Letter #1
Author: Ethan Heilman
Date: 15th of July

Dom, David and the rest of the IOTA team,

We have found serious cryptographic weaknesses in the cryptographic hash function curl used by IOTA, curl. These weaknesses threaten the security of signatures and PoW in IOTA as PoW and Signatures rely on curl to be pseudo-random and collision resistant.

We provide a report on these attacks at the end of this email. In this report we describe two attacks on the collision and second-preimage security of curl, (attack 1 and attack 2) and an attack on the pseudo-randomness of curl (attack 3). Finally we provide recommendations to mitigate these weaknesses in IOTA and suggestions on how to maintain the desired properties of IOTA.

We omit a description of our differential cryptanalysis of the transformation used in curl as these efforts are ongoing and have yet to result in full collisions. We are making steady progress (we can break 24 of 27 rounds of curl) and expect to have far more powerful collisions and preimage attack in a few weeks. However attacks 1 and 2 already generate collisions for the full 27 rounds of curl.

For replication purposes we wrote a short python script using pyota which is attached. The examples of each attack use this python script and are also replicated in the python script (curlexamples.py).

We are planning on publishing these results within the next two weeks. Let us know what actions you plan to take so that we can coordinate.

Thanks,
Ethan, Tadge

Attack 1 (Prefix-zero based collisions and second-preimages)**:**

Curl has the following properties:
1. The sponge state is initialized to all zero.
2. The curl transformation when applied to an all zero state results in an all zero state, i.e. zero is a fixed point of the transformation.
Transform([0]*729]) = [0]*729

Thus if the first N message blocks are all zeros, then the hash output and the state will be all zeros. This allows the creation of an unlimited number of collisions and second-preimages.

H([0]*243) = [0]*243 = H([0]*486) = H([0]*243*X)

[0]*243 denotes a string of zeros of length 243. H denotes a call to the curl hash function.

Since the state is the same regardless of the number of prefixed zero input blocks we can create more complex collisions and second-preimages:

H(arbitrary string) = H([0]*243||arbitrary string) = H([0]*243*X||arbitrary string)

where || denotes concatenation. Thus, given any message we can find a collision by prepending 243 zeros.

Simple all zero collisions:
msg1 =
b'999999999999999999999999999999999999999999999999999999999999999999999999999999999'
msg2 =
b'999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999'
hash1 = H(msg1)
hash2 = H(msg2)

print hash1
999999999999999999999999999999999999999999999999999999999999999999999999999999999

print hash2
999999999999999999999999999999999999999999999999999999999999999999999999999999999

print hash1 == hash2, msg1 == msg2
True False

Zero prefix collisions/second-preimages:
msg1 =
b'KXRVLFETGUTUWBCNCC9DWO99JQTEI9YXVOZHWELSYP9SG9KN9WCKXOVTEFHFH9EFZJKFYCZKQPPBXYSGJ'
msg2 =
b'999999999999999999999999999999999999999999999999999999999999999999999999999999999KXRVLFETGUTUWBCNCC9DWO99JQTEI9YXVOZHWELSYP9SG9KN9WCKXOVTEFHFH9EFZJKFYCZKQPPBXYSGJ'
hash1 = H(msg1)
hash2 = H(msg2)

print hash1
NGAZRRBUKTEJLNOQEZJZIRMY9DAVNZBOHCMELEOEFCGSJEIKDWETOMDOOQDBQILEJLD9MVXZZXZQGNHRK

print hash2
NGAZRRBUKTEJLNOQEZJZIRMY9DAVNZBOHCMELEOEFCGSJEIKDWETOMDOOQDBQILEJLD9MVXZZXZQGNHRK

print hash1 == hash2, msg1 == msg2
True False
Attack 2: Collisions resulting from failure to use message padding:

As message padding is not used in curl, we can construct messages of different lengths that result in the same sponge state. Because the sponge state is identical both messages result in the same hash output. This results in collisions/second-preimages for all messages whose trit length is not a multiple of 243.

For instance H(1) = H(10) = H(100).

This also works for messages which are longer than 243 trits. In such instances the sponge state is prepended to the second-preimage.

Short message padding collisions:
msg1 =
b'RSWWSFXPQJUBJROQBRQZWZXZJWMUBVIVMHPPTYSNW9YQIQQF9RCSJJCV
ZG9ZWITXNCSBBDHEEKDRBHV'

msg2 =
b'RSWWSFXPQJUBJROQBRQZWZXZJWMUBVIVMHPPTYSNW9YQIQQF9RCSJJCV
ZG9ZWITXNCSBBDHEEKDRBHV9'

msg3 =
b'RSWWSFXPQJUBJROQBRQZWZXZJWMUBVIVMHPPTYSNW9YQIQQF9RCSJJCV
ZG9ZWITXNCSBBDHEEKDRBHV99'
hash1 = H(msg1)
hash2 = H(msg2)
hash3 = H(msg2)

print hash1
QRXLIX9IFYTYEFVDUDLGHRJSKMXLIXMMMJAGOBABU9UAZVKMPBYOQBERC9L
TDVQGBVIMKGQVMQHJVKKQX
print hash2
QRXLIX9IFYTYEFVDUDLGHRJSKMXLIXMMMJAGOBABU9UAZVKMPBYOQBERC9L
TDVQGBVIMKGQVMQHJVKKQX

print hash3
QRXLIX9IFYTYEFVDUDLGHRJSKMXLIXMMMJAGOBABU9UAZVKMPBYOQBERC9L
TDVQGBVIMKGQVMQHJVKKQX

print hash1 == hash2 == hash3, msg1 == msg2, msg2 == msg3
True False False

Long message collisions:
In this example we choose to include IWASFLYINGINVERTED and KENNYLOGGINS in the messages to show the level of control this attack has over the colliding messages. In fact the only parts of the message not under the control of the attack is the pretended trytes, in this case “SMJFFCW'

msg1 =
b'IWASFLYINGINVERTEDNAARIPONBWXUOQUFGNOCUSSLYWKOZMZUKLNITZIF
XFWQAYVJCVMDTRSHORGNSTKX9Z9DLWNHZSMNOYTU9AUCGYBVIITEPEKIXB
COFCMQPBGXYJKSHPXNUKFTXIJVYRFILAVXEWTUICZCYYPCKENNYLOGGINS'

msg2 =
b'IWASFLYINGINVERTEDNAARIPONBWXUOQUFGNOCUSSLYWKOZMZUKLNITZIF
XFWQAYVJCVMDTRSHORGNSTKX9Z9DLWNHZSMNOYTU9AUCGYBVIITEPEKIXB
COFCMQPBGXYJKSHPXNUKFTXIJVYRFILAVXEWTUICZCYYPCKENNYLOGGINSS
MJFFCW'

hash1 = H(msg1)
hash2 = H(msg2)

print hash1
PHKPFOCZPLUCVK99GSSPZS9PIVRGLZCCZFQZFDQSSGDRNHWTBVFQPOSGSJ
PO9G9CNWSYTGRCWYNAWZMJP
print hash2
PHKPFOCZPLUCVK99GSSPZS9PIVRGLZCCZFQZFDQSSGDRNHWTBVFQPOSGSJ
PO9G9CNWSYTGRCWYNAWZMJP

print hash1 == hash2, msg1 == msg2
True False

Attack 3: Prefix-zero non-randomness

The security definitions for hash based signatures assume that the hash function is pseudo-random. However curl has non-random properties. This non-randomness includes the collisions/second-preimages properties shown above. We have also found another form of non-randomness. Namely two similar messages result in states that are different only by a positional shift.

Prefix-zero non-randomness:
I have highlighted the similarity in the output messages.
msg1 = b'9999LOGGINS'
msg2 = b'9LOGGINS'

hash1 = H(msg1)
hash2 = H(msg2)

print hash1
CUHDNR9BWKDDSGTNGFGF9WGTISDCCPKTVUZINXZWGOAPTYQZT99999Q9FS
MVXBDHHECKCQNGIKOM9IJKF

print hash2
DNR9BWKDDSGTNGFGF9WGTISDCCPKTVUZINXZWGOAPTYQZT99999Q9FSMVX
BDHHECKCQNGIKOM9IJKFEPU

Note also the substring 99999 should a occur very infrequently, but we can anecdotally observed values of the initial state bleed into the final hash message. Furthermore we have observed clear difference patterns in the hash output when each third trit is different between two messages.

Recommendations:

We make the following recommendations:

1. We recommend that IOTA immediately stop relying on curl for cryptographic purposes and instead replace curl with a vetted and well studied cryptographic hash function such as MD6, BLAKE2, Grøstl, Skein, SHA-3 or SHA-256.
2. We recommend that as soon as possible exchanges and users are made aware of these weaknesses so that they can put mitigations in place to avoid potential lose of funds.

Suggestions:

In this section we provide suggestions on ways in which IOTA can achieve similar goals as curl while minimizing the risk. These suggestions should not be seen as endorsements, prior to implementing any of these suggestions consult with a professional cryptographer or crypto-engineer.

1. If the IOTA project wishes to use a cryptographic hash function different from other cryptocurrencies for the purposes of securing PoW from established ASIC miners we suggest that they use the MD6 hash function. And as far as we are aware MD6 is not currently being used by any cryptocurrency. We will assume the use of MD6 in the second suggestion, but other secure cryptographic hash function could be used.
2. If the IOTA project wishes to ensure that trinary logic is involved in the proof of work and signature hashing process we suggest that a trinary function could be composed with a secure hash algorithm in a construction such as:

   Hash(msg) = MD6(msg || TrinaryFunction(msg)).

   This would ensure that any cryptographic weakness in Hash would imply a cryptographic weakness in MD6. Thus making this scheme at least as secure as MD6 even if the TrinaryFunction is shown to be insecure.

Letter #2
Author: Sergey Ivancheglo
Date: 15th of July

Hello.
Thank you for your interest to Curl and IOTA. I read your analysis, the phrase “we can break 24 of 27 rounds of curl” is really interesting. I’d like to ask how many transform() function invocations you need for that. It’s not clear if you meant only breaking collision-resistant part or one-wayness too, it’s pretty important to know because the security of IOTA signatures requires only the latter. I wish you have contacted me in advanced regarding the other attacks, what you identified as weaknesses is features added intentionally.
Below I’ll explain every attack/feature in detail, but before that I’d like to remind about Keccak padding rule. I didn’t verify it but I suspect that without the padding rule Attack 2 (Collisions resulting from failure to use message padding) is applicable to Keccak too. Just like Keccak, Curl is based on sponge construction, unlike Keccak, Curl has the padding rule moved outside to enable things, not possible in Keccak. One of those feature is an easy-to-find X = hash(X). This X is used as NULL transaction hash in IOTA and also as the genesis transaction, because it references itself. IOTA transactions don’t need the padding rule, they all have the fixed size of 8019 trits.
Let’s look at Attack 1 based on prepending zeros. This feature is needed for IOTA to save resources for broadcasting/storing of transactions with short payload. Because of this the payload part (named “signatureMessageFragment”) goes in the very beginning before the header part. A single transaction can hold 6561 trits of payload, if a payload is just, for example, 486 trits then we can fill the first 6075 trits with zeros and save more than 75% of bandwidth/storage without changing anything else. The transaction hash will still be the same.
Attack 2 scenario based on appending extra trits doesn’t affect IOTA negatively nor it gives any benefits. The transactions have the fixed size and data stored in the payload will use the padding rule explained below.
While I was using “padding rule” phrase, technically Curl-related method used as a countermeasure against padding attacks, probably, can’t be called “padding rule”. Before absorbing a message of a variable size we are supposed to initialize the internal part of Curl state with the size of the message. Giving a user write access to the internal state may be a not very good idea in some use-cases, so we are considering replacing this method with absorbing the size trits before absorbing the message. In either case I believe the both attacks are counteracted, aren’t they?
Now I’d like to move to Attack 3 (Prefix-zero non-randomness). You, probably, already noticed that the code of Curl is very simple. Making an algorithm as simple as possible makes analysis of the algorithm simple too (in most cases). Curl transformation function has such feature: the result of an analysis of a single output trit can be applied to all other output trits with a simple shifting. I treat it as an advantage, while most of the hashing algorithms are “protected” by obscurity of their internal mechanisms, Curl doesn’t have such drawback. “99999” in that very place of the output can be met in 1 case of 3^(5*3). The chance to see at least one occurrence of “99999” is even higher. I think that “initial state bleed into the final hash” phrase needs a better proof than few anecdotal occurrences. Personally, I running statistical tests detected no correlation between input and output trits, of course, there is always a non-zero chance that my tests had a bug. Regarding “Furthermore we have observed clear difference patterns in the hash output when each third trit is different between two messages”, could you

provide more details? It's unclear what was meant. Human brain is notorious for seeing patterns where they don't actually exist (like seeing faces in electricity sockets), a quantification of the patterns would be very helpful.

In the end of my letter I'd like to address your recommendations and suggestions.

"We recommend that IOTA immediately stop relying on curl for cryptographic purposes and instead replace curl with a vetted and well studied cryptographic hash function such as MD6, BLAKE2, Grøstl, Skein, SHA-3 or SHA-256."

Curl is based on the well-studied sponge construction. All the requirements are satisfied thus making Curl as secure as its transformation function. The function passed all standard tests (avalanche, trit-independence). Of course, vetting is a very important part and you are doing it now, so before rushing to follow the recommendation I'd like to get your answers, particularly about "24 of 27".

"We recommend that as soon as possible exchanges and users are made aware of these weaknesses so that they can put mitigations in place to avoid potential lose of funds."

We need to double-check the weaknesses before any publishing to ensure that what is being published is indeed based on sound evidences. We would love to carry on this conversation, would you guys be fine with being invited into the IOTA Foundation slack to discuss further with the rest of the team?

"If the IOTA project wishes to use a cryptographic hash function different from other cryptocurrencies for the purposes of securing PoW from established ASIC miners we suggest that they use the MD6 hash function."

IOTA project doesn't wish to use a cryptographic hash function different from other cryptocurrencies. It wishes to use a function which is fast, energy efficient and tiny codebase-wise.

"If the IOTA project wishes to ensure that trinary logic is involved in the proof of work and signature hashing process we suggest that a trinary function could be composed with a secure hash algorithm in a construction…"

This suggestion is not compatible with the wishes above, so first we are planning to assess severity of the weaknesses found by you.

Looking forward to hearing from you soon,
Sergey

Letter #3
Author: Ethan Heilman
Date: 15th of July

Hi Sergey,

I don't have much time to work on this today, but I wanted to understand more about the cryptographic assumptions that IOTA is making and respond to two of your comments.

>It's not clear if you meant only breaking collision-resistant part or one-wayness too, it's pretty important to know because the security of IOTA signatures requires only the latter.

Can you provide some documentation that IOTA signatures only require that the function is a OWF (One Way Function) and not a PRF (Pseudo Random Family). This paper [0] reduces the Winternitz one-time signature scheme to the PRF property of a function not onewayness. You can break PRness without breaking one-wayness.

>Curl is based on the well-studied sponge construction. All the requirements are satisfied thus making Curl as secure as its transformation function. The function passed all standard tests (avalanche, trit-independence). Of course, vetting is a very important part and you are doing it now, so before rushing to follow the recommendation I'd like to get your answers,

As a participant in the SHA-3 hash function contest who broke one of the 51 Round-1 SHA-3 proposals and who worked on security proofs for another SHA-3 proposal I can say with some authority that using the sponge construction and showing statistical properties of the transformation function is not sufficient to ensure security. Of the 51 Round-1 SHA-3 proposals all of them passed statistical tests and at least one round of review by NIST yet 33/51 were broken.

A more general point is that you should never roll your own crypto and if you must then it should be submitted for peer review by cryptographers before using it in a security critical application.

>I read your analysis, the phrase "we can break 24 of 27 rounds of curl" is really interesting. I'd like to ask how many transform() function invocations you need for that.

We have a differential path that propagates as a single difference until the 15th round of the transform function. Based on the per round diffusion we believe if extend this attack to result in a single difference until the 18th round, we will be able to create collisions with high probability for random states by ensuring that all differences that exist only occur with state[0:243] at the end of a transform. It should work for any number of transform invocations greater than 1.

