

Forum:[Hack Forums](#)

Thread: [Vanity Cryptocurrency Address Generation | Any Currency | Ultra Secure](#)

Orca - September 15, 2019, 3:39 am

<https://vanitycrypto.com/>

Turbo Generator

Quickly and easily generate a cryptocurrency address right here in your web browser without any downloads or payment information! With this tool you can quickly generate a custom bitcoin wallet address, and many more! Use this for your bitcoin wallet.

Kill Joy - September 16, 2019, 6:56 am

Please fix the link in the OP.

Users be aware that a 3rd party is generating the private key for any address given out. There are safer methods for doing this.

Orca - September 16, 2019, 2:36 pm

(Sep 16, 2019 06:56 AM)Kill Joy Wrote: [\[CITE\]](#) Please fix the link in the OP.

Users be aware that a 3rd party is generating the private key for any address given out. There are safer methods for doing this.

Hi Kill Joy,

Thank you for notifying me to the broken link.

I appreciate the warning, but with our methods we never actually get access to your private keys. We use a split-key system - there is no safer method for doing this. ?

Orca - September 16, 2019, 2:50 pm

(Sep 16, 2019 02:44 PM)M0HX Wrote: [\[CITE\]](#) there is. offline bruh.

Offline is not a service - nor would it provide any more security than a cryptographically secure split-key generation

cryp - September 16, 2019, 2:51 pm

DO NOT USE THIS. It is a public 3rd party wallet generation site.

USE VANITYGEN. OFFLINE.

<https://en.bitcoin.it/wiki/Vanitygen>

<https://github.com/samr7/vanitygen>

Orca - September 16, 2019, 2:54 pm

Thank you for the feedback, but why do you recommend they not use it if we have a secure split-key implementation?

Surely you are referring to services that do not use this feature. ?

XMR - September 16, 2019, 9:03 pm

(Sep 16, 2019 02:54 PM)Orca Wrote: [\[CITE\]](#) Thank you for the feedback, but why do you

recommend they not use it if we have a secure split-key implementation?

Surely you are referring to services that do not use this feature. ?

There is nothing stopping you from generating the real private key, saving the real one, then encrypting it with the user's public key and sending them that. There is no known secure way to have someone else generate private keys for you, because you should never trust someone else with your private keys.

Anyone reading: do not trust any online service with your keys. If you can't generate your own vanity addresses, you're better off without them.

(Sep 16, 2019 02:50 PM)Orca Wrote: [\[CITE\]](#) Offline is not a service - nor would it provide any more security than a cryptographically secure split-key generation

Actually yes it would be more secure. It is extremely easy to backdoor a website or implement something like what I've outlined above. It's a lot harder to backdoor a specific computer, let alone the thousands of computers who would be generating their own addresses rather than trusting an online service to do it for them. At that point, you're practically asking to get robbed.

Orca - September 16, 2019, 9:15 pm

(Sep 16, 2019 09:03 PM)XMR Wrote: [\[CITE\]](#) There is nothing stopping you from generating the real private key, saving the real one, then encrypting it with the user's public key and sending them that. There is no known secure way to have someone else generate private keys for you, because you should never trust someone else with your private keys.

Anyone reading: do not trust any online service with your keys. If you can't generate your own vanity addresses, you're better off without them.

Actually yes it would be more secure. It is extremely easy to backdoor a website or implement something like what I've outlined above. It's a lot harder to backdoor a specific computer, let alone the thousands of computers who would be generating their own addresses rather than trusting an online service to do it for them. At that point, you're practically asking to get robbed.

Hi XMR,

Thank you for the feedback. While your claims may be true for non split-key generation that is not the case for split-key generation.

When using split-key generation, we never even have access to your keys so backdooring is mathematically impossible.

We will be providing more information on split-key on our website soon. ?

JoeJonas - September 16, 2019, 9:25 pm

ya op sorry but no one ever is using this as takes 2 mins for you to steal a users bitcoin. even with your "split key tech"

Orca - September 16, 2019, 9:26 pm

(Sep 16, 2019 09:25 PM)JoeJonas Wrote: [\[CITE\]](#) ya op sorry but no one ever is using this as takes 2 mins for you to steal a users bitcoin. even with your "split key tech"

We have thousands of people using it. And no it's not possible for us to steal your key.

Thanks for the feedback ?

Kill Joy - September 17, 2019, 3:41 am

Split-key is way safer but I will look into how you are doing it myself later and may redact what I said.

Orca - September 17, 2019, 3:43 am

(Sep 17, 2019 03:41 AM)Kill Joy Wrote: [\[CITE\]](#) Split-key is way safer but I will look into how you are doing it myself later and may redact what I said.

Much appreciated ? ?

XMR - September 17, 2019, 8:38 pm

(Sep 16, 2019 09:15 PM)Orca Wrote: [\[CITE\]](#) Hi XMR,

Thank you for the feedback. While your claims may be true for non split-key generation that is not the case for split-key generation.

When using split-key generation, we never even have access to your keys so backdooring is mathematically impossible.

We will be providing more information on split-key on our website soon. ?

I don't think you understand. Split key generation still has you generate the private key. You then encrypt it with the user's public key. Between these steps, you have access to the unencrypted private key. Nothing is stopping you from saving it and sending the user the encrypted one.

Again, if you can't generate your own vanity addresses, you're better off without one.

Orca - September 17, 2019, 8:52 pm

(Sep 17, 2019 08:38 PM)XMR Wrote: [\[CITE\]](#) I don't think you understand. Split key generation still has you generate the private key. You then encrypt it with the user's public key. Between these steps, you have access to the unencrypted private key. Nothing is stopping you from saving it and sending the user the encrypted one.

Again, if you can't generate your own vanity addresses, you're better off without one.

Hi XMR,

That is incorrect and not how split-key works.

With split key we never have access to your private key.

The process you described is called SSL, and we use that also by default.

Best,

Nick

IOlwtf - September 18, 2019, 6:55 am

(Sep 17, 2019 08:52 PM)Orca Wrote: [\[CITE\]](#) The process you described is called SSL, and we use that also by default.

SSL no longer exists. There is TLS.

XMR - September 18, 2019, 8:54 am

(Sep 17, 2019 08:52 PM)Orca Wrote: [\[CITE\]](#) Hi XMR,

That is incorrect and not how split-key works.

With split key we never have access to your private key.

The process you described is called SSL, and we use that also by default.

Best,
Nick

If you instantly generated an encrypted key, you wouldn't know if it matched the user's desired pattern because it would produce an entirely different address. This means you would not know which one of the millions upon millions of keys you generate to send to the user. Split key is more secure than regular generation, but only if one can guarantee the other party is using an unmodified version of the software. The service itself certainly cannot guarantee this if it isn't the case that nobody at all can.

In order to know which key matches the user's pattern, you must have an unencrypted version of the key to check its derived address. This is the point of failure. You have the address' unencrypted private key. You then encrypt it with the public key supplied by the user and then send it to them. It prevents an MITM attack, but nothing stops you from saving the unencrypted key before sending it off.

What I described is not even related to SSL/TLS. TLS has nothing to do with cryptocurrency address generation and is not used to encrypt data that the service isn't supposed to see. TLS is meant to encrypt data so it can't be modified or read by third parties. The service is not a third party. The goal of split-key generation is to ensure that only the end user can see the private key. This is not (yet) possible as there are no known methods to do this. The service can see everything before it's encrypted, including private keys for bitcoin addresses (despite the claim of secure split-key) and the web pages before they're transmitted.

Orca - September 18, 2019, 9:29 am

(Sep 18, 2019 08:54 AM)XMR Wrote: [\[CITE\]](#) If you instantly generated an encrypted key, you wouldn't know if it matched the user's desired pattern because it would produce an entirely different address. This means you would not know which one of the millions upon millions of keys you generate to send to the user. Split key is more secure than regular generation, but only if one can guarantee the other party is using an unmodified version of the software. The service itself certainly cannot guarantee this if it isn't the case that nobody at all can.

In order to know which key matches the user's pattern, you must have an unencrypted version of the key to check its derived address. This is the point of failure. You have the address' unencrypted private key. You then encrypt it with the public key supplied by the user and then send it to them. It prevents an MITM attack, but nothing stops you from saving the unencrypted key before sending it off.

What I described is not even related to SSL/TLS. TLS has nothing to do with cryptocurrency address generation and is not used to encrypt data that the service isn't supposed to see. TLS is meant to encrypt data so it can't be modified or read by third parties. The service is not a third party. The goal of split-key generation is to ensure that only the end user can see the private key. This is not (yet) possible as there are no known methods to do this. The service can see everything before it's encrypted, including private keys for bitcoin addresses (despite the claim of secure split-key) and the web pages before they're transmitted.

Hi XMR,

It still appears that you fundamentally do not understand split-key.

And yes what you described is SSL/TLS - the concept of encrypting communication between website and visitor. Split-Key is not this.

Split-Key guarantees, by math, that the only person who will have access to the key is the user.

Best,
Nick

XMR - September 19, 2019, 6:48 pm

(Sep 18, 2019 09:29 AM)Orca Wrote: [\[CITE\]](#) Hi XMR,

It still appears that you fundamentally do not understand split-key.

And yes what you described is SSL/TLS - the concept of encrypting communication between website and visitor. Split-Key is not this.

Split-Key guarantees, by math, that the only person who will have access to the key is the user.

Best,
Nick

You make baseless claims and keep repeating the same thing over and over with the gist being "split key ensures only the end user will have access." You cannot guarantee this. There is no (known) mathematical system that does this and there's a good chance that such a guarantee is impossible.

Quote:And yes what you described is SSL/TLS - the concept of encrypting communication between website and visitor. Split-Key is not this.

That isn't what I said at all.

Quote:The goal of split-key generation is to ensure that only the end user can see the private key. Split key is basic public key cryptography. You need an unencrypted copy to encrypt with the end user's public key. You need an unencrypted copy to check if the address matches the end user's desired pattern. Without an unencrypted copy, you can't check this because it would produce an entirely different address.

Public key cryptography requires plaintext data and a public key for the service to encrypt with. If you already have an encrypted key then split key isn't necessary.

Quote:Split-Key guarantees, by math, that the only person who will have access to the key is the user.

Then you should have no problem posting mathematical proof of split key being infallible. Here, infallible is defined as only the end user having access to the key. I'll wait.

Orca - September 19, 2019, 8:43 pm

(Sep 19, 2019 06:48 PM)XMR Wrote: [\[CITE\]](#) You make baseless claims and keep repeating the same thing over and over with the gist being "split key ensures only the end user will have access." You cannot guarantee this. There is no (known) mathematical system that does this and there's a good chance that such a guarantee is impossible.

That isn't what I said at all.

Split key is basic public key cryptography. You need an unencrypted copy to encrypt with the end user's public key. You need an unencrypted copy to check if the address matches the end user's

desired pattern. Without an unencrypted copy, you can't check this because it would produce an entirely different address.

Public key cryptography requires plaintext data and a public key for the service to encrypt with. If you already have an encrypted key then split key isn't necessary.

Then you should have no problem posting mathematical proof of split key being infallible. Here, infallible is defined as only the end user having access to the key. I'll wait.

Hi XMR,

Again you seem to be stuck grasping the concept of split-key. Split-key is not encrypted messaging.

I have attached a few resources that may be able to help you get an understanding as you don't seem to want to listen to the facts when I provide them.

<https://bitcoin.stackexchange.com/questions/3853/can-one-safely-buy-vanity-addresses-from-a-third-party-without-risking-ones-coi>

There are in depth mathematical breakdowns that might clear things up, but I doubt it as you are having a conceptual problem rather than mathematical one.

Best,
Nick

RUMRUN - September 19, 2019, 9:50 pm

I highly discourage using this.

Orca - September 19, 2019, 10:29 pm

(Sep 19, 2019 09:50 PM)exec_sh Wrote: [\[CITE\]](#) I highly discourage using this.

Hi Exec_SH,

Why is that? If there is anything you are concerned about, we can address that to better suit your needs.

Best,
Nick

RUMRUN - September 19, 2019, 10:47 pm

High risk with having these tools online, in a browser.

It shouldn't be there.

There's a vanity CLI for this purpose.

Nothing against your site, just a bad idea in general.

335i - September 19, 2019, 10:50 pm

I'm curious how you're monetizing this? What's the incentive for you in providing this service? I don't see any ads or anything like that.

Other than that it's pretty cool, looks like OP knows his shit despite everybody dogging him for this.

Orca - September 19, 2019, 10:51 pm

(Sep 19, 2019 10:47 PM)exec_sh Wrote: [\[CITE\]](#) High risk with having these tools online, in a browser.

It shouldn't be there.

There's a vanity CLI for this purpose.

Nothing against your site, just a bad idea in general.

I appreciate your concern, but did you see our split-key feature?

(Sep 19, 2019 10:50 PM)335i Wrote: [\[CITE\]](#) I'm curious how you're monetizing this? What's the incentive for you in providing this service? I don't see any ads or anything like that.

Other than that it's pretty cool, looks like OP knows his Clay Davis despite everybody dogging him for this.

Thank you for the feedback! I appreciate the objectivity.

We charge for addresses that require heavy computational power, in addition to charging for our API access.

SIM-2015410 - September 20, 2019, 6:20 am

I do not endorse this website nor trust it at all. When I typed something into the public key box to test the split key method, it ignored it and just generated a normal keypair.

(Sep 19, 2019 06:48 PM)XMR Wrote: [\[CITE\]](#) You make baseless claims and keep repeating the same thing over and over with the gist being "split key ensures only the end user will have access." You cannot guarantee this. There is no (known) mathematical system that does this and there's a good chance that such a guarantee is impossible.

Split key is basic public key cryptography. You need an unencrypted copy to encrypt with the end user's public key. You need an unencrypted copy to check if the address matches the end user's desired pattern. Without an unencrypted copy, you can't check this because it would produce an entirely different address.

Public key cryptography requires plaintext data and a public key for the service to encrypt with. If you already have an encrypted key then split key isn't necessary.

Then you should have no problem posting mathematical proof of split key being infallible. Here, infallible is defined as only the end user having access to the key. I'll wait.

You are incorrect. The OP is actually correct in saying that split key generation is possible. Keypairs on elliptic curves are linear so $(\text{private} + \text{private}) * G = (\text{public} + \text{public})$. The site only needs to know the public key to test the address. I do not trust the site or its implementation of this idea but I am simply saying it is possible and you misunderstand how it works.

XMR - September 20, 2019, 9:45 am

(Sep 20, 2019 06:20 AM)MeshCollider Wrote: [\[CITE\]](#) I do not endorse this website nor trust it at all. When I typed something into the public key box to test the split key method, it ignored it and just generated a normal keypair.

You are incorrect. The OP is actually correct in saying that split key generation is possible. Keypairs on elliptic curves are linear so $(\text{private} + \text{private}) * G = (\text{public} + \text{public})$. The site only needs to know the public key to test the address. I do not trust the site or its implementation of this idea but I am simply saying it is possible and you misunderstand how it works.

Okay, so I've been proven wrong and split key is possible. However, it seems as though it would only work if the site properly implements it without some type of foul play. But how do we know the site has properly implemented it (according to your post, they haven't) or don't have some type of trick? As far as I know, that can't be guaranteed. You've shown that they can test the address for a match without its private key, but can we guarantee that they're not saving the real private key before they encrypt it?

Kill Joy - September 20, 2019, 11:22 am

(Sep 20, 2019 09:45 AM)XMR Wrote: [\[CITE\]](#) Okay, so I've been proven wrong and split key is possible. However, it seems as though it would only work if the site properly implements it without some type of foul play. But how do we know the site has properly implemented it (according to your post, they haven't) or don't have some type of trick? As far as I know, that can't be guaranteed. You've shown that they can test the address for a match without its private key, but can we guarantee that they're not saving the real private key before they encrypt it?

They never have access to the private key if this is implemented correctly, which on this website is done incorrectly.

1. You generate a random 256-integer that is less than the SECP256k1 generator (G) and keep this. No one else but you should see this integer as this is essentially a private key.

G = a point on the elliptical curve. SECP256k1 using the ECDSA algorithm has a finite field with (X,Y) coordinates which indicate corresponding public and private keys.

G compressed looks like:

02 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798

which has one coordinate, X, on the curve and you are supposed to solve for Y.

G uncompressed looks:

04 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798
483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8

2. Now obviously every private key has a corresponding public key. You find this public key along the SECP256k1 curve and give it to the person generating your vanity address.

3. The person generating the vanity address now will also generate a random 256-integer less than (G) and add it to your "public key" and then hash it to see if it corresponds to the vanity address you are seeking. They keep reproducing random 256-integers until they reach the desired address.

4. Once they find the correct random integer, which is currently a public key, they will find the corresponding private key and give it to you.

5. You add the corresponding private key they gave to you to your private key and then reduce it with G.

6. Congrats. You now have the private key to your desired vanity address.

We achieve 6 because of the rule which Mesh gave,
Code (private+private)*G = (public+public)

Does this make sense?

Edit: There is also a way to create any public address you would like. I created an example just for shits and giggles.

[1KiLLJoyisSatoshiNakamotoxxxX5vsZ](#)

Orca - September 20, 2019, 4:24 pm

Hi Everyone,

Thank you for clarifying things for some of the newer users, it is much appreciated!

Also regarding our split-key implementation, please try it again and let me know if it is working fine for you now.

Best,
Nick

XMR - September 20, 2019, 8:22 pm

(Sep 20, 2019 11:22 AM)Kill Joy Wrote: [\[CITE\]](#) They never have access to the private key if this is implemented correctly, which on this website is done incorrectly.

1. You generate a random 256-integer that is less than the SECP256k1 generator (G) and keep this. No one else but you should see this integer as this is essentially a private key.

G = a point on the elliptical curve. SECP256k1 using the ECDSA algorithm has a finite field with (X,Y) coordinates which indicate corresponding public and private keys.

G compressed looks like:

02 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798

which has one coordinate, X, on the curve and you are supposed to solve for Y.

G uncompressed looks:

04 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798
483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8

2. Now obviously every private key has a corresponding public key. You find this public key along the SECP256k1 curve and give it to the person generating your vanity address.

3. The person generating the vanity address now will also generate a random 256-integer less than (G) and add it to your "public key" and then hash it to see if it corresponds to the vanity address you are seeking. They keep reproducing random 256-integers until they reach the desired address.

4. Once they find the correct random integer, which is currently a public key, they will find the corresponding private key and give it to you.

5. You add the corresponding private key they gave to you to your private key and then reduce it with G.

6. Congrats. You now have the private key to your desired vanity address.

We achieve 6 because of the rule which Mesh gave,
Code (private+private)*G = (public+public)
Does this make sense?

Edit: There is also a way to create any public address you would like. I created an example just for shits and giggles.

[1KiLLJoyisSatoshiNakamotoxxxX5vsZ](#)

It makes perfect sense. Many thanks for the detailed explanation.

So now as I understand it, if it's implemented properly, the address is guaranteed private, but otherwise, the address won't match because the end user's public key is crucial to the process? It seems like it has a sort of built in protection against trickery (eg. do it right or you get a mismatched address).

Again, thanks for the in-depth explanation. It appears as though I was indeed wrong. The more you know lol ?

Orca - September 20, 2019, 9:30 pm

Yes, that's correct the privacy and security of the address is guaranteed!

Glad you can enjoy the service now =)