they want. One way to mitigate this is to only give a piece of the machine to each participant (the MPC ceremony).
Trusting the person who generated the keys, on the basis that they deleted them after to remove the 'toxic waste'. If they don't delete the keys afterwards, they can create unlimited fake transactions and essentially create unlimited Zcash undetected.
Use ethereum and see the transaction mapping
Very difficult, long and exhaustive procedure required to create a set of parameters for zero knowledge protocols. The point is that no particular participant of trusted setup will possess any secret values, which may help him to break Zero-Knowledge schemes.
via a rental application or job application, where you don't want to share exact details
We cast a spell to prevent lying about money. You can only lie if you know all the magic words. As part of the spell, we had dozens of people make magic words and throw them away, so hopefully no one can get them all
We have a function s.t y=f(x). I want to prove that I know x without actually revealing x given y and f. For the proof to be of small size, I need to to create a circuit for computing the function f with certain public values which is a function of some secret parameters. If the secret parameters are known to anyone, it is possible for him/her to convince anyone that they know x without actually knowing x. Hence, the secret parameters cannot be known to anyone. To achieve that, we perform a ceremony, in which n people provide random values as input and the output is the public parameters of the circuit. Even if one of the people generated the random value honestly (without colluding with other participants), the secret parameters can be never known to the world and nobody can generate fake proofs. This ceremony is called the trusted setup and is used in cryptocurrencies like ZCash. In the context of ZCash, if anyone compromised the trusted setup, then they can effectively create money out of thin air. That's why its extremely important to get the trusted setup right with minimal trust assumptions among the n people.
When 2 parties must not trust each other but use math to proof that what they are doing is correct. -> enables business
When teaching about ZKP systems, and it comes time to explain the trusted set up, the friend has already been briefed on the homomorphic hiding properties of polynomials and their evaluations at hidden points. It functions as a way to solve the chicken-and-egg problem inherent in the conversation between the prover and the verifier, and sets the scene for the game being played between the two parties. The output of the trusted set up is like a forbidden secret property, a magic proportion, designed into the game itself, such that if a player of the game knew this "toxic" secret , they could cheat their opponent at will. Locking away this secret forever, by destroying it after the game's inception, allows the game to forever enforce fair play between both players. For this reason it is safest for the game to be conceived not by one game designer but by many, such that all shards of this forbidden secret have a minimal chance of forming together again, as they did originally to set the game. This happens so long as at least one game designer throws away their contribution to the forbidden proportion.
When using SNARKS to verify transactions & add them to the blockchain without revealing any details to the public, you need to use some sort of one time setup. Thus “Trusted Setup” is the fact of having to trust the generator of the keys destroyed them after inception.
When you move into a new house, do you change the locks? What if the locks can't be changed...
You ask a locksmith to install a door lock on the front door of your house, in which you keep several Picasso paintings. After the lock is installed and you have received several copies of the keys, you must trust the locksmith to destroy the key pattern, so that he cannot make any new keys. That is a type of trusted setup. In the key generation process of a zero-knowledge proof setup, the party generating the keys will use a certain secret value which needs to be destroyed after the keys are generated - this secret value is like the key pattern. If that value is not destroyed, it can be used to forge a proof.
You can pay, in ether, to be cryptographically certain that your access management policies will be respected even after you go offline.
You participate in a communal ritual in which a wizard with a phd in arcane cryptography instructs you what to do, what to keep and what to destroy. You might be able to get a good youtube video out of it blowtorching your hardware. In the end, some magic numbers are created and are used to prove somehow that nobody is creating money from thin air - more black magic. This is all true unless some smart wizard finds an issue and all artifacts of the ritual have to be destroyed in order to prevent more out of thin air money printing.
You want to create randomness and you want to be sure that there is not a single party giving you the ""randomness"" (which may or may not be random for him/her). So you combine the randomness of different parties you trust not to collude.
Zero Knowledge Proofs are like a sudoku. They are for the proofer and verifier fun.
Zero-knowledge proofs is a cryptographic technique. Loosely speaking, it allows one party, the prover, to convince the other party, the verifier, that the prover knows the secret, without revealing it. In the recent years, this technique found various applications to real world problems. There are different types of zero-knowledge proofs. One of them, called SNARKs, is amazingly efficient. No matter how big the secret is, the proof is always of the same (tiny) size. But this compactness comes with a downside: to communicate so efficiently, the prover and the verifier should both have access to what is called Common Reference String (CRS), a piece of data, that, somehow describes what kind of secrets they are dealing with. If this string is created unfaithfully, the prover may get the ability to cheat, presenting fake secrets to the verifier. In some cases, for example when the CRS is generated by the verifier, it is not a problem (it has no reason to fool itself). In others -- take zCash, where each node is a verifier, and each wallet is a prover -- things become more complicated, as some verifier may appear malicious. So the CRS has to be generated by a “trusted third party”, hence the name -- “trusted setup”. In practice, however, such an entity rarely exists. As a solution the setup is performed collectively by many interested parties in a way that guarantees that if at least one of them follows the procedure faithfully, fake proofs are infeasible to create.

How would you explain the trusted setup to a friend?