Thanks for the fast response,

Ethan

Letter #4
Author: Sergey Ivancheglo
Date: 15th of July

Hi Ethan,
Unfortunately, we don't have the documentation you asked for, it's work-in-progress. I checked the paper you linked to and think that we can't rely on it. Unlike conventional signatures, IOTA signatures don't need to be non-repudiable and this relaxes security assumptions. I "see" that one-wayness is enough for our case but it's so obvious to me that I have no idea where to start from to show this to the others. I'm afraid I can't provide a convincing proof within a reasonable timeframe. While it's hard to prove that unicorns don't exist it's pretty easy to prove the opposite if we have one. Maybe you could show an example of a successful attack on IOTA signatures assuming that we use a one-way but non-resistant to collisions function?
I agree that even perfect statistical properties of a transformation function are not enough to accept it as secure. At this point only unsuccessful attempts to break the function can increase our assurance but we'll likely never reach 100% anyway, waiting indefinitely wasn't a good way to launch IOTA so at one point we decided to do it.
Not sure if David told you, the current version of Curl will be changed a little. Instead of X = F(A, B) we'll be using X = F(A, B, C) where X is an output trit and A, B and C are input trits. Old design doesn't provide good mixing, for example, after 10 rounds we still don't get every output trit dependent on every input trit (only 718 of 729 trits are included, if I recall the number correctly) and the 11th round is required for 729 of 729. We also need to add at least 4 (= log2(9) rounded up) extra rounds to cover all input combinations because some combinations give an easy collision. 11+4 gives the lower bound of 15 rounds after which finding a collision becomes non-trivial. I'm curious if this 15 is the same 15 from your "We have a differential path that propagates as a single difference until the 15th round of the transform function", is the 15th round included or not?

Sergey

Letter #5
Author: Dominik Schiener
Date: 17th of July

Hey guys,
Should we invite you all into our Slack where we can continue the conversation in real time?
As I have mentioned to Ethan, we should definitely discuss a financial reward for the findings and also discuss how we proceed with fixing amd disclosing this.
All the best
Dominik

Letter #6
Author: Ethan Heilman
Date: 19th of July

Dominik pinged me on twitter, in response I'm sending out a short email summarizing current state of things. Nothing has really changed because we haven't had much time to work on things.

1. Curl's collision and second preimage resistance are broken when used as a hash function,

2. IOTA isn't using curl as a hash function but instead using it as a function which maps fixed length tryte strings to fixed length tryte strings.

3. Even when used in this setting Curl is not pseudo-random. It is unclear if this breaks the signature scheme used by IOTA because the signature scheme used by IOTA is not well documented and does not have security arguments. As far as I can tell most hash-based WOTS schemes can be reduced to the PRF property of the hash function.
We can't attack the signature scheme until we understand it, and understanding a signature scheme from source code alone is a time intensive process.

4. I would strongly recommend IOTA move to a vetted and peer-reviewed hash function for their signature scheme. There are several lightweight hash functions designs available that have been studied (although not as well as SHA-3 or SHA-256): https://www.cryptolux.org/index.php/Lightweight_Hash_Functions

Ask any cryptographer, they will tell you running custom crypto primitives in production is very dangerous and violates basic industry best practices. The number one rule of cryptography is "Don't invent your own."
http://www.lauradhamilton.com/10-cryptography-mistakes-amateurs-make
https://twitter.com/petertoddbtc/status/714447656928415744

Thanks,
Ethan

Letter #7
Author: David Sønstebø
Date: 19th of July

Hey Ethan,

Thanks a lot for the effort you are putting in. There's a lot here that would be very good to discuss in a more elucidating fashion in a fora where we can go back and forth, it's hard to get to the nuances without waiting weeks while conversing through mail. I will sent an invite to the Slack now, if you join I think we can make great progress on this.

re: 'Don't roll your own' we have been very much aware of this since the beginning, naturally, but since we built it on top of Keccak proven sponge construction, we feel that it's not as much "rolling ones own" as rather optimizing preexisting proven principles.

Best,
Davd

Letter #8
Author: David Sønstebø
Date: 20th of July

Hey Ethan,

Did you receive the invite? We can also setup a chat with our ex-NSA post-Quantum hash function experts after we get the initial confusion out of the way.

Best,
David

Letter #9
Author: Neha Narula
Date: 20th of July

Hello all,

Very nice to e-meet you!

From my perspective, it would be great to continue the discussion over email -- I'm experiencing Slack fatigue.

Perhaps we could all just commit to responding faster? Feel free to cc anyone you think is important to include in the discussion.

Best,
Neha

Letter #10
Author: David Sønstebø
Date: 20th of July

Hello Neha,

long time no speak.

While I certainly understand Slack fatigue (we got the biggest slack in all of crypto and have to deal with way too many msgs daily), I want to really push on this. Responding via email will simply not be sufficient to hash out the nuances of this topic in a timely fashion, will at least take several weeks. What could be achieved in 1 hour Slack chat will take at least a month via mail.

Best,
David

Letter #11
Author: Ethan Heilman
Date: 22nd of July

IOTA team,

We can now create collisions of messages of the same size (examples follow at the end of this email). This appears to break the standard definition of security for signature schemes called EU-CMA (Existential Unforgeability under a Chosen Message Attack) [0]. Can you confirm that IOTA's signature scheme is indeed not EU-CMA secure?

We note these collisions result from the exceptionally poor performance of curl's transform function when attacked by textbook differential cryptanalysis attack. We hope this demonstrates that merely employing a sponge construction is not a sufficient condition for security and that IOTA does indeed meet the definition of rolling their own crypto.

- We recommend in the strongest possible terms that you stop deploying in-house developed cryptography in production and immediately switch to standardized and peer reviewed libraries and designs.
- We recommend that any cryptography and signature algorithms used in IOTA be well documented and submitted for public review prior to their introduction into production software given the potential risk of a loss of funds to the public. This documentation should include any non-standard security assumptions and detailed formal arguments of security.

As discussed earlier we are planning on publishing these weaknesses within a week or so. Please inform us what immediate mitigation steps you taking so that we can coordinate the public release of these weaknesses.

Thanks,
Ethan

Attack on EU-CMA security of IOTA's signature scheme

We base this on the following description of the signature scheme used in IOTA.

hashedmsg = Curl(msg)
sig = Sign(SecretKey, hashedmsg)

We present the following attack on the EU-CMA security of the signature scheme used in IOTA.

1. Eve chooses two messages, msg1, msg2 such that:
curl(msg1) = curl(msg2) and msg1 != msg2

2. Eve sends msg1 to Alice and asks Alice to sign it.

3. Alice sends Eve a signature on msg1:
sig1 = signature(SK, curl(msg1))

4. Eve produces a valid signature,message pair (sig1,msg2) where msg1 is a message which Alice's has not signed thus breaking CMA-unforgability.

signature(SK, curl(msg1)) = signature(SK, curl(msg2))

Example collision:

We have verified these collisions using the ccurl-digest tool. We used the lyrics to the 80's hit single "push it to the limit" in the colliding messages to demonstrate that we fully collide the internal state of curl and thus have arbitrary control over most of the message. We note that this internal state collision enables us to instantly create millions of other colliding messages.

$ ./ccurl-digest
RETHT9ES9HRCUITBHVCUHOBPUUUHT9PHLUNWRWGKBKF9YUMDWRXTRVGZ
HFZEHGATZXZAUPGVEKNMQXFVRXHF9QJQHUTILIPIXUYRVSJEIOJDRIUVWMUA
BSIKIBAKENE9KVFJUEQUHFRVGELFGJIDXQARWH99XTORHXRETHT9ES9HRCUI
TBHVCUHOBPUUUHT9PHLUNWRWGKBKF9YUMDWRXTRVGZHFZEHGATZXZAUP
GVEKNMQXFVRPUSHITTOTHELIMIT9WALKALONGTHERAZORSEDGE9BUTDONT
LOOKDOWNJUSTKEEPYOURHEAD9ORYOULLBEFINISHED99OPENUPTHELIMIT9
PASTTHEPOINTOFNORETURN9YOUVEREACHEDTHETOPBUTSTILLYOUGOTTAL
EARN9HOWTOKEEPIT99HITTHEWHEELANDDOUBLETHESTAKES9THROTTLEWID
EOPENLIKEABATOUTOFHELL9YOUCRASHTHEGATESCRASHTHEGATES9GOING
FORTHEBACKOFBEYOND9NOTHINGGONNASTOPYOUTHERESNOTHINGTHATST
RONG9SOCLOSENOWYOURENEARLYATTHEBRINK9SOPUSHITOOHYEAH99WEL
COMETOTHELIMITLIMIT9TAKEITBABYONESTEPMORE9THEPOWERGAMESSTILL
PLAYINGSO9YOUBETTERWINIT99PUSHITTOTHELIMITLIMIT9NOONELEFTTOSTA
NDINYOURWAY9YOUMIGHTGETCARELESSBUTYOULLNEVERBESAFE9WHILEYO
URESTILLINIT99WELCOMETOTHELIMITLIMIT9STANDINGONTHERAZORSEDGE9D
ONTLOOKDOWNJUSTKEEPYOURHEAD9ORYOULLBEFINISHED99WELCOMETOT
HELIMIT9PUSHITTOTHELIMITPUSHITTOTHELIMIT9WALKALONGTHERAZORSEDG
E9BUTDONTLOOKDOWNJUSTKEEPYOURHEAD9ORYOULLBEFINISHED99OPENU
PTHELIMIT9PASTTHEPOINTOFNORETURN9YOUVEREACHEDTHETOPBUTSTILLY
OUGOTTALEARN9HOWTOKEEPIT99HITTHEWHEELANDDOUBLETHESTAKES9TH
ROTTLEWIDEOPENLIKEABATOUTOFHELL9YOUCRASHTHEGATESCRASHTHEGA
TES9GOINGFORTHEBACKOFBEYOND9NOTHINGGONNASTOPYOUTHERESNOTH
INGTHATSTRONG9SOCLOSENOWYOURENEARLYATTHEBRINK9SOPUSHITOOH
YEAH99WELCOMETOTHELIMITLIMIT9TAKEITBABYONESTEPMORE9THEPOWER
GAMESSTILLPLAYINGSO9YOUBETTERWINIT99PUSHITTOTHELIMITLIMIT9NOON

ELEFTTOSTANDINYOURWAY9YOUMIGHTGETCARELESSBUTYOULLNEVERBESA
FE9WHILEYOURESTILLINIT99WELCOMETOTHELIMITLIMIT9STANDINGONTHERA
ZORSEDGE9DONTLOOKDOWNJUSTKEEPYOURHEAD9ORYOULLBEFINISHED99
WELCOMETOTHELIMIT9PUSHITTOTHELIMITPUSHITTOTHELIMIT9WALKALONGT
HERAZORSEDGE9BUTDONTLOOKDOWNJUSTKEEPYOURHEAD9ORYOULLBEFIN
ISHED99OPENUPTHELIMIT9PASTTHEPOINTOFNORETURN9YOUVEREACHEDTH
ETOPBUTSTILLYOUGOTTALEARN9HOWTOKEEPIT99HITTHEWHEELANDDOUBLE
THESTAKES9THROTTLEWIDEOPENLIKEABATOUTOFHELL9YOUCRASHTHEGATE
SCRASHTHEGATES9GOINGFORTHEBACKOFBEYOND9NOTHINGGONNASTOPYO
UTHERESNOTHINGTHATSTRONG9SOCLOSENOWYOURENEARLYATTHEBRINK9
SOPUSHITOOHYEAH99WELCOMETOTHELIMITLIMIT9TAKEITBABYONESTEPMOR
E9THEPOWERGAMESSTILLPLAYINGSO9YOUBETTERWINIT99PUSHITTOTHELIMI
TLIMIT9NOONELEFTTOSTANDINYOURWAY9YOUMIGHTGETCARELESSBUTYOUL
LNEVERBESAFE9WHILEYOURESTILLINIT99WELCOMETOTHELIMITLIMIT9STANDI
NGONTHERAZORSEDGE9DONTLOOKDOWNJUSTKEEPYOURHEAD9ORYOULLBE
FINISHED99WELCOMETOTHELIMIT9PUSHITTOTHELIMITPUSHITTOTHELIMIT9WA
LKALONGTHERAZORSEDGE9BUTDONTLOOKDOWNJUSTKEEPYOURHEAD9ORY
OULLBEFINISHED99OPENUPTHELIMIT9PASTTHEPOINTOFNORETURN9YOUVER
EACHEDTHETOPBUTSTILLYOUGOTTALEARN9HOWTOKEEPIT99HITTHEWHEELA
NDDOUBLETHESTAKES9THROTTLEWIDEOPENLIKEABATOUTOFHELL9YOUCRA
SHTHEGATESCRASHTHEGATES9

BUWSNIAGAYCVUPGUJMWIIYHR9DQAPVHRKHXPJKB9BTCKPUXTFSNPXIBHCIIY
CJRAQJOGHXEYLXSJURUFS

$ ./ccurl-digest
RETHT9ES9HRCUITBHVCUHOBPUUUHT9PHLUNWRWGKBKF9YUMDWRXTRVGZ
HFZEHGATZXZAUPGVEKNMQXFVRXHF9QJQHUTILIPIXUYRVSJEIPJDRIUVWMUA
BSIKIBAKENE9KVFJUEQUHFRVGELFGJIDXQARWH99XTORHXRETHT9ES9HRCUI
TBHVCUHOBPUUUHT9PHLUNWRWGKBKF9YUMDWRXTRVGZHFZEHGATZXZAUP
GVEKNMQXFVRPUSHITTOTHELIMIT9WALKALONGTHERAZORSEDGE9BUTDONT
LOOKDOWNJUSTKEEPYOURHEAD9ORYOULLBEFINISHED99OPENUPTHELIMIT9
PASTTHEPOINTOFNORETURN9YOUVEREACHEDTHETOPBUTSTILLYOUGOTTAL
EARN9HOWTOKEEPIT99HITTHEWHEELANDDOUBLETHESTAKES9THROTTLEWID
EOPENLIKEABATOUTOFHELL9YOUCRASHTHEGATESCRASHTHEGATES9GOING
FORTHEBACKOFBEYOND9NOTHINGGONNASTOPYOUTHERESNOTHINGTHATST
RONG9SOCLOSENOWYOURENEARLYATTHEBRINK9SOPUSHITOOHYEAH99WEL
COMETOTHELIMITLIMIT9TAKEITBABYONESTEPMORE9THEPOWERGAMESSTILL
PLAYINGSO9YOUBETTERWINIT99PUSHITTOTHELIMITLIMIT9NOONELEFTTOSTA
NDINYOURWAY9YOUMIGHTGETCARELESSBUTYOULLNEVERBESAFE9WHILEYO
URESTILLINIT99WELCOMETOTHELIMITLIMIT9STANDINGONTHERAZORSEDGE9D
ONTLOOKDOWNJUSTKEEPYOURHEAD9ORYOULLBEFINISHED99WELCOMETOT
HELIMIT9PUSHITTOTHELIMITPUSHITTOTHELIMIT9WALKALONGTHERAZORSEDG
E9BUTDONTLOOKDOWNJUSTKEEPYOURHEAD9ORYOULLBEFINISHED99OPENU
PTHELIMIT9PASTTHEPOINTOFNORETURN9YOUVEREACHEDTHETOPBUTSTILLY
OUGOTTALEARN9HOWTOKEEPIT99HITTHEWHEELANDDOUBLETHESTAKES9TH
ROTTLEWIDEOPENLIKEABATOUTOFHELL9YOUCRASHTHEGATESCRASHTHEGA
TES9GOINGFORTHEBACKOFBEYOND9NOTHINGGONNASTOPYOUTHERESNOTH
INGTHATSTRONG9SOCLOSENOWYOURENEARLYATTHEBRINK9SOPUSHITOOH
YEAH99WELCOMETOTHELIMITLIMIT9TAKEITBABYONESTEPMORE9THEPOWER

GAMESSTILLPLAYINGSO9YOUBETTERWINIT99PUSHITTOTHELIMITLIMIT9NOON
ELEFTTOSTANDINYOURWAY9YOUMIGHTGETCARELESSBUTYOULLNEVERBESA
FE9WHILEYOURESTILLINIT99WELCOMETOTHELIMITLIMIT9STANDINGONTHERA
ZORSEDGE9DONTLOOKDOWNJUSTKEEPYOURHEAD9ORYOULLBEFINISHED99
WELCOMETOTHELIMIT9PUSHITTOTHELIMITPUSHITTOTHELIMIT9WALKALONGT
HERAZORSEDGE9BUTDONTLOOKDOWNJUSTKEEPYOURHEAD9ORYOULLBEFIN
ISHED99OPENUPTHELIMIT9PASTTHEPOINTOFNORETURN9YOUVEREACHEDTH
ETOPBUTSTILLYOUGOTTALEARN9HOWTOKEEPIT99HITTHEWHEELANDDOUBLE
THESTAKES9THROTTLEWIDEOPENLIKEABATOUTOFHELL9YOUCRASHTHEGATE
SCRASHTHEGATES9GOINGFORTHEBACKOFBEYOND9NOTHINGGONNASTOPYO
UTHERESNOTHINGTHATSTRONG9SOCLOSENOWYOURENEARLYATTHEBRINK9
SOPUSHITOOHYEAH99WELCOMETOTHELIMITLIMIT9TAKEITBABYONESTEPMOR
E9THEPOWERGAMESSTILLPLAYINGSO9YOUBETTERWINIT99PUSHITTOTHELIMI
TLIMIT9NOONELEFTTOSTANDINYOURWAY9YOUMIGHTGETCARELESSBUTYOUL
LNEVERBESAFE9WHILEYOURESTILLINIT99WELCOMETOTHELIMITLIMIT9STANDI
NGONTHERAZORSEDGE9DONTLOOKDOWNJUSTKEEPYOURHEAD9ORYOULLBE
FINISHED99WELCOMETOTHELIMIT9PUSHITTOTHELIMITPUSHITTOTHELIMIT9WA
LKALONGTHERAZORSEDGE9BUTDONTLOOKDOWNJUSTKEEPYOURHEAD9ORY
OULLBEFINISHED99OPENUPTHELIMIT9PASTTHEPOINTOFNORETURN9YOUVER
EACHEDTHETOPBUTSTILLYOUGOTTALEARN9HOWTOKEEPIT99HITTHEWHEELA
NDDOUBLETHESTAKES9THROTTLEWIDEOPENLIKEABATOUTOFHELL9YOUCRA
SHTHEGATESCRASHTHEGATES9

BUWSNIAGAYCVUPGUJMWIIYHR9DQAPVHRKHXPJKB9BTCKPUXTFSNPXIBHCIIY
CJRAQJOGHXEYLXSJURUFS
Shorter colliding messages are also possible for instance:

msg1 =
"RETHT9ES9HRCUITBHVCUHOBPUUUHT9PHLUNWRWGKBKF9YUMDWRXTRVGZ
HFZEHGATZXZAUPGVEKNMQXFVRXHF9QJQHUTILIPIXUYRVSJEIOJDRIUVWMUA
BSIKIBAKENE9KVFJUEQUHFRVGELFGJIDXQARWH99XTORHXRETHT9ES9HRCUI
TBHVCUHOBPUUUHT9PHLUNWRWGKBKF9YUMDWRXTRVGZHFZEHGATZXZAUP
GVEKNMQXFVR"
msg2 =
"RETHT9ES9HRCUITBHVCUHOBPUUUHT9PHLUNWRWGKBKF9YUMDWRXTRVGZ
HFZEHGATZXZAUPGVEKNMQXFVRXHF9QJQHUTILIPIXUYRVSJEIPJDRIUVWMUA
BSIKIBAKENE9KVFJUEQUHFRVGELFGJIDXQARWH99XTORHXRETHT9ES9HRCUI
TBHVCUHOBPUUUHT9PHLUNWRWGKBKF9YUMDWRXTRVGZHFZEHGATZXZAUP
GVEKNMQXFVR"

Results in a colliding hash.

Letter #12
Author: David Sønstebø
Date: 23rd of July

Hello Ethan,

Thanks for the continued effort. We will have a thorough response tomorrow and then we can hopefully come to consensus on the next steps.

Re:

"*As discussed earlier we are planning on publishing these weaknesses within a week or so. Please inform us what immediate mitigation steps you taking so that we can coordinate the public release of these weaknesses.*"

This is also something we obviously need to discuss in some depth as well, I presume that you also see it as axiomatic and agree that rush-publishing this would be very detrimental and irresponsible. Presuming your findings are correct (tomorrow's response will elucidate this further) a change to the protocol on this level will take several weeks to resolve on a technical level, not to mention logistics. One step at a time: let's first see eye to eye on the actual issue at hand. I am confident that you guys are in agreement with this.

Thanks,
All the best,
David

Letter #13
Author: David Sønstebø
Date: 23rd of July

For some reason Mail.ru has some issues, so I have to send the mail along for Sergey:

"Hi,
Yesterday David promised to provide a thorough response. Unfortunately, I still haven't got an answer to my question from the very first letter, without that the leap of faith is required to follow your recommendations. The question was:
How many transform() function invocations do you need to "break 24 of 27 rounds of curl"?
(You have broken 27 rounds too, the corresponding number of transformations for 27 rounds together with 24 rounds would let me assess if your analysis is more efficient than the analysis utilizing properties of function F(A, B).)
PS: By the way, please, use "Curl-P-27" name instead of "curl" to conform to the naming convention where "P" stands for "prototype" (i.e. Curl version with 2-argument truth table function instead of the 3-argument one) and "27" shows the number of rounds in transform() function.

Sergey"

Letter #14
Author: Ethan Heilman
Date: 23rd of July

Hi Sergey,

>How many transform() function invocations do you need to “break 24 of 27 rounds of curl”?
I believe I answered that question in the second email when I wrote:
">I read your analysis, the phrase “we can break 24 of 27 rounds of curl” is really interesting. I’d like to ask how many transform() function invocations you need for that.

We have a differential path that propagates as a single difference until the 15th round of the transform function. Based on the per round diffusion we believe if extend this attack to result in a single difference until the 18th round, we will be able to create collisions with high probability for random states by ensuring that all differences that exist only occur with state[0:243] at the end of a transform. It should work for any number of transform invocations greater than 1."
Perhaps I misunderstood your question. Were you asking how many queries to the transform function were necessary to prevent diffusion to 15-th round? If so, here is my answer:

The attack I described involves finding a series of equations which if satisfied prevent the diffusion of a single trit for 14 rounds (diffusion then begins in the 15th round). Our attack relies on finding two message blocks, MB1 and MB2 which meet the equations. MB1 is determined by making random queries to the transform until the equation is satisfied, around 30-40 queries to transform. The second MB2 is just determined by the equations, i.e., 0 queries to the transform.

As predicted we extended this attack to the 20-th round (thereby enabling a 27th round collision). However, we noticed that things were worse than we originally thought and we didn't actually need to solve the equations out to 14 rounds. Instead we only use these equations for the first 8 rounds and then just rely on the poor differential properties of F(A,B) to brute force the next 11-12 rounds. Not sure exactly how many transform queries we are calling, I just let my slow python code run over night on my laptop.

> By the way, please, use “Curl-P-27” name instead of “curl” to conform to the naming convention where “P” stands for “prototype” (i.e. Curl version with 2-argument truth table function instead of the 3-argument one) and “27” shows the number of rounds in transform() function.
I am shocked that you would call a hash function deployed in production, with "a 800 million dollar bug bounty" as Dominik put it, a prototype. Are you telling me the IOTA project deployed "prototype crypto" in production?
Is this 3-argument version of curl documented anywhere?

I am becoming increasingly concerned that you are going to just patch curl and not switch to a well vetted standardized secure hash function. Can you confirm that you are not going to do this?

Thanks,
Ethan

Letter #15
Author: Sergey Ivancheglo
Date: 24th of July

Hi Ethan,
Thanks for the reply. I wasn't asking how many queries to the transform function had been necessary to prevent diffusion to 15-th round. My question was aimed to assess complexity of your attack, but now I see that I misunderstood you. I was thinking that you had found an attack allowing to generate an arbitrary collision ("selective forgery" in the classical terminology), after reading your extended explanation I see that it's not the case.
> We note these collisions result from the exceptionally poor performance of curl's transform function…
> … just rely on the poor differential properties of F(A,B) to brute force the next 11-12 rounds
If you look at Keccak round function you will see that it diffuses inputs with a higher strength than Curl round function, a single Keccak round equals to several Curl rounds. Fine-graininess of Curl gives us what Keccak can't give – fine-grain control over performance and energy consumption. We contacted you ~3 months ago regarding reviewing Curl, I wish you would have replied back then before starting your analysis, this would save a lot of time for you… In order to ensure no more time gets wasted on either side, I want to give a basic overview of Curl, which I believe will clear up some of the misunderstanding that seem to be present in our dialog.
Curl is a hash function based on the sponge construction. Its round function is very simple to allow easy algebraic analysis of Curl structure which is very important because we need to get lower bounds on the number of the rounds in different use-cases. In other words, we could use 27 rounds for one-wayness, 6 rounds for string hashing, 45 rounds for collision resistance, 21 rounds for key generation, 39 rounds for pseudo-random number generation, you've got the idea. This feature is so important that it's reflected in the name of the function – Curl. Each round is a single curl, more curls – more beautiful (more secure) our haircut (our hash function) is. One size doesn't fit all - http://keccak.noekeon.org/is_sha3_slow.html shows how Keccak solves the problem of universalism (spoiler: it solves it in the same way, just abandons universalism).

…Now, after the short introduction, I'm moving to the other issues.
> 1. Curl's collision and second preimage resistance are broken when used as a hash function,
So far you have broken them for Curl-P-1, Curl-P-2, …, Curl-P-26, Curl-P-27 functions, I'd like to know your assessment on the min number of rounds which makes your attack infeasible. This may be useful if we decide to unlock "repudiation" before Curl-729-R (with the 3-argument truth table) is ready. By the way, we don't run a competition similar to http://keccak.noekeon.org/crunchy_contest.html but we will include your findings if we do.
> 2. IOTA isn't using curl as a hash function but instead using it as a function which maps fixed length tryte strings to fixed length tryte strings.
Not entirely sure what you mean, maybe you omitted "cryptographic" in front of "hash function"? In this case you are right, second-preimage resistance is an anti-feature, collision resistance threat is nullified by Coordinator while allows us to easily attack scam-driven copycats. One of such copycat even conducted a sophisticated scam using

99% of IOTA codebase, they got published on https://cointelegraph.com/news/crypto-world-has-been-turning-into-ponzi-scheme-opinion, simulate activity on https://twitter.com/Aidos_kuneen and were even listed on CoinMarketCap.com according to https://mobile.twitter.com/Aidos_kuneen/status/876042685563514880/photo/1. As you are well aware, this space is riddled in scammers, and sometimes scammers can hurt the reputation of honest projects, so having such a simple fix to mitigate them is a bonus.

> 3. Even when used in this setting Curl is not pseudo-random.

Could you tell what randomness tests are not passed by Curl-P-27?

> …We present the following attack on the EU-CMA security of the signature scheme used in IOTA…

Your attack is based on a wrong assumption about IOTA signing scheme. As you know, some signing schemes counteract your scenario by higher level protocols (e.g. RSA), IOTA follows the same route. Let's apply your findings on the actual scheme, do you have time for that? If yes, then IOTA team will prioritize the documentation.

> We can now create collisions of messages of the same size (examples follow at the end of this email). This appears to break the standard definition of security for signature schemes called EU-CMA (Existential Unforgeability under a Chosen Message Attack) [0]. Can you confirm that IOTA's signature scheme is indeed not EU-CMA secure?

I claim that IOTA's signature scheme is EU-CMA secure, to prove that I'll need some time, but your reply to this email will clarify some issues and make it easier to prove this to you.

> We recommend in the strongest possible terms that you stop deploying in-house developed cryptography in production and immediately switch to standardized and peer reviewed libraries and designs.

Your recommendation makes perfect sense, but before that we need to get a 3rd-party confirmation, this will take some time. I also want to highlight here that we are working with multiple security researchers, cryptographers, mathematicians, students and universities with supercomputers (Imperial College London and St. Petersburg Polytechnic University) to verify our assumptions both in IOTA consensus as well as Curl cryptography. This is not something we take lightly, we are very aware of the security risks involved and therefore approach it in a measured fashion while also pushing for progress.

> I am shocked that you would call a hash function deployed in production, with "a 800 million dollar bug bounty" as Dominik put it, a prototype.

I see Greek wasn't your favorite subject in school :), don't worry, word "prototype" is similar to https://en.wikipedia.org/wiki/Prototype_pattern, not to what you thought about. It is also important to keep in mind that all distributed ledgers are currently in a "prototype phase".

> Is this 3-argument version of curl documented anywhere?

No, it's still being tested and these tests by nature will take several months.

> I am becoming increasingly concerned that you are going to just patch curl and not switch to a well vetted standardized secure hash function. Can you confirm that you are not going to do this?

We are definitely not going to "just patch curl", we are going to get 3rd-party audit of your claims and then follow your recommendations (if the auditors greenlight them). I want to reassure you that we very much appreciate your input and that we take it seriously, this is why we reached out to you in the first place 3 months ago as well. We are already checking this now with two 3rd parties, including our own inhouse cryptographers and mathematicians. We appreciate your understanding that this will take some time, but your response to this mail will be very helpful in gauging the exact

issue and provide ETA, then we can coordinate on the next steps (implementation change, publication etc.)

PS: I'm CCing Paul and Alon, they are devs who will be working on the changes.
Thanks,
Sergey

Letter #16
Author: Ethan Heilman
Date: 25th of July

IOTA team,

1. It has been 11 days since we first contacted you. Please send us a concrete timeline detailing when you are planning replace curl with standard and peer-reviewed cryptography?

>universities with supercomputers (Imperial College London and St. Petersburg Polytechnic University) to verify our assumptions both in IOTA consensus as well as Curl cryptography.

2. Please CC us on any communications with other cryptographers as this our unpublished research you are disseminating. If you would prefer we could use our relationships with cryptographers at MIT, BU and elsewhere. As mentioned earlier we did consult informally with additional subject matter experts.

>Your attack is based on a wrong assumption about IOTA signing scheme.

3. Can you explain your signature scheme then? When generating a signature do you or do you not hash the message? We have looked in multiple implementations and it appears that you do. If you do then our attack holds and your signing scheme EU-CMA forgery depends on the collision resistance of curl.

4. We have heard vague rumors of another team with attacks against curl. Hopefully they are also doing a responsible disclosure progress with you. If such attacks are made public it does not make sense for us to withhold publication.

>I'd like to know your assessment on the min number of rounds which makes your attack infeasible. This may be useful if we decide to unlock "repudiation" before Curl-729-R (with the 3-argument truth table) is ready. By the way, we don't run a competition similar to http://keccak.noekeon.org/crunchy_contest.html but we will include your findings if we do.

5. This sounds very much like you are going to increase the number of rounds in curl and update the transformation function. I would strongly recommend against continuing to roll your own crypto.

These attacks we have developed will always get better. The current results are from roughly 20 hours of work. Even teams of experts in symmetric cryptography make mistakes which is we have decade long competitions to choose new symmetric primitives.

Thanks,
Ethan

Letter #17
Author: Sergey Ivancheglo
Date: 25th of July

Hi, Ethan

> 1. It has been 11 days since we first contacted you. Please send us a concrete timeline detailing when you are planning replace curl with standard and peer-reviewed cryptography?
Such a procedure is not done overnight obviously. We have been very responsive on this matter and have internally prioritized it over any other project we have currently on-going. Apart from validating the claims put forth, we are currently assessing how to best integrate a peer-reviewed hash function (4 of our developers are working on this). Here is a proposed timeline:
5th of August – We change the code to replace Curl with Keccak. This means that the Core client, libraries and wallets will be updated by then.
5th of August – 10th of August – Users move their tokens to addresses generated with Keccak
12th of August – the day when we disclose the details
As you know, the worst thing to do at this stage is to release a rushed fix. Which is why we are taking extra effort in ensuring that these drastic changes to the software are done securely, while also taking the time into account (we want this done as soon as possible as well). I hope you understand this and agree with the proposed timeline. We will further communicate any changes, updates and proposals to you for review.

> 2. Please CC us on any communications with other cryptographers as this our unpublished research you are disseminating. If you would prefer we could use our relationships with cryptographers at MIT, BU and elsewhere. As mentioned earlier we did consult informally with additional subject matter experts.
Will CC for sure. Thank you for your offer, for now we are focusing on having the transition to Keccak done in time. Indeed please disclose who you have talked to about this.
> 3. Can you explain your signature scheme then? When generating a signature do you or do you not hash the message? We have looked in multiple implementations and it appears that you do. If you do then our attack holds and your signing scheme EU-CMA forgery depends on the collision resistance of curl.
Extra checks are done on higher level, we'll provide documentation after the transition.
> 4. We have heard vague rumors of another team with attacks against curl. Hopefully they are also doing a responsible disclosure progress with you. If such attacks are made public it does not make sense for us to withhold publication.
There was a site on the Internet with "IOTA seed cracker". We tested the code provided on that site but it didn't help to crack a seed, we were unable to reach the author because of absence of his name and contact info. But apart from yourself, we have not heard about any other attacks on Curl. Maybe you can check with the source of the rumor if this is true and we can coordinate this with them as well?
> 5. This sounds very much like you are going to increase the number of rounds in curl and update the transformation function. I would strongly recommend against continuing to roll your own crypto.
We are not going to increase the number of rounds, we are already replacing Curl with Keccak. We will however continue the development of Curl and are putting up more

considerable resources to do so. This will obviously only make it into the protocol after it has been vetted and peer reviewed.
PS: Does the provided timeline fit into your plans?

Sergey

Letter #18
Author: Neha Narula
Date: 25th of July

Hi Sergey,

I would very much appreciate a pointer to where this is done in the code. Hopefully that's something you can accommodate relatively quickly, without waiting for the transition.

Thank you,
Neha

Letter #19
Author: Sergey Ivancheglo
Date: 25th of July

Hi, Neha

https://github.com/iotaledger/iri/blob/dev/src/main/java/com/iota/iri/TransactionValidator.java#L72
Here the collision would invalidate a transaction if it adjusted “timestamp” to an invalid value.
https://github.com/iotaledger/iri/blob/dev/src/main/java/com/iota/iri/TransactionValidator.java#L77
Here the collision would invalidate a transaction if it adjusted “value” to an invalid value.
https://github.com/iotaledger/iri/blob/dev/src/main/java/com/iota/iri/BundleValidator.java#L35
Here the collision would invalidate a bundle if it broke “currentIndex” sequence or “lastIndex” didn’t match the real number of the transactions in the bundle.
https://github.com/iotaledger/iri/blob/dev/src/main/java/com/iota/iri/BundleValidator.java#L43
Here the collision would invalidate the bundle if it changed the transferred value.
https://github.com/iotaledger/iri/blob/dev/src/main/java/com/iota/iri/BundleValidator.java#L73
Here the collision would invalidate the bundle if it changed “address” field of a multi-sig address (there are no single-sig addresses in official wallet, all addresses are 2-of-2 ones, but custom software might generate single-sig addresses).
https://github.com/iotaledger/iri/blob/dev/src/main/java/com/iota/iri/BundleValidator.java#L79
Here the collision would invalidate the bundle if it changed the spending address.
I don’t have the source code of the Coordinator on this notebook, the code is not public, but I can get it tomorrow if it’s important. The Coordinator is used as an extra protection measure. Particularly, it stores all transactions that reach it, this allows us to recover iotas sent to addresses with typo (we detect it because users usually publish the addresses on the tangle before usage), during the latest snapshot we used the Coordinator to restore iotas sent to addresses generated with the previous (now obsolete) address generation scheme.
The above is the higher level protocol that I mentioned in previous letters. We are migrating to Keccak anyway, we’ll provide the proof of infeasibility of your attack after the transition.

Best regards,
Sergey

Letter #20
Author: Dominik Schiener
Date: 27th of July

Hey Neha and Ethan,

we're currently working on the implementation and the new unit tests. Could we get some feedback with regards to the timeline? We are putting up more effort for testing, especially taking the user transition into account, but we think the proposed timeline makes a lot of sense and should be achievable.

Best
Dominik

Letter #21
Author: Sergey Ivancheglo
Date: 27th of July

Hi Ethan,
I have already prepared the Coordinator for the transition (the others are still working on back- and front-end parts though). I removed unnecessary security checks and now IOTA can do an even higher CTPS, the unfortunate trade-off is, of course, that now scam copycats can do frivolous ICOs without us being able to prevent it anymore.

Now I'm working on a paper addressing your claims and I have stumbled into something I can't get. English is my third language, I would appreciate if you explained the following:

On the 22nd of July you wrote:
> We can now create collisions of messages of the same size (examples follow at the end of this email). This appears to break the standard definition of security for signature schemes called EU-CMA (Existential Unforgeability under a Chosen Message Attack) [http://www.cs.tau.ac.il/~canetti/f08-materials/scribe8.pdf].

On the 19th of July you wrote:
> As far as I can tell most hash-based WOTS schemes can be reduced to the PRF property of the hash function.

And on the 15th of July you wrote:
> Can you provide some documentation that IOTA signatures only require that the function is a OWF (One Way Function) and not a PRF (Pseudo Random Family). This paper [https://eprint.iacr.org/2011/191.pdf] reduces the Winternitz one-time signature scheme to the PRF property of a function not onewayness. You can break PRness without breaking one-wayness.

The first referenced paper states:
> if OWFs exist then EU-CMA signature schemes exist

And you state:
> You can break PRness without breaking one-wayness.

When I combine all this in my head I get the following conclusion: It's possible to have an EU-CMA signature scheme using a one-way function which is not pseudo-random. Is it a correct claim?

PS: Should we review each other's paper before the publication?
Sergey

Letter #22
Author: Neha Narula
Date: 28th of July

Hi,

Thanks for the updated timeline. Assuming that no one else publishes an attack we are willing to adhere to your timeline and delay publication until August 12. We are glad to hear you are replacing curl. I think we're in agreement, but I'd like to clarify a few points:

1. On August 5th, you will need to inform users to upgrade and change addresses. How do you plan on messaging this? At this point people will start looking for vulnerabilities.

2. Related to (1), could you please send us documentation of your signature scheme? We are a bit concerned about that as well, and once you deploy this fix I'm sure it will start getting a lot more attention.

We might be a bit slow in responding -- Ethan and Tadge are in a different timezone for a few days.

Thanks,
Neha

Letter #23
Author: Sergey Ivancheglo
Date: 28th of July

Hi,

> 1. On August 5th, you will need to inform users to upgrade and change addresses. How do you plan on messaging this? At this point people will start looking for vulnerabilities.

We are still preparing everything internally. We will give you a headsup on the way this is going to be communicated.

> 2. Related to (1), could you please send us documentation of your signature scheme? We are a bit concerned about that as well, and once you deploy this fix I'm sure it will start getting a lot more attention.

Address generation:

1. A user supplies a seed (https://github.com/iotaledger/iri/blob/dev/src/main/java/com/iota/iri/hash/ISS.java#L18)
2. A subseed is generated as Curl(seed+N) where N is the index of an address to generate (https://github.com/iotaledger/iri/blob/dev/src/main/java/com/iota/iri/hash/ISS.java#L42)
3. The subseed is absorbed into Curl and 54 243-trit chunks are squeezed out to get the private key (https://github.com/iotaledger/iri/blob/dev/src/main/java/com/iota/iri/hash/ISS.java#L46)
4. Each chunk is modified with Chunk=Curl(Chunk) formula 26 times to get the public key fragment (https://github.com/iotaledger/iri/blob/dev/src/main/java/com/iota/iri/hash/ISS.java#L72)
5. Curl(First27Chunks) is computed to get Digest0, Curl(Remaining27Chunks) is computed to get Digest1 (https://github.com/iotaledger/iri/blob/dev/src/main/java/com/iota/iri/hash/ISS.java#L85)
6. Curl(Digest0 || Digest1) is computed to get the address (https://github.com/iotaledger/iri/blob/dev/src/main/java/com/iota/iri/hash/ISS.java#L101)

Bundle hash generation:

A typical bundle consists of 4 transactions going in the following order:
1. Transaction depositing AAA tokens to Alice
2. Transaction spending BBB tokens from Bob
3. Transaction spending 0 tokens from Bob (required for 2-of-2 multisig)
4. Transaction depositing (BBB-AAA) tokens to Bob

Each transaction has the essence which contains:

1. 243 trits of the address
2. 33 trits of the transacted value
3. 48 trits set to 0 (padding, a single non-zero trit would invalidate the transaction)
4. 81 trits of the tag
5. 27 trits of the timestamp

6. 27 trits of the transaction index in the bundle
7. 27 trits of the last transaction (in the bundle) index

The bundle hash is computed by absorbing all the essences in the correct order and squeezing out 243 trits.

Here is an example of such a bundle:

http://iota.tips/search/?kind=transaction&hash=SOKGFAZNVJMEDT9LRODFGSTHDLOIJWSFHYHECIE9YQ9NOEAJGFFDOLKCRVGXLXBU9SMIHKEGDCRZ99999
http://iota.tips/search/?kind=transaction&hash=VETNZBBSTRUNVYVPRRZHCRTLQVG9JOOHQIPYHWWAZXZKILIOQXLLKHOIHZCSMGEPA9DG9ZPJYJLN99999
http://iota.tips/search/?kind=transaction&hash=MFIRBWASZPEZFOKRERDCCVILWIXUHSCZFUJ9NKIXFUQLQWMANHLFNOUOJIIORWMYDHIOOCHVGEUZ99999
http://iota.tips/search/?kind=transaction&hash=DSHH9TNNPJPUYZVRIBLIUTBIWSOFKVCAGYSKYWGA9LVCFULRAO9WKJFKMDFRNETDQBYVZOVCRGFS99999

The raw trytes can be seen by clicking on “Raw trytes (click to show)”

Signing and verification:

Those who spend the tokens authorize the spending by signing the bundle hash. IOTA signing scheme is based on Winternitz signing scheme where checksum is replaced with the following requirement:

If we assign the following numeric values to the trytes:

9 = 0
A = 1
B = 2
...
L = 12
M = 13
N = -13
O = -12
...
Y = -2
Z = -1

then the sum of the trytes of each 27-tryte bundle hash chunk must be equal to 0. This ensures that a change of a single tryte must be compensated by a change of some other tryte in the opposite direction thus requiring to break the one-wayness.

Only 1 of 2^20 bundle hashes satisfies the requirement, this is why the final design contains “bundle nonce” field and some PoW must be done by incrementing the field value and recomputing the bundle hash. The current design doesn’t require that because of a technique added to weaken the signing scheme and disguised as a bundle normalization.

The bundle normalization is done by splitting the bundle hash into 27-tryte chunks and incrementing or decrementing the first few trytes until the sum becomes 0 (https://github.com/iotaledger/iri/blob/dev/src/main/java/com/iota/iri/hash/ISS.java#L106)

.
Each 27-tryte bundle hash chunk is signed with a separate transaction. 2-of-2 addresses sign only 2 of 3 chunks thus reducing the collision resistance from 243/2 trits (~384/2 bits) to 162/2 trits (~256/2 bits).

Signing and verification is done as in the classical Winternitz scheme where each 243-trit fragment of the private key signs a single tryte of the bundle hash thus requiring from 0 to 26 Curl invocations. Upon a verification the obtained public key is hashed to get the corresponding digests which are then hashed to be compared to the address.
The hash of the public key fragments (aka digest) is used as a part of a higher level protocol protecting against an adversary successfully breaking Curl-P-27 one-wayness. Some other techniques are used to reduce the security requirements of IOTA signing scheme to one-wayness. All these things are planned to be outlined in a specification we are currently writing. Before revealing this to the general public we are still working on the final design of IOTA, as such I would appreciate it if you would keep this information private and not share it with anyone outside of this group.

Sergey

Letter #24
Author: Sergey Ivancheglo
Date: 30th of July

Hi, Ethan
I'm working on the document addressing your claims.

On the 19th of July you wrote:

> 1. Curl's collision and second preimage resistance are broken when used as a hash function

During the verification of your claim that Curl's second preimage resistance was broken I faced a little problem. Your example of a collision containing the lyrics of "Push It To The Limit" seems to not satisfy the requirement that (from Wikipedia) "given an input M1 it should be difficult to find different input M2 such that hash(M1) = hash(M2)."

While the lyrics might show that M1 was indeed "given" to you, "RETHT9ES9HRCUITBHVCUHOBPUUUHT9PHLUNWRWGKBKF9YUMDWRXTRVGZHFZEHGATZXZAUPGVEKNMQXFVRXHF9QJQHUTILIPIXUYRVSJEIOJDRIUVWMUABSIKIBAKENE9KVFJUEQUHFRVGELFGJIDXQARWH99XTORHXRETHT9ES9HRCUITBHVCUHOBPUUUHT9PHLUNWRWGKBKF9YUMDWRXTRVGZHFZEHGATZXZAUPGVEKNMQXFVR" in front of it ruins the demonstration. A better proof of your claim is required, could you, please, provide it by finding M2 for "9ABCDEFGHIJKLMNOPQRSTUVWXYZ9ABCDEFGHIJKLMNOPQRSTUVWXYZ9ABCDEFGHIJKLMNOPQRSTUVWXYZ" used as M1?

This is important because googling for "second preimage hash" I found https://security.stackexchange.com/questions/69405/difference-between-second-pre-image-resistance-and-collision-resistance-in-crypt/69409 which among the other things states:

> A second-preimage is also a collision, but we keep the concept distinct because second-preimages are supposed to be substantially harder. If the hash function has an output of n bits and is "perfect" (no known weakness), then the cost of finding a collision is 2^(n/2), while the cost of finding a second-preimage is 2^n (i.e. a lot more).

and

> If the attacker sees a valid, signed message m, then he may want to find a message m' that hashes to the same value. This is the second-preimage model. For the signature system to be robust, the hash function must provide second-preimage resistance. Collision resistance, on the other hand, is not necessary in that case.

and

> In the case of signatures, you need at least second-preimage resistance; but if the context is such that the attacker can obtain signatures on data that he chooses, then collision resistance is also needed.

In IOTA an attacker doesn’t choose the signed message, the chance to guess it correctly is negligible.

Of course, someone might question the validity of the quoted words, but the person who answered that Stack Exchange question has the reputation score above 136000. Judging by your words, namely:

> As a participant in the SHA-3 hash function contest who broke one of the 51 Round-1 SHA-3 proposals and who worked on security proofs for another SHA-3 proposal I can say with some authority that using the sponge construction and showing statistical properties of the transformation function is not sufficient to ensure security.

and your references to the opinion of other experts:

> Ask any cryptographer, they will tell you running custom crypto primitives in production is very dangerous and violates basic industry best practices. The number one rule of cryptography is "Don't invent your own."
> http://www.lauradhamilton.com/10-cryptography-mistakes-amateurs-make
> https://twitter.com/petertoddbtc/status/714447656928415744

I can safely assume that we can rely on the reputation score as on the measurement of the correctness.

All the above makes me wonder if we can use any excuse for the transition other than necessity to use a well-studied and vetted cryptographic primitive.

PS: If it takes too much time for a successful second preimage attack on Curl-P-27, I think, we might go with Curl-P-3 (with only 3 rounds).

Sergey

Letter #25
Author: Ethan Heilman
Date: 30th of July

The definition given on crypto.stackexcahnge is an informal and is intended to give general gist of what a second preimage attack is. For three formal definitions seehttp://web.cs.ucdavis.edu/~rogaway/papers/relates.pdf (page 6).

For a non-negligible subset of the range, we can find second-preimages. If this breaks second-preimage resistance of curl depends on how that is defined and is completely unimportant for our attack.

>In IOTA an attacker doesn't choose the signed message, the chance to guess it correctly is negligible.

We presented an EU-CMA (Existential Unforgability - Chosen Message Attack) which is viewed as a basic security requirement of a signature scheme. As the name would imply this is a chosen message attack in which the attacker chooses the message. The EU-CMA security of your signature scheme depends on the collision resistance of curl.

Letter #26
Author: Sergey Ivancheglo
Date: 30th of July

Hi Ethan,

> For three formal definitions
see http://web.cs.ucdavis.edu/~rogaway/papers/relates.pdf (page 6).
Thank you for pointing me to the exact page. There are 3 definitions, could you tell which one you used in your claim that the second-preimage resistance was broken? This ought to be done to have the same level of formalism.
> For a non-negligible subset of the range, we can find second-preimages.
What does make you say that you can find second-preimages for a non-negligible subset? From one example it makes sense to assume that you can find collisions only for messages beginning with “RETHT9ES9HRCUITBHVCUHOBPUUU”, this would explain why “RETHT9ES9HRCUITBHVCUHOBPUUU” is met again in 162 trytes.
> If this breaks second-preimage resistance of curl depends on how that is defined and is completely unimportant for our attack.
It’s you who already chose the definition, so please provide the definition to let us verify the claim. It’s great if you indeed found an easy way to generate collisions, I hope your approach can be extended to 3-argument Curl thus helping us to unlock the repudiation feature. It’s the only feature which protects end-users
against https://en.wikipedia.org/wiki/Key_disclosure_law (where the anonymity feature fails).
> We presented an EU-CMA (Existential Unforgability - Chosen Message Attack) which is viewed as a basic security requirement of a signature scheme.
I’m pretty sure it’s only for a spherical signature scheme in vacuum. Quick googling returns https://en.wikipedia.org/wiki/Digital_signature_forgery#Existential_forgery_.28existential_unforgeability.2C_EUF.29 where we can see that “Nevertheless, many state-of-art signature algorithms allow existential forgery.” Unless you claim that those words are wrong, I think we are in agreement that your phrase should be read as "which is viewed as a basic security requirement of SOME signature schemes".

Sergey

Letter #27
Author: Ethan Heilman
Date: 31st of July

If you don't find my statements credible then talk to a cryptographer you trust at a partner University and ask them if you should use a signature scheme whose EU-CMA security is broken.

Probably best not to use informal stackoverflow answers and Wikipedia for understanding the security of your system.

Letter #28
Author: Ethan Heilman
Date: 31st of July

>Thank you for pointing me to the exact page. There are 3 definitions, could you tell which one you used in your claim that the second-preimage resistance was broken? This ought to be done to have the same level of formalism.

In regards to second preimage I am using the first and easiest to satisfy definition. I believe that our attacks meet the first definition but I have not made a formal study of this yet as I prioritized notifying the IOTA team. In any public report we plan to exactly qualify the nature of our attacks. In any event our attack on the EU-CMA security does not depend on having second preimage attacks.

Letter #29
Author: Ethan Heilman
Date: 31st of July

Has the IOTA project assigned a CVE id for this vulnerability?
Thanks,
Ethan

Letter #30
Author: Sergey Ivancheglo
Date: 31st of July

Hi Ethan,

> If you don't find my statements credible then talk to a cryptographer you trust at a partner University and ask them if you should use a signature scheme whose EU-CMA security is broken.
It's near-impossible to find an expert having experience in application of cryptography to real-world use-cases targeted by IOTA. I questioned the credibility of your statements because I had spotted few signs of a shallow analysis (which you confirmed in the yesterday's letter by "I have not made a formal study of this yet as I prioritized notifying the IOTA team"). Your letters titled "Responsible Disclosure: Cryptographic Weaknesses in the Curl hash function in IOTA" sounded pretty official and I thought I had to address everything. Now I see that I was wrong, I'm providing 20 statements which are still not fully processed, please, inform me which statements (the numbers) are still actual, all the others will be considered retracted and won't be mentioned in the document.

#1: [Jul 15] We have found serious cryptographic weaknesses in the cryptographic hash function curl used by IOTA, curl.
#2: [Jul 15] These weaknesses threaten the security of signatures and PoW in IOTA as PoW and Signatures rely on curl to be pseudo-random and collision resistant.
#3: [Jul 15] (Thus if the first N message blocks are all zeros, then the hash output and the state will be all zeros.) This allows the creation of an unlimited number of collisions and second-preimages.
#4: [Jul 15] As message padding is not used in curl, we can construct messages of different lengths that result in the same sponge state.
#5: [Jul 15] The security definitions for hash based signatures assume that the hash function is pseudo-random.
#6: [Jul 15] However curl has non-random properties.
#7: [Jul 15] Note also the substring 99999 should a occur very infrequently, but we can anecdotally observed values of the initial state bleed into the final hash message.
#8: [Jul 15] Furthermore we have observed clear difference patterns in the hash output when each third trit is different between two messages.
#9: [Jul 15] You can break PRness without breaking one-wayness.
#10: [Jul 15] Based on the per round diffusion we believe if extend this attack to result in a single difference until the 18th round, we will be able to create collisions with high probability for random states by ensuring that all differences that exist only occur with state[0:243] at the end of a transform. It should work for any number of transform invocations greater than 1.
#11: [Jul 19] Curl's collision and second preimage resistance are broken when used as a hash function.
#12: [Jul 19] IOTA isn't using curl as a hash function but instead using it as a function which maps fixed length tryte strings to fixed length tryte strings.
#13: [Jul 19] Even when used in this setting Curl is not pseudo-random.
#14: [Jul 19] As far as I can tell most hash-based WOTS schemes can be reduced to the PRF property of the hash function.
#15: [Jul 22] We can now create collisions of messages of the same size.

#16: [Jul 22] We note these collisions result from the exceptionally poor performance of curl's transform function when attacked by textbook differential cryptanalysis attack.
#17: [Jul 22] We present the following attack on the EU-CMA security of the signature scheme used in IOTA.
#18: [Jul 23] …your signing scheme EU-CMA forgery depends on the collision resistance of curl.
#19: [Jul 30] For a non-negligible subset of the range, we can find second-preimages.
#20: [Jul 30] The EU-CMA security of your signature scheme depends on the collision resistance of curl.

> In any event our attack on the EU-CMA security does not depend on having second preimage attacks.
I have a feeling that you refuse to accept existence of cryptographic protocols not mentioned in the textbooks read by you. I can imagine an EU-CMA insecure scheme which is broken only after second-preimage is broken.

> Has the IOTA project assigned a CVE id for this vulnerability?
No. It's unclear how many vulnerabilities have been found, we may need to use such naming convention that would allow grouping of several vulnerabilities. We prefer to read the draft of your formal study before assigning the id.

Sergey

Letter #31
Author: David Sønstebø
Date: 4th of August

Hey Neha, Ethan and Tadge,

We are still awaiting a full response to this mail before we can go public. As you know we already prepared a migration due to upgrading to Curl 2 for months, but your claimed attack accelerated our priorities, but your sudden reluctance/absence in verifying the claims make us a bit weary. We have prepared everything and are currently writing up our update/migration blog post/newsletter to the community, but beyond what we have already communicated for months we were expecting a thorough breakdown of Curl, which has not happened yet...

We are already teamed up with world leading sponge construction hash function experts who are awaiting Ethan's discovery, but for almost a week now there has been no communication...

Before the next phase we need Ethan to respond to all of Sergey/CfBs concerns, so that we can publicly display the concerns in a fully transparent manner.

Letter #32
Author: Ethan Heilman
Date: 5th of August

>We have prepared everything and are currently writing up our update/migration blog post/newsletter to the community

On what day are you planning on publishing this information?

>Before the next phase we need Ethan to respond to all of Sergey/CfBs concerns, so that we can publicly display the concerns in a fully transparent manner.

Sergey's long list questions didn't seem to serve much of a purpose but I'm happy to summarize our findings.

We showed practical examples for the following:
(1). If curl is used as a cryptographic hash function, two methods for finding trivial collisions exist: (a). Collisions which exploit the failure to apply message padding and (b). Collisions which exploit the fact that an all zero message is a fixed point in curl enabling multi-collisions between messages of different lengths.
(2). Curl is non-pseudorandom, as certain inputs which are related by a bit-rotation results in outputs which are related by a bit-rotation. This does not exploit the weaknesses in (1), but relies on additional weaknesses in the curl function design.
(3). Curl's s-box is vulnerable to differential cryptanalysis, using this vulnerability we can create full-state collisions in messages of the same size. These collisions do not exploit the weakness in message padding or the all zero-fixed point. This does not exploit the weaknesses in (1) or (2), but relies on additional weaknesses in the curl function design namely weak per round diffusion and bad s-box design.

Based on a understanding of the IOTA signature scheme from reading of the code and conversations via email we also have the following attack:
(4). IOTA uses a signature scheme which hashes messages and then applies WOT-S-like operations on these hashes. Using (3) we can break the EU-CMA security of this signature scheme.

>We are already teamed up with world leading sponge construction hash function experts who are awaiting Ethan's discovery, but for almost a week now there has been no communication…

We have provided practical collisions on messages of the same length which we sent to you some time ago and showed how this breaks EU-CMA security of IOTA WOT-S-like signature scheme. What discovery are you waiting on?

Which cryptographers have you been talking to? Can you put us in touch with them?

Letter #33
Author: Sergey Ivancheglo
Date: 5th of August

Hi Ethan,

> On what day are you planning on publishing this information?
David or Dominik will inform you about this in another letter.
> Sergey's long list questions didn't seem to serve much of a purpose
The purpose was to get which statements were still actual. You listed none, so I take it as all of them being retracted.
> (1). If curl is used as a cryptographic hash function, two methods for finding trivial collisions exist: (a). Collisions which exploit the failure to apply message padding and (b). Collisions which exploit the fact that an all zero message is a fixed point in curl enabling multi-collisions between messages of different lengths.
The both are caused by an incorrect usage of Curl (the length should be absorbed first).
> (2). Curl is non-pseudorandom, as certain inputs which are related by a bit-rotation results in outputs which are related by a bit-rotation. This does not exploit the weaknesses in (1), but relies on additional weaknesses in the curl function design.
This is caused by an incorrect usage of Curl (the length should be absorbed first).
> (3). Curl's s-box is vulnerable to differential cryptanalysis, using this vulnerability we can create full-state collisions in messages of the same size. These collisions do not exploit the weakness in message padding or the all zero-fixed point. This does not exploit the weaknesses in (1) or (2), but relies on additional weaknesses in the curl function design namely weak per round diffusion and bad s-box design.
This is a design choice (fine-graininess). 15 rounds of Curl-P have diffusion and S-box non-linearity equivalent to 3 rounds of Keccak. Unlike Keccak, Curl-P uses identical algebraic representations for all trit transitions to make all attacks as simple as possible thus making finding the min number of rounds for a required security level trivial. Curl-P state size being only 72% of Keccak state size makes all analyses even simpler. (Not sure this information is really necessary, you should have noticed all this already, I provide it in case if you just fed a binary version of Curl into existing cryptanalytical software.)
> (4). IOTA uses a signature scheme which hashes messages and then applies WOT-S-like operations on these hashes. Using (3) we can break the EU-CMA security of this signature scheme.
We would like to see a proof of this claim to make sure that you took the higher level protocol into account.
> We have provided practical collisions on messages of the same length which we sent to you some time ago and showed how this breaks EU-CMA security of IOTA WOT-S-like signature scheme. What discovery are you waiting on?
The provided collisions didn't break EU-CMA security, we are waiting for an algorithm which indeed allows to break that.
> Which cryptographers have you been talking to? Can you put us in touch with them?
You haven't supplied us with the details we requested so therefore they are very confused as well because your finding was a single internal collision while you were mistakenly claiming that it broke second-preimage resistance of Curl-P-27. Your summary of the findings shows that you finally saw that too. We hesitate to inform them about your claim of breaking EU-CMA security of IOTA signatures because we haven't

seen your proof yet and are pretty sure that you are wrong again. Could we see the draft of the paper you are planning to publish?

Sergey

Letter #34
Author: Ethan Heilman
Date: 5ht of August

Hi Sergey

>The purpose was to get which statements were still actual. You listed none, so I take it as all of them being retracted.
We have not retracted any statements. We provided a summary of our results which we continue to stand by. We excluded second-preimage attacks from the summary since although we believe we have broken the second-preimage resistance of curl we have not formally quantified the second-preimage attack.

>We hesitate to inform them about your claim of breaking EU-CMA security of IOTA signatures because we haven't seen your proof yet and are pretty sure that you are wrong again.
Why not let them judge for themselves? We have provided detailed description of our attack such that any serious-cryptographer could tell if we have broken the EU-CMA security of IOTA's signature scheme.

Who are they? How does this agree with your earlier statement that

>It's near-impossible to find an expert having experience in application of cryptography to real-world use-cases targeted by IOTA

In regards to breaking the EU-CMA security of IOTA:
As you said in an earlier email "Signing and verification is done as in the classical Winternitz scheme where each 243-trit fragment of the private key signs a single tryte of the bundle hash thus requiring from 0 to 26 Curl invocations."

This agrees with what we have seen in the code. As a result a collision in messages of the same length breaks EU-CMA security of the signing scheme since unlike WOT-S the EU-CMA security of the IOTA signing scheme depends on collision resistance of the hash function.
Can you provide some details on the proof you are waiting for?

Thanks,
Ethan

Letter #35
Author: Sergey Ivancheglo
Date: 5th of August

Hi,
> We have not retracted any statements.
Here is a list I'm currently working on, is anything missing?

S01: We have found serious cryptographic weaknesses in the cryptographic hash function curl used by IOTA, curl.
Only a single internal collision was demonstrated for a weakened version of Curl-P. It was expectable, especially after a collision on a Keccak equivalent was found (http://keccak.noekeon.org/crunchy_mails/coll-r6-w1600-20170226.txt).

S02: These weaknesses threaten the security of signatures and PoW in IOTA as PoW and Signatures rely on curl to be pseudo-random and collision resistant.
Awaiting a proof.

S03: This allows the creation of an unlimited number of collisions and second-preimages.
Incorrect usage of Curl.

S04: Since the state is the same regardless of the number of prefixed zero input blocks we can create more complex collisions and second-preimages.
Incorrect usage of Curl.

S05: As message padding is not used in curl, we can construct messages of different lengths that result in the same sponge state.
Incorrect usage of Curl.

S06: We have also found another form of non-randomness. Namely two similar messages result in states that are different only by a positional shift.
Incorrect usage of Curl.

S07: Note also the substring 99999 should a occur very infrequently, but we can anecdotally observed values of the initial state bleed into the final hash message.
The same can be observed for SHA-256:
SHA256(*999999*2d224a9a45e7c8eb185971e53574024801252322d93550f4f5987addf8) = 567be78f9537b18990e32b6164fe06*999999*19b50c7a23556cc6aeb85dbe4778
SHA256(*999999*16e09f623d0cc86cf5eba24e9e1d14f86f12a6a1ea46a4fc984b46c85d) = 699ec27b2b9e4a800841bccd6a4bdd*999999*2a03cceefa6b014dc3d2edd35f60
SHA256(*999999*74d828d6d9d420acd5f1fcf44bff94b45cb6a3fe539fb20e245cbcdcea) = 6b16590e1583885741dbd77bb4d3dd*999999*33d03cb2c20d13ac7cf134e9da76

S08: Furthermore we have observed clear difference patterns in the hash output when each third trit is different between two messages.
An example wasn't provided.

S09: This paper [https://eprint.iacr.org/2011/191.pdf] reduces the Winternitz one-time signature scheme to the PRF property of a function not onewayness.

The abstract states: “We show that the Winternitz one-time signature scheme is existentially unforgeable under adaptive chosen message attacks when instantiated with a family of pseudo random functions. Compared to previous results, which require a collision resistant hash function, our result provides significantly smaller signatures at the same security level.” Which shows that we can reduce WOTS to different properties. One can reduce WOTS to one-wayness by using the technique from https://pdfs.semanticscholar.org/aaf3/602cdcd40276d2239f22f2eeaf5c2d45446f.pdf.

S10: [W]e will be able to create collisions with high probability for random states by ensuring that all differences that exist only occur with state[0:243] at the end of a transform. It should work for any number of transform invocations greater than 1.
Awaiting for an example of such collision.

S11: Curl's collision and second preimage resistance are broken when used as a hash function.
A single internal collision for a weakened version was demonstrated. No second-preimage attack was demonstrated.

S12: IOTA isn't using curl as a hash function but instead using it as a function which maps fixed length tryte strings to fixed length tryte strings.
According to Wikipedia “A hash function is any function that can be used to map data of arbitrary size to data of fixed size”. Unclear if the difference between “arbitrary size” and “fixed length” is really significant. Confused what this statement could mean.

S13: Even when used in this setting Curl is not pseudo-random.
Awaiting a proof with correct usages of Curl.

S14: It is unclear if this breaks the signature scheme used by IOTA because the signature scheme used by IOTA is not well documented and does not have security arguments.
Some information was provided, no requests for more information were received.

S15: As far as I can tell most hash-based WOTS schemes can be reduced to the PRF property of the hash function.
See S09.

S16: We can't attack the signature scheme until we understand it, and understanding a signature scheme from source code alone is a time intensive process.
An attack was provided later which implies that the signature scheme had been understood. Should, probably, be grouped together with S14.

S17. This appears to break the standard definition of security for signature schemes called EU-CMA (Existential Unforgeability under a Chosen Message Attack) [http://www.cs.tau.ac.il/~canetti/f08-materials/scribe8.pdf].
Awaiting for a proof.

S18: We note these collisions result from the exceptionally poor performance of curl’s transform function when attacked by textbook differential cryptanalysis attack.
This is a design choice because making attacks simple allows to pick the min number of rounds for a required security level without worrying that the attacks will become more

complex in the future (the evolution of slide attacks is a good example of how this happens).

S19: We base this on the following description of the signature scheme used in IOTA. The described attack can't be applied to IOTA because it doesn't take into account higher level protocols.

S20: We used the lyrics to the 80's hit single "push it to the limit" in the colliding messages to demonstrate that we fully collide the internal state of curl and thus have arbitrary control over most of the message. We note that this internal state collision enables us to instantly create millions of other colliding messages.
These "millions of other colliding messages" cover less than 3^(-333) part of all possible messages hence they can appear with negligible probability even if we generate transactions with 1 billion TPS during 1 billion years.

S21: Can you explain your signature scheme then? When generating a signature do you or do you not hash the message? We have looked in multiple implementations and it appears that you do. If you do then our attack holds and your signing scheme EU-CMA forgery depends on the collision resistance of curl.
It also depends on higher level protocols, as I have already described in previous correspondence and will elaborate on in the next points.

S22: These attacks we have developed will always get better. The current results are from roughly 20 hours of work.
Satoshi's invention led to a paradigm shift in a lot of areas. Unfortunately, some cryptographers still didn't fully embrace that and keep sticking to obsolete security assumptions. In our very case it means that window for an attack is very small (maximum - until a transaction is confirmed).

S23: For a non-negligible subset of the range, we can find second-preimages.
You use an unorthodox definition of "non-negligible" because odds to select a message which can be second-preimage attacked are below 1 of 871896424859609582029110705858607716969640724047317500855252194379909670937234399434755499068316831167910552256656 27 for a single attempt. (Here I assume that you are planning to use a method based on an internal collision.)

S24: If this breaks second-preimage resistance of curl depends on how that is defined and is completely unimportant for our attack.
All the provided definitions show that second-preimage resistance of Curl hasn't been broken. Awaiting for more definitions to check.

S25: The EU-CMA security of your signature scheme depends on the collision resistance of curl.
Awaiting the description of a reduction of higher level protocols to collision resistance of Curl.

S26: In any event our attack on the EU-CMA security does not depend on having second preimage attacks.
Awaiting the description of the attack.

> We excluded second-preimage attacks from the summary since although we believe

we have broken the second-preimage resistance of curl we have not formally quantified the second-preimage attack.
Looking forward to seeing these attacks. If you are unable to break second-preimage resistance for even 27 rounds then this will mean that the versions with a larger number of rounds are safe for sure, we'll need to reduce the number to 15+ rounds then.

> Why not let them judge for themselves?
Their time is expensive so we have to do some basic filtering.

> We have provided detailed description of our attack such that any serious-cryptographer could tell if we have broken the EU-CMA security of IOTA's signature scheme.
What you have provided is not applicable to IOTA's signature scheme because you ignored higher level protocols.

> Who are they?
As you may know, official cooperation requires weeks of paperwork, I think you understand why we can't reveal their names now.

> How does this agree with your earlier statement that
Well, I would answer honestly but this may be considered as a personal insult, so before that I'd like to get the absolution from you in advance.

> As a result a collision in messages of the same length breaks EU-CMA security of the signing scheme since unlike WOT-S the EU-CMA security of the IOTA signing scheme depends on collision resistance of the hash function.
It seems to me you didn't receive my letter to Neha, here is the essential part:

https://github.com/iotaledger/iri/blob/dev/src/main/java/com/iota/iri/TransactionValidator.java#L72
Here the collision would invalidate a transaction if it adjusted "timestamp" to an invalid value.
https://github.com/iotaledger/iri/blob/dev/src/main/java/com/iota/iri/TransactionValidator.java#L77
Here the collision would invalidate a transaction if it adjusted "value" to an invalid value.
https://github.com/iotaledger/iri/blob/dev/src/main/java/com/iota/iri/BundleValidator.java#L35
Here the collision would invalidate a bundle if it broke "currentIndex" sequence or "lastIndex" didn't match the real number of the transactions in the bundle.
https://github.com/iotaledger/iri/blob/dev/src/main/java/com/iota/iri/BundleValidator.java#L43
Here the collision would invalidate the bundle if it changed the transferred value.
https://github.com/iotaledger/iri/blob/dev/src/main/java/com/iota/iri/BundleValidator.java#L73
Here the collision would invalidate the bundle if it changed "address" field of a multi-sig address (there are no single-sig addresses in official wallet, all addresses are 2-of-2 ones, but custom software might generate single-sig addresses).
https://github.com/iotaledger/iri/blob/dev/src/main/java/com/iota/iri/BundleValidator.java#L79
Here the collision would invalidate the bundle if it changed the spending address.
I don't have the source code of the Coordinator on this notebook, the code is not public, but I can get it tomorrow if it's important. The Coordinator is used as an extra protection

measure. Particularly, it stores all transactions that reach it, this allows us to recover iotas sent to addresses with typo (we detect it because users usually publish the addresses on the tangle before usage), during the latest snapshot we used the Coordinator to restore iotas sent to addresses generated with the previous (now obsolete) address generation scheme.

> Can you provide some details on the proof you are waiting for?
We'd like to see how you avoid generation of collisions which wouldn't pass the verifications above. Also we'd like to know if the collisions can be generated in real-time to win the propagation race against legitimate transactions.

Sergey

Letter #36
Author: Ethan Heilman
Date: 5th of August

>It seems to me you didn’t receive my letter to Neha, here is the essential part:
EU-CMA security does not require that the messages pass validation checks outside of the signature scheme.

>As you may know, official cooperation requires weeks of paperwork, I think you understand why we can’t reveal their names now.

I have never heard of any paper work being required to talk to a cryptographer, nor have I heard of any cryptographers requesting that their names be withheld to answer basic cryptography questions. It is very unusual.
This is not intended as an insult but the list of questions asked here show a lack of understanding of the basics of how cryptographic primitives and schemes are assessed. As you don't seem to believe me on such issues I would ask that you reach out to a cryptographer, if one isn't available I would be willing to put you in touch with one to give you a second opinion.

Letter #37
Author: David Sønstebø
Date: 5th of August

Hey Ethan,

Regarding the issue of cryptographers, like we mentioned weeks ago, we have been working on finalizing Curl 2 with leading cryptographers with special expertise in the realm of sponge family for a long time. This is simply a formality. They represent a company, not an academic institution, so we want to clear it with them beforehand before releasing their affiliation, which I hope you can understand. Beyond this, those details we have requested is something they really want also to be able to make a thorough review of your claims.

We asked you a week or so ago to disclose who these other cryptographers you had discussed this with was, you never replied to that, but now that you are offering to do so for a second opinion, I would absolutely welcome that.

I propose you put us in touch with those right away, then I'll get the paperwork sorted with the team we are working with by Monday or Tuesday and then you can also talk with them.

All the best,
David

Letter #38
Author: Sergey Ivancheglo
Date: 5th of August

Hi, Ethan

> EU-CMA security does not require that the messages pass validation checks outside of the signature scheme.

You mean "…outside of a spherical signature scheme in vacuum", don't you? In our letters we are discussing a concrete signature scheme used in IOTA.

> This is not intended as an insult but the list of questions asked here show a lack of understanding of the basics of how cryptographic primitives and schemes are assessed.

I can explain why these questions were asked, just need the absolution from you (in case if you are offended by my words, lack of English vocabulary makes me sound pretty blunt). Could I have it?

> As you don't seem to believe me on such issues…

We have taken these issues very seriously from day one, recall that we contacted you, among a lot of other people in the space, to review Curl several months ago. On top of this we are taking concrete action, but when it comes to these claims we want to stick to the Popperian principles of empirical verification rather than rely on belief, which belong in the realm of religion. Again this should simply be interpreted as us taking this very seriously and not wanting to let the devil hide in the details.

Sergey

Letter #39
Author: Neha Narula
Date: 5th of August

You have no such absolution if your response takes us beyond the level of professional and civil discourse. If anyone personally insults a member of my team, we will have to cease communication.

Letter #40
Author: Sergey Ivancheglo
Date: 5th of August

Hi Neha,

don't worry. From our side this is only about facts and making sure we understand each other properly. By "absolution" I am simply attempting to ensure that we indeed stick to the civil discourse, like I mentioned English is my third language, and I was getting worried that my blunt wording might come off as offensive. I believe this entire chain of letters make it evident that we are focused on a civil professional relation.

PS: I even unsure that putting "Hi, <name>" everywhere is suitable, I wish I took normal English classes instead of learning it by reading Java documentation)

Sergey

Letter #41
Author: Sergey Ivancheglo
Date: 6th of August

Hi Ethan,

This morning I read an interesting blog post - https://medium.com/@avivzohar/responsible-disclosure-in-cryptocurrencies-74833fc4f211. Unfortunately, it ended abruptly, the author admitted that he didn't know answers for the questions he had raised. But the post triggered something in my brain and the latter produced one scheme...

"I hope that we find clear and ethical practices that promote more secure and stable systems" - these were the very last words from that blog post. I know what should be added into the box with these practices, it's FORMALIZATION. Several letters back you and me were arguing if your findings can break the second-preimage resistance, the issue was solved after you provided those three formal definitions. I suggest to apply this method to your claim regarding EU-CMA of IOTA signature scheme.

You gave me this link - http://www.cs.tau.ac.il/~canetti/f08-materials/scribe8.pdf. On page 2 we can see a formal EU-CMA definition (not sure what steps 1-6 you chose for your proof to make sense because IOTA uses WOTS but I assume that step 7 is preserved):

7. Adversary wins if Ver(vk, m*, s*) = accept and m* wasn't previously signed by Sign

This is a step where your attack fails because of the higher level protocols. Particularly, the failure happens in some of those lines of the code that I sent to Neha and later re-sent to you. I'm pretty sure that if I manage to formalize the higher level protocols then we'll come to a consensus on this matter. I think the consensus is even inevitable. I'd like to know the date when you finalize your publication text, I'll do my best to have the formalization ready a few days earlier so we could get it included into your paper. There are approximately 5 days left, right?

PS: Aviv's blog post is really important to the industry of cryptoplatforms, what do you think of the idea to co-author the Postmortem of your responsible disclosure as a continuation of Aviv's post? We might include several quotes from our letter exchange to let the readers get real feeling of how all this happens.

Sergey

Letter #42
Author: David Sønstebø
Date: 7th of August

Hello Neha,

We have slightly moved up the timeline on our side to accommodate some of the entities in the ecosystem that rely on IOTA, which weren't able to participate in this discussion until now. The newly proposed timeline is as follows:

1) Tonight we will be announcing a new snapshot with the transition from Curl to Keccak

2) Tomorrow (8th of August) we will be doing a snapshot, requesting everyone to move their iotas. The wallets and the core clients are already ready and have been tested by the team & people from the community in the last week.

3) August 16th we will be doing a second snapshot as a security precaution.

In order to ensure that everyone involved comes out of this in a positive manner, we would like to review Ethan's paper before the official publication. It has become quite obvious that Ethan doesn't trust our team's expertise, but he demonstrated integrity by contacting us despite of working on a competing project (Paragon), and we feel obligated to protect his renome which may be harmed if he publishes a paper with incorrect information resulting from sheer misunderstandings. Judging by the letter exchange we are sure that he indeed got some aspects of IOTA's security wrong, which again reiterates the importance of him providing the details we have requested for 2 weeks now. Then we can finally file this into the academic annals.

All the best,
David

Letter #43
Author: Neha Narula
Date: 7th of August

Hi David,

Thanks for the timeline update. There are more responses forthcoming to the other emails.

How are you planning on describing the reason for the transition in your messaging to users?

Thanks,
Neha

Letter #44
Author: Neha Narula
Date: 7th of August

Ethan has nothing to do with Paragon the network/cryptocurrency. Ethan is associated with a completely different organization known as the Paragon Foundation. I think his renown will be fine.

Letter #45
Author: David Sønstebø
Date: 7th of August

Hello Neha,

Looking forward to the replies to the other mails, they should be elucidating the issue for us.

We will describe it as is. IOTA is a non-profit foundation that is all about transparency of process.

Re: Ethan and Paragon, I of course meant his role in the Paragon Foundation. We even have admission from Ethan that he is working on a 'similar project' from back in May when we asked him to audit Curl, where he adviced that we not disclose information that wasn't known, precisely due to the potential conflict he himself highlighted. This is a very sensitive matter, so I'd prefer we stear clear of assumptions.

Letter #46
Author: Neha Narula
Date: 7th of August

Great. Could you share how you will "describe it as it is"? What will you say is the reason for the transition?

Letter #47
Author: David Sønstebø
Date: 7th of August

Hey,

It will be quite straight forward, this is just another basic update in an evolving cutting edge project, nothing special, particularly because we need to see Ethan's paper and his answers to the details we have requested. If all he found was one of the anti-scam-copycat mechanisms then it's nothing extraordinary, another one of the security auditers we contacted around the same time as Ethan found another one too. This is to be expected from all new projects in their beta to maturation stage.

We have disclosed Curl's current state numerous times, like here in IOTA's Transparency Compendium and already considered using SHA-3 in the intermediary period since inception, we even discussed Curl and SHA-3 with the Keccak team since early 2015, so this is not some big thing for us, it's just a basic update to be on the extra safe side. Since it pertains to security it is something to always be taken very seriously, but it's not really some major finding, even if Ethan were to demonstrate an attack vector (which he so far has been unable to do).

We will also mention that we contacted this cryptographer (Ethan) back in May, but that he then declined due to working on a competing project, just for full transparency and disclosure. Accuracy and facts is extremely important to us, as should be abundantly clear by now.

In addition it is accompanied with other planned upgrades to the protocol, so the timing is great to just combine it.

Best,
David

Letter #48
Author: David Sønstebø
Date: 7th of August

Hey, quick question: since it's approaching 8 PM here in Europe I want to know whether we should name the 3 of you already now, or wait with that until we have discussed the details, received responses from Ethan and agree on the publication? I could refer to you as "the party that contacted us for responsible disclosure" for now if you wish.

Best,
David

Letter #49
Author: Neha Narula
Date: 7th of August

I'm a little unclear on how you plan on discussing us, so it's difficult for me to answer this question. Could you please share the section which references us, if not the whole post?

Letter #50
Author: David Sønstebø
Date: 7th of August

This is the relevant section:

One of the cryptographers we reached out to months ago to review Curl has disclosed that he is worried there might be a potential vulnerability in Curl. We have since had our internal team, as well as other cryptographers review it and asked the disclosing party for more information. While the party that did the responsible disclosure has been quite forthcoming, there are still some of the last details to be discussed more thoroughly with the respective teams in order to reproduce the claims and verify if there was even any vulnerability.

However, even though we have protection mechanisms in place that would render even most valid attacks useless in this 'training wheel stage' (due to the Coordinator and the higher-level protocol), as you are working on the cutting edge you have to take every precaution possible and always be on guard. Therefore we have made the simple decision to temporarily switch Curl with Keccak (SHA-3) for cryptographic signing in IOTA. This is something we had contemplated previously as well, and is now put in motion. Our stellar developers have been tremendous at executing this in a secure and expedient manner. More details on the technical changes will follow in a document tomorrow.

The party that contacted us will be releasing a publication of these potential results after we together nail down the details. We are thrilled about the upcoming publication as it will potentially provide deeper insight into Curl itself. Curl's origins date back over 2 years, since then we have engaged numerous cryptographers, in particular experts in the domain of sponge family hash functions, which Keccak and Curl both belong in, in order to further optimize and audit the final incarnation of Curl. We are very excited about this aspect of IOTA. Curl is a hash function specifically tailored for IoT, that also happens to be the world's first trinary one, so we spare no expense on this part of the project, as we deem it necessary for IOTA and IoT in general to realize its full potential.

Letter #51
Author: Neha Narula
Date: 7th of August

Thank you for sharing, David, this is really helpful.

At the moment, I think we're fine with not being mentioned by name in your post.

We'll make sure to also mention how quickly you got the new software release completed.

Our document is not done yet; we're still working on it.

Letter #52
Author: David Sønstebø
Date: 7th of August

Great to hear Neha. I respect your decision. Here is the full blog post

https://blog.iota.org/upgrades-updates-d12145e381eb

Looking forward to it, thanks.

Best,
David

Letter #53
Author: Neha Narula
Date: 8th of August

Hi IOTA team,

Here are two bundles which differ in one trit (Address[72] of the last transaction) which hash to the same value, and thus have the same signature. Both bundles pass your bundle.IsValid() checks in the go library.

I'd like to confirm with you the following:

1) The bundles do hash to the same value
2) The bundles are well formed according to your application validity checks (the "higher level protocols"). I have not tested this with your Java library.

We were able to produce these conflicting bundles in a few minutes.

Best,
Neha

```
var BUNDLE0 []giota.Trytes = []giota.Trytes{
        "999…<snip>…BKE",
        "T9E…<snip>…ROK",
        "AZZ…<snip>…TCO",
        "999…<snip>…CQR",
}

var BUNDLE1 []giota.Trytes = []giota.Trytes{
        "999…<snip>…BKE",
        "T9E…<snip>…ROK",
        "AZZ…<snip>…TCO",
        "999…<snip>…CQR",
}
```

Letter #54
Author: Sergey Ivancheglo
Date: 8th of August

Hi Neha,

We can't get your bundles to be validated successfully. It's unclear if it's a bug in the unfinished Go version or something else, we can't check it right now because of the preparations to the snapshot. The bundles fail in Java version because they don't form the correct chain of the transactions. The chain starts from the transaction with current index = 0, the trunk transaction should be set to the hash of the following transaction (with current index = 1) and so on. In other words the chain of trunk transaction references must go from the first to the last transaction. The trunk transaction of the very last transaction must reference any transaction with current index = 0, "999…999" is a safe choice. Branch transactions must reference any transactions with current index = 0 too.
Some other issue we noticed: Only one of the bundles can be confirmed (because they share some transactions) but redoing PoW would allow to confirm the both if there was enough money on the spending address balance. In any case the original variant doesn't allow to trigger a network consensus divergence because of Coordinator.

> 1) The bundles do hash to the same value
Yes.
> 2) The bundles are well formed according to your application validity checks (the "higher level protocols").
No. The chain of references should be fixed, it's easy.
> We were able to produce these conflicting bundles in a few minutes.
What are the mean and the variance of this process? A single number is not very helpful to assess probability of finding conflicting bundles while the window for an attack is open.

Sergey

Letter #55
Author: Neha Narula
Date: 8th of August

Might I trouble you to check these two bundles? They should have valid trunk transactions and valid PoW. I will try to get the java code running as it seems to do more checks.

Could you recommend a server for a tangle that we could connect to?

Letter #56
Author: Sergey Ivancheglo
Date: 9th of August

Maybe it's because of deep night here but I see strange things:

1. The both transaction sets seem to be identical
2. The declared bundle hash
is PMII9EP9GYVEEOHAYKW9N9WWHABYMK9IDOMEK9LTHNOJAEKACUDEQA9H
RXRVNVRSZAWC9HOW9GNMLGXUK while the actual bundle hash
is LAFLFDFYMSWKWHBEXYHQ9AFJATGGXR9MKF9ZXHGMZ9TSJJTFHZAAHLNS
WBEJFYBADYPHQCCIAUSJQJDME
3. The trunk transaction fields reference transactions generated with minWeightMagnitude 18 (15 would be enough) while actual transaction hashes are generated with MWM ~13

Do you see the same?

The following nodes are on the testnet (mainnet is already upgraded): udp://52.209.174.231:14800, udp://52.211.185.237:14800, udp://52.211.201.40:14800, udp://52.214.207.109:14800 (no mutual tethering). The testnet works with a dummy Coordinator but it's still useful for debugging.

If most of the bundles are generated within 2 minutes then there is no need to measure the variance, we can assume that with 100x more cores you'll get a delay of the same order of magnitude as the propagation time, just need to make sure that your algorithm is parallelizable to such degree. If you can't outpace a legitimate transaction on its way to Coordinator then you have to conduct an eclipse attack first. Ethan has experience in that.

Sergey

Letter #57
Author: Neha Narula
Date: 9th of August

Hi Sergey,

Sorry, you're not crazy -- I added a bug to my code when I started doing the trunk transaction chaining. I've since fixed it, and confirmed that these two bundles actually *are* different.

minWeightMagnitude -- I do not understand your code well enough to know what this should be. What should I set it to? Is what's in the current bundles sufficient (13) or should I set it to 15?

Letter #58
Author: Sergey Ivancheglo
Date: 9th of August

Hi Neha,

minWeightMagnitude should be set to 15 for the mainnet, it means that the last 15 trits of a transaction hash must be 0s (i.e. last 5 trytes must be 9s). I checked the bundles, the transaction hashes are
9Z9OZXAUCGRRYLLTNLJDALTQAFGRWACPVEIPOAPVV9WAIYZQIWQDUHXWITF
LZMJEEQEWYBMBNAWZX9999
UOFMPUZZOVYPAFYLEHKVI9T9WMMVDTE9TLDK9QXPXHY9ZNFNLLCVPRNF9YJ
TSYPNUEWPZBBKEDJLW9999
BCAXBUCQ9VQNPEFWSXXNHWNFJZYXZDYKTHTWJYEPHQYCZHNZUQQYNXIYA
PKDKCICZNOMEYERJSIBY9999
ZKDXLQTHOMLPCCSRGDYBBF9EYMOCIFEIXINNF9RENSEJUPB9UKVHVXPCKQ
MQUNWUWWWLBPGNJTWTD9999

and

JKFQJQFHCYQFPQTCWOZEWML9HDWYMZVMEKJWRBNDUSSEZIVUF99SLJNZY
JX9KRD9TLOI9ALWEOKKW9999
GCSRXZAQMAMNDRYTAHXAUWYIPOCRXJJEDCUULSTREL9IKSP9YEAWONRDV
EMXUQVPVQ9WTGPFJXQZZ9999
NMYQYBDWOFMY9CSPQTASNAPRDLZSHRWZNOENWGPLICFPXFKLUCMCMCJ9
PFDEKFIBQGFQQCPZDBERZ9999
HGJDFDGW9LXYHL99FOBOSOBUNIVVQKVDQZDJD9DTFGIIAFEOFRXHQ9DNSGJ
9FPLVPYE9ZCDJYIVSC9999.

But according to the chain of trunk transaction references they must be

<ANY>
WALUYLVJRDESKVWYOHVTHTB9XSWCEJGPWIWGR9OMYCDLTVQJVUJQVLD9X
HHQKSRPRRWGEORCFGGKZ9999
QSAQTCBJDVBZYAUZHQZVJOZUTQUNREOFBDHHJPPXCKWMBONLXVPDWJLM
VPMITLFIIHXBDLPVXGGE99999
EMBSIPCFGAVCZPX9BMVCMBXJXYDWJC9BDWZWUMFSCTMACUVHLJQZNAQX
YDBPAUXNZIYIAELVFTNRY9999

and

<ANY>
VTJGPEDIALNEUCFULFSIGQMSFBSDYUVKXVYYNCCGHAHPQTVFRYCDI9TORP
CBLZC9ULSWVGICTKGYD9999
CYMX9ELXYX9WCTHF9OAMBGEQDARMDGCRIBBEACV9ZLYVZNTUCEZXPHRKQ
NHQRHB99BSOXZTJLKQXB9999
GOSPZKDMTSBXYDX9PEZWKIPKZWHNZHNLLMJDNDZRTDE9ENTFJFTKVBQAAV
CKNPE9YPXQLWWVXNMOC9999.

For a general case if we have a bundle with transaction hashes AAA, BBB, CCC and DDD these transactions should look this way:

Tx AAA references BBB in “trunkTransaction” field
Tx BBB references CCC in “trunkTransaction” field
Tx CCC references DDD in “trunkTransaction” field
Tx DDD references 999…999.

Sergey

Letter #59
Author: Neha Narula
Date: 10th of August

Ah, I see the problem -- I was calculating the transaction hashes and doing proof of work *before* signing and copying over the signature from bundle0 to bundle1, but of course the signatures do not depend upon that, while the transaction hashes do depend upon the signatures.

Thank you for bearing with me in this back-and-forth. I hope that I've finally correctly constructed the two bundles:

var BUNDLE0 []giota.Trytes = []giota.Trytes{
"999…<snip>…BKE",
"T9E…<snip>…ROK",
"AZZ…<snip>…TCO","999…<snip>…CQR",
}

var BUNDLE1 []giota.Trytes =
[]giota.Trytes{"999…<snip>…BKE","T9E…<snip>…ROK",
"AZZ…<snip>…TCO",
"999…<snip>…CQR",
}

Letter #60
Author: Sergey Ivancheglo
Date: 10th of August

These bundles look identical to the previous ones.

Letter #61
Author: Neha Narula
Date: 10th of August

I apologize! That's what I get for sending an email in a rushed fashion. Thank your for continuing to bear with me.

I am really sure (this time) that these are different. The 27th tryte of the address of the last transaction differs -- one is an R, the other is an S (look for RR9C and RS9C).

The minWeight is 15.

I hope I set up the trunk transactions appropriately. I can't seem to get a Java test to work correctly (I haven't written Java in a long time).

Thanks,
Neha

```
var BUNDLEn0 []giota.Trytes = []giota.Trytes{
        "999…<snip>…BKE",
        "T9E…<snip>…ROK",
        "AZZ…<snip>…TCO",
        "999…<snip>…CQR",
}

var BUNDLEn1 []giota.Trytes = []giota.Trytes{
        "999…<snip>…BKE",
        "T9E…<snip>…ROK",
        "AZZ…<snip>…TCO",
        "999…<snip>…CQR",
}
```

Letter #62
Author: Sergey Ivancheglo
Date: 11th of August

They look identical to the previous ones.

Letter #63
Author: Neha Narula
Date: 11th of August

Then I sent you the right ones before -- these two bundles are absolutely not identical. I bolded the difference below.

Do they validate in the java client?

Letter #64
Author: Sergey Ivancheglo
Date: 11th of August

Hi Neha,

You provided 3 distinct pairs of bundles, they all failed validation. While the latest pair contains elements which differ in a single tryte they are identical to the pairs from the previous letters. To better explain my thought I prepared a list of the bundle pairs and their corresponding SHA-256 hashes. The hashes were generated by concatenating all the trytes without line breaks and feeding into http://www.xorbin.com/tools/sha256-hash-calculator. The pairs go in chronological order.
BUNDLE0 SHA-256 hash =
9a3288b913d1c7d7e3d7135fe21b4852ec9512bf55ee9047ebaf25349c4101b4
BUNDLE1 SHA-256 hash =
008123e1aace3cf54987ffcbeccad6a62a5fce20a3bc322f4898ce5eec0c9da5

bundle0 =
055b6c0a16492f8f95a12ab737d0253833706e6f7505e4997e0497198adbb0b1
bundle1 =
055b6c0a16492f8f95a12ab737d0253833706e6f7505e4997e0497198adbb0b1

BUNDLE0 = dfbb8f968a739df4fc5bf810defe1536f3dd78b2184aefcca37e1de55aa2af54
BUNDLE1 =
e48567d8d743b2f3bb17a0c303dc6a510678cc5f311f2a3f3eeef0fafa585d90

BUNDLE0 = dfbb8f968a739df4fc5bf810defe1536f3dd78b2184aefcca37e1de55aa2af54
BUNDLE1 =
e48567d8d743b2f3bb17a0c303dc6a510678cc5f311f2a3f3eeef0fafa585d90

BUNDLEn0 =
dfbb8f968a739df4fc5bf810defe1536f3dd78b2184aefcca37e1de55aa2af54
BUNDLEn1 =
e48567d8d743b2f3bb17a0c303dc6a510678cc5f311f2a3f3eeef0fafa585d90

Best,
Sergey

Letter #65
Author: Neha Narula
Date: 11th of August

Got it -- there was a misunderstanding. I thought you were saying the *bundle0* and *bundle1* were identical, and ALSO identical to the ones sent before.

Based on an email you sent previously, I have a question. You said:

> But according to the chain of trunk transaction references they must be

> <ANY>
>
WALUYLVJRDESKVWYOHVTHTB9XSWCEJGPWIWGR9OMYCDLTVQJVUJQVLD9X
HHQKSRPRRWGEORCFGGKZ9999
>
QSAQTCBJDVBZYAUZHQZVJOZUTQUNREOFBDHHJPPXCKWMBONLXVPDWJLM
VPMITLFIIHXBDLPVXGGE99999
>
EMBSIPCFGAVCZPX9BMVCMBXJXYDWJC9BDWZWUMFSCTMACUVHLJQZNAQX
YDBPAUXNZIYIAELVFTNRY9999
>
> and

> <ANY>
>
VTJGPEDIALNEUCFULFSIGQMSFBSDYUVKXVYYNCCGHAHPQTVFRYCDI9TORP
CBLZC9ULSWVGICTKGYD9999
>
CYMX9ELXYX9WCTHF9OAMBGEQDARMDGCRIBBEACV9ZLYVZNTUCEZXPHRKQ
NHQRHB99BSOXZTJLKQXB9999
>
GOSPZKDMTSBXYDX9PEZWKIPKZWHNZHNLLMJDNDZRTDE9ENTFJFTKVBQAAV
CKNPE9YPXQLWWVXNMOC9999.

> For a general case if we have a bundle with transaction hashes AAA, BBB, CCC and DDD these transactions should look this way:
>
> Tx AAA references BBB in “trunkTransaction” field
> Tx BBB references CCC in “trunkTransaction” field
> Tx CCC references DDD in “trunkTransaction” field
> Tx DDD references 999…999.

The piece of text is inconsistent. Based on the last paragraph, I would expect the "<ANY>" to be in the *last* in the list, not the first.

I am operating off the premise that if a bundle b consists of four transactions, b[0], b[1], b[2], and b[3], it is the case that b[i] includes the transaction hash of b[i+1] as its trunk transaction, and the last transaction b[3] includes a reference to *any* transaction with

index 0, and in particular a good default is 999...999 (which corresponds to the "<ANY>" designation you used above).

If this is true, then refer to new bundles
here: https://gist.github.com/narula/c48ff8d11aad5125ad1d359776cdc73d

And please let me know if they are valid bundles.

Letter #66
Author: Sergey Ivancheglo
Date: 11th of August

> The piece of text is inconsistent. Based on the last paragraph, I would expect the "<ANY>" to be in the *last* in the list, not the first.
It was the other way around. We had a chain of trunk transaction references, from it we derived the correct transaction hashes. Nothing was referencing b[0] hence it could have any hash.
> And please let me know if they are valid bundles.
The chain of trunk transaction references allows to construct the complete bundles now. I assume that the signatures are correct, they won't be accepted by Coordinator because it was upgraded but the old version would accept the signatures if the bundles went through anti-DoS routine.

Letter #67
Author: Sergey Ivancheglo
Date: 12th of August

Hi Neha,

The below is a quantitative analysis of the demonstrated collisions. It addresses the claims that second-preimages can be found with non-negligible chance and that full state collisions can be found. Precisely speaking, the odds of second-preimage collisions are shown to be below one of a million. Regarding the full state collisions, these collisions are actually just by-products of internal state collisions, we should, probably, agree on formal definitions. Anyway, please, reply if you disagree with anything from the analysis.

I don't provide a quantitative analysis of Curl-P randomness, it would take a lot of time to show that output changes caused by input changes have binomial distribution, before starting doing it I'd like to know when you are planning to publish Ethan's findings.

PS: You may find our analysis not sophisticated enough. This is because Curl-P is very simple, solving a system of equations, like Ethan did, is an overkill.

A typical bundle transferring value consists of 4 transactions:
Transaction #0: A deposit of AAA iotas to a recipient's address
Transaction #1: A withdrawal of BBB iotas from a sender's address
Transaction #2: A withdrawal of 0 iotas from the sender's address
Transaction #3: A deposit of (BBB-AAA) iotas to a sender's address (the change)
To assess the percentage of the bundles which can have a single trit changed
in "address" field without having the bundle hash changed as well, we should
pay attention to function F(A, B) which is the main part of the S-box.
The truth table of F() is:

| - | 0 | + | <- A

- | + | 0 | - |
---
0 | + | - | 0 |

+ | - | + | 0 |
---
^ \
| \
B F(A, B)

As we see, if A = -1 then flipping B between -1 and 0 doesn't change F().
The same is true if A = 1 and B = 0 / 1.
Another observation:
By changing a single trit (input) of Curl-P state in the beginning of a round
we can change not more than 2 trits (output) in the end of the round.

The changed output trits depend on the changed input trit and 2 other input trits which are kept unchanged.
Let's denote the first changed output trit as ALPHA and the second changed output trit as BETA. Let's denote the changed input trit as XXX and the remaining 2 input trits as C1 and C2 (they are constants).
We can write that:
ALPHA = F(XXX, C1)
and
BETA = F(C2, XXX)

By using exhaustive search we get that out of all possible combinations of input values (27) and all possible ways to change XXX (just 2, which are increment and decrement) we get 54 variants. Only 12 of them lead to a change of a single output trit, the remaining 42 variants change 2 output trits.
Note that none of the variants allows to keep all output trits unchanged thus showing that full state collision within a single transform() invocation is impossible for any number of rounds.
Out of those 12 variants none includes a combination with internal state having only zeros, this means that the address of Transaction #0 can't be changed. Transactions #1 and #2 can't be changed without failing the signature verification in the most cases. Transaction #4 is the only option left.

So, we have 243 trits of an address from Transaction #4. Our goal is to change a single trit in such a way that an avalanche of the propagated changes affects only first 243 trits of Curl-P state. This trick allows to generate a collision because the next absorbed 243 trits (value+timestamp+etc.) will overwrite the changes anyway. To simplify the analysis we'll do the following trade based on the observation that a single change in a typical input to Curl-P requires 11-15 rounds for a full diffusion to happen (full diffusion leads to ~2/3 of the output trits changed):
We reduce the number of rounds from 27 to 13 (=(11+15)/2) and allow a single output trit change to happen anywhere in the state.
We model our attack as a game consisting of 13 rounds. During a single round we win with probability 12/54. To win the game, all 13 rounds must be won. The probability to win the game by changing a single trit from 243 trits of an address is
(12/54)^13 or ~0.0000000032.
For a single address we can attempt 243 games. The probability of winning in at least 1 game in this case is
1 - ProbabilityOfLosingInAllGames
or
1 - (1 - (12 / 54) ^ 13) ^ 243
or
~0.00000078
according to www.wolframalpha.com.

The physical meaning of our analysis results is the following:
In a long run we need to process 1 million bundles (or more than 4 million transactions) to be able to conduct the attack successfully.

Sergey

Letter #68
Author: Sergey Ivancheglo
Date: 13th of August

Hi Ethan,

It's been a long time since we've heard anything from you. We thought you were busy with other projects so we analyzed your words related to EU-CMA security of IOTA signature scheme without distracting you. At this point we believe we found a mistake (in your chain of thoughts) which had led to the false claim of the EU-CMA security being broken.

In one of the letters you linked us to http://www.cs.tau.ac.il/~canetti/f08-materials/scribe8.pdf as your choice of EU-CMA definition. We were confused by the fact that you preferred such an oversimplified definition which barely has any use outside of teaching students the basics of Cryptography but later we realized that you provided that definition assuming that we didn't have cryptographers in our team (thank you for your attempts to make everything as clear to us as possible). We don't know what definition you prefer to use but it very likely has an equivalent to this fragment:

The verification algorithm is deterministic, that is, for any vk, m, s:
Prob[Ver(vk, m, s) = accept] ∈ {0, 1}
In other words, it is not possible that the verification algorithm accepts a signature in one run but
rejects it in another.

This is where your and our points of view diverge.

As we wrote, Satoshi's invention led to a paradigm shift in Cryptography (to some degree, some cryptographers still resist this change refusing to think outside of the box) and this shift forces us to critically review old definitions and to create new ones more suitable to our needs. If you remember, our team is not a big fan of "appeal to authority", still the following words from "Constructing Cryptographic Definitions" by Phillip Rogaway (http://web.cs.ucdavis.edu/~rogaway/papers/iscisc.pdf) should be repeated:
"I would emphasize that definitions are not written in stone. They emerge, change, and die out far more often than people imagine. They are part of a dialectic within a community."

This is about our very case.

The signature verification oracle in IOTA is not deterministic. While it doesn't reject signatures which were accepted, it can reject a signature at one moment and then accept it later. The oracle runs inside Coordinator, its reports are published on the tangle as milestones.

It's worth noting that the approach of relying on a public ledger for signature verification is not unique to IOTA. The idea was explored by several people, e.g. in "MAVE, NEW

LIGHTWEIGHT DIGITAL SIGNATURE PROTOCOLS FOR MASSIVE VERIFICATIONS” by SERGIO DEMIAN LERNER (https://bitslog.files.wordpress.com/2012/04/mave1.pdf) and “Fawkescoin, A cryptocurrency without public-key cryptography” by Joseph Bonneau and Andrew Miller (http://www.jbonneau.com/doc/BM14-SPW-fawkescoin.pdf). We are sure you won’t be questioning the expertise of the mentioned persons, so we ask you to provide a definition of EU-CMA security that could be applied to MAVE and Fawkescoin, after that we’ll use it to verify your claim of IOTA’s EU-CMA security being broken.

PS: If you find time for something outside of the main topic of these letters we would be happy to discuss implications of blockchains on old cryptographic algorithms like the one described in "Constructing digital signatures from one-way functions" by Leslie Lamport.

Sergey

Letter #69
Author: David Sønstebø
Date: 22nd of August

Hello all,

It has been 10 days without an update at this point, is this still being worked out or should the statements and claims be publicly retracted? We have been working around the clock to ensure 100% best practices, but so far what we have received is inconclusive and no clarification on the very bold statements made by Ethan.
Our cryptographers are starting to wonder what this is all about at this point, so it would be good to conclude this ASAP. We have a community of 100K+ people at this point and everyone will be eager to have these claims properly audited and verified/falsified ASAP.

Best,
David

Letter #70
Author: Neha Narula
Date: 22nd of August

Hi David,

We've had a few time-consuming things going on on our end. A response is forthcoming. Unfortunately it might not be on the time frame you'd like.

Best,
Neha

Letter #71
Author: Neha Narula
Date: 1st of September

Hi,

I'm not sure how to respond to the previous set of emails, I find them confusing. If you'd like a response to something specific, please simplify and clarify and I'll do my best! We do still maintain that EU-CMA security is broken.

We have produced a set of bundles which we believe could be used to steal funds, as follows. Could you verify that these bundles validate according to the state of IOTA before you hard forked?

Eve is sending the bundle to Alice to sign.

In STEAL_BUNDLE0:
- tx0 50000 spend
- tx1 -939211930 fund, alice's sig
- tx2 0, alice's sig
- tx3 100 to eve
- tx4 0
- tx5 939161830 to alice2
- tx6 0

In STEAL_BUNDLE1:
- tx0 50000 spend
- tx1 -939211930 fund, alice's sig
- tx2 0, alice's sig
- tx3 129140263 to eve
- tx4 0
- tx5 810021667 to alice2
- tx6 0

```
var STEAL_BUNDLE0 []string = []string{
        "999…<snip>…BKE",
        "NER…<snip>…ROK",
        "AZZ…<snip>…MTCO",
        "999…<snip>…BKE",
        "999…<snip>…BKE",
        "999…<snip>…CQR",
        "999…<snip>…BKE",
}
var STEAL_BUNDLE1 []string = []string{
        "999…<snip>…BKE",
        "NER…<snip>…ROK",
        "AZZ…<snip>…TCO",
        "999…<snip>…BKE",
        "999…<snip>…BKE",
```

```
        "999…<snip>…CQR",
        "999…<snip>…BKE",
}
```

Thanks,
Neha

Letter #72
Author: Sergey Ivancheglo
Date: 2nd of September

Hi,

> We have produced a set of bundles which we believe could be used to steal funds, as follows. Could you verify that these bundles validate according to the state of IOTA before you hard forked?

One of the bundles is not valid, but it shouldn’t be hard to fix it, just write the correct bundle hash into the corresponding field and redo the PoW.

We see you used tx4 to “decouple” probabilities of finding inner collisions for tx3 and tx5. Does this mean you will retract the claim of Curl-P not being pseudo-random?

> If you'd like a response to something specific, please simplify and clarify and I'll do my best!

Perhaps it will be easier to get the full overview if you can send over the draft you intend to publish. Seeing it in advance would also help to reduce the gap between your and our publications. Any ETA on your publication, by the way?

Sergey

Letter #73
Author: Neha Narula
Date: 6th of September

OK. I think I did that! Thank you for validating bundles over an email transport layer.

ETA is that we are ready to publish. Here is a copy of our vulnerability report. We would greatly appreciate it if you could let us know if there are any mistakes.

IOTA Vulnerability Report: Cryptanalysis of the Curl Hash Function Enabling Practical Signature Forgery Attacks on the IOTA Cryptocurrency

Authored by Ethan Heilman, Neha Narula, Thaddeus Dryja, Madars Virza

Summary: We present attacks on the cryptography used in the IOTA blockchain including under certain conditions the ability to forge signatures. We have developed practical attacks on IOTA's cryptographic hash function Curl, allowing us to quickly generate short colliding messages. These collisions work even for messages of the same length. Exploiting these weaknesses in Curl, we break [the EU-CMA security](http://www.cs.tau.ac.il/~canetti/f08-materials/scribe8.pdf) of the IOTA signature scheme. Finally we show that in a chosen message setting we can forge signatures of valid spending transactions (called bundles in IOTA). We present and demonstrate a practical attack whereby an attacker could exploit this vulnerability in IOTA to steal funds from an IOTA user. This report provides example demonstrations of these vulnerabilities but does not detail the exact cryptanalytic process to generate the collisions. A second publication will provide an in-depth study of our cryptanalysis of Curl.

Responsible Disclosure statement: This report is the product of a responsible disclosure process. Prior to publishing this report we disclosed these vulnerabilities to the IOTA developers. In response the IOTA developers have updated IOTA to no longer use the Curl hash function to hash transactions as part of the IOTA signing process. Curl is still used for other purposes in IOTA. We are now publishing our attacks since the IOTA developers have deployed their fixes. See our timeline of events below.

Our analysis is based only on publicly available information including the [IOTA open source repos on github](https://github.com/iotaledger), [the IOTA forum](https://forum.iota.org/) and email discussions with the IOTA developers during the disclosure process. At no time did we send any of these forged signatures to the IOTA network or interfere in the IOTA network in any way. We validated all of our attacks offline using publicly available libraries.

1. Systems Impacted:

* **The Cryptographic Hash Function Curl:** The cryptographic hash function Curl designed by the IOTA project has serious weaknesses. We have demonstrated practical attacks which break Curl's collision resistance for messages of different length and the same length. We also demonstrate additional non-random behavior between

certain classes of related messages, namely messages which are bit rotations of other messages. Curl should not be relied on for randomness or collision resistance. IOTA is still using Curl for transaction ID generation and for proof of work, we present no practical attacks against these uses. We provide examples of collisions and non-random behavior at the end of the report.

* **Prior versions of IOTA:** The signature scheme used by IOTA prior to recent updates relied on the collision resistance of Curl for hashing messages as part of the signature algorithm. The attacks we developed against Curl lead to practical signature forgery attacks against payments in IOTA. We provide an example forged signature on a valid IOTA payment (before the fix was deployed) at the of the report.

Mitigations: On Aug 7 2017 IOTA deployed [a hardfork](https://blog.iota.org/upgrades-updates-d12145e381eb) to their system to stop using Curl for signature message hashing. In order to perform the upgrade, trading was [halted on Bitfinex](https://www.bitfinex.com/posts/215) for [roughly 3 days](https://twitter.com/bitfinex/status/895915034102177792). All users who hold IOTA directly (not via an exchange) must [upgrade their wallets and addresses](https://blog.iota.org/new-gui-transition-phase-explained-b536ed80f412).

The signature forgery vulnerability was fixed in [IOTA Reference Implementation (IRI) version 1.3](https://github.com/iotaledger/iri/releases/tag/v1.3), [IOTA wallet version 2.4.0](https://github.com/iotaledger/wallet/releases/tag/v2.4.0).

2. Disclosure Timeline

* **July 14 2017:** We disclosed a weaknesses in the Curl hash function to the IOTA developers and informed them that we were making steady progress on additional attacks. Because of these weaknesses we recommended that they replace the Curl hash function with a recognized and publicly vetted hash function. This disclosure included example collisions on messages of different lengths and an example of non-pseudo-randomness.

* **July 22 2017:** We disclosed improved attacks on Curl and showed how these attacks could break the collision resistance of Curl even for messages of the same length. We also outlined how we could use these collisions to break the EU-CMA security of the IOTA signature scheme. This disclosure included an example collision we created via [differential cryptanalysis](https://en.wikipedia.org/wiki/Differential_cryptanalysis).

* **July 25 2017:** After continued discussion, the IOTA developers proposed a timeline for fixing these vulnerabilities. On Aug 5th Curl would be replaced with the [KECCAK hash function (also known as SHA-3)](https://en.wikipedia.org/wiki/SHA-3). Then on Aug 5th-10th users would move their tokens from Curl to KECCAK. On Aug 12th the IOTA devs would disclose the existence of the vulnerability.

* **Aug 7 2017:** On Aug 7th the IOTA developers [merged code](https://github.com/iotaledger/iri/commit/539e413352a77b1db2042f46887e41d558f575e5) which replaced Curl with a hash function they named [Kerl](https://github.com/iotaledger/kerl) which according to [the Kerl specification is a ternary variant of the KECCAK hash function](https://github.com/iotaledger/kerl/blob/master/IOTA-Kerl-spec.md). Around this

same time the IOTA developers posted a blog entry titled ["Upgrades & Updates"](https://blog.iota.org/upgrades-updates-d12145e381eb) which discussed the reasons for moving away from Curl stating:

>Creating a new cryptographic hash function is no trivial undertaking, even when it is being built on preexisting world class standards. “Don’t roll your own crypto” is a compulsory uttered mantra that serves as a good guiding principle for 99.9% of projects, but there are exceptions to the rule. When spearheading technology for a new paradigm this statement is no longer axiomatic. Progress must march on. Therefore audits, reviews and continued research on Curl has been a given from day 1. One of the cryptographers we reached out to months ago to review Curl has disclosed that he is worried there might be a potential vulnerability in Curl. We have since had our internal team, as well as other cryptographers review it and asked the disclosing party for more information. While the party that did the responsible disclosure has been quite forthcoming, there are still some of the last details to be discussed more thoroughly with the respective teams in order to reproduce the claims and verify if there was even any vulnerability.

>However, even though we have protection mechanisms in place that would render even most valid attacks useless in this ‘training wheel stage’ (due to the Coordinator and the higher-level protocol), as you are working on the cutting edge you have to take every precaution possible and always be on guard. Therefore we have made the simple decision to temporarily switch Curl with Keccak (SHA-3) for cryptographic signing in IOTA. - ["David Sønstebø - Upgrades & Updates"](https://blog.iota.org/upgrades-updates-d12145e381eb)

* **Aug 10 2017:** We sent the IOTA developers valid IOTA payments (bundles in IOTA terminology) with different output addresses which hash to the same value under Curl as used in IOTA signatures (prior to the Aug 7th fix). This showed that not only could we forge signatures for some messages, but that these messages could be valid payments in IOTA.

* **Sept 1 2017:** We sent the IOTA developers valid IOTA payments which pay out different amounts and hash to the same value under Curl as used in IOTA signatures. This showed that under certain circumstances an attacker could exploit this cryptologic vulnerability in Curl to steal funds (prior to the Aug 7th fix). In the example Alice signs a payment paying Eve 100 IOTA, and Eve can use the signature on this payment to authorize a payment where Eve receives 129140263 IOTA from Alice's funds.

3. IOTA Background
The IOTA network launched July 11, 2016 and its token was listed on Bitfinex on June 14, 2017. A fixed supply of 2.77 Billion MIOTA (currency units) was created when the network was launched. 1 MIOTA (mega/million IOTA) is equal to 1,000,000 IOTA. As of [YYY IOTA has a market capitalization of YYY](https://coinmarketcap.com/currencies/iota/historical-data/?start=20170819&end=20170820) making it the YYY most valuable blockchain based cryptocurrency by market cap.

IOTA as it is currently deployed has several uncommon design features and terminology:

* [IOTA is built on the concept of a tangle](https://iota.org/IOTA_Whitepaper.pdf) (known also as a DAGchain or Directed Acyclic Graph Blockchain) where instead of a single chain of blocks, transactions are linked together in a graph. A transaction is a simple object specifying an address, signature, value, tag, and a few other fields. A group of transactions which together specify a transfer is called a bundle; in a bundle, [transactions roughly correspond to Bitcoin inputs or outputs](https://forum.iota.org/t/what-are-the-4-contents-of-a-bundle/2211). Each transaction must include a small amount of proof of work, and point to two other transactions already in the Tangle.

* IOTA uses [balanced ternary](https://en.wikipedia.org/wiki/Balanced_ternary) (base 3) instead of binary (base 2). That is *trits* and *trytes* instead of *bits* and *bytes*. A tryte consists of three trits.

* IOTA currently relies on a trusted party called a coordinator [to approve and checkpoint state](http://www.tangleblog.com/2017/01/25/the-tech-behind-iota-explained/). This has led to concerns that [IOTA is centralized](https://medium.com/@ercwl/iota-is-centralized-6289246e7b4d). The IOTA developers argue [IOTA is not centralized and that this is a temporary measure "to provide 34% attack protection"](https://blog.iota.org/the-transparency-compendium-26aa5bb8e260). The code for the coordinator is not open source.

4. Practical Attacks Against the Cryptographic Hash Function Curl

The IOTA team designed their own cryptographic hash function called Curl. It is used for a number of purposes in IOTA including transaction address creation, message digest creation, Proof-of-Work (PoW) and hash-based signatures. Unlike most cryptographic hash functions, Curl operates on base-3 numbers called [“balanced ternary”](https://en.wikipedia.org/wiki/Balanced_ternary). Curl is built on the popular [sponge construction](https://en.wikipedia.org/wiki/Sponge_function).

Curl takes a message, breaks it into message blocks and then interatively copies each message block into the current state and runs Transform on the state it update the state.

```
Curl(message)
    # the state consists of 729 trits. It is initalized to all zero.
    state = [0]*729

    # The message is broken into message blocks of size 243
    MB_0, MB_1, ... MB_n = split(message)
    for MB_i in MB_0, MB_1, ... MB_n:
        # The current message block is copyed into the first 243 trits of the state
          state[0:243] = MB_i
          state = Transform(state)

    # The output is the first 243 trits of the state
    output = state[0:243]
    return output
```

The function Transform used by Curl is an unkeyed [permutation substitution network](https://en.wikipedia.org/wiki/Substitution-permutation_network).
```
Transform(state)
    for round in 27
        i = 0
          new_state = [0]*729

        for pos in 729
            i = j
            j += (364 if j < 365 else -365)

            x = state[i]; y = state[j]
              z = sbox[x, y]
              new_state[pos] = z

          state = new_state
    return new_state
```
The sbox takes two trits and returns a third trit.
```
      y:  -1, 0, 1
    x: -1 [1, 1, -1]
    x:  0 [0, -1, 1]
    x:  1 [-1, 0, 0]

```

A close inspection of the Curl source code revealed that Curl was vulnerable to a well known technique for breaking hash functions called [differential cryptanalysis](https://en.wikipedia.org/wiki/Differential_cryptanalysis). Using this observation, we were able write software that could quickly generate practical collisions for messages of the same lengths. Since these collisions fully collide the internal state of the hash function, a single collision enables us to generate an unbounded number of additional colliding messages. These collisions are for all rounds of Curl and can be generated in seconds on commodity hardware. The nature of our attack allows us great control over values of the colliding messages.

Thus, Curl as used in IOTA does not provide collision resistance. We have provided example collisions and non-pseudo random behavior at the end of this report. In the next section we will show how we exploit these collisions to break the EU-CMA security of IOTA's signature scheme.

5. Breaking the EU-CMA Security of the IOTA Signature Scheme

IOTA uses a signature scheme based on [Winternitz One-Time Signatures (WOTS)](https://cryptoservices.github.io/quantum/2015/12/04/one-time-signatures.html) with an important difference: IOTA operates on [the hashes of messages](https://github.com/iotaledger/iota.lib.java/blob/f43d606f041d1bf6eceb44d6758b710149b15d0a/src/main/java/jota/utils/Signing.java#L186) instead of operating directly on messages as is done WOTS. This is a critical difference since [WOTS only requires that the hash function is a One Way

Function](https://eprint.iacr.org/2011/191.pdf), whereas the IOTA signature scheme requires that the hash function must also be collision resistant. Important for our attack is that if two messages, msg1 and msg2, hash to the same output, a signature on msg1 will also verify as a signature on msg2.

```
IOTA_Sign(SK, msg):
    h_msg = CURL_Hash(msg)
    sig = WOTS_Sign(SK, h_msg)
    return sig
```

Before explaining more we need to provide some background on how the security of a digital signature scheme is defined. The standard security definition for a digital signature schemes is called Existential Unforgeability against a Chosen Message Attack or [EU-CMA](http://www.cs.tau.ac.il/~canetti/f08-materials/scribe8.pdf). Let's break down what this means:

* Existential Unforgeability (EU) means that the attacker can should not be able to forge signatures for any messages even if the messages are complete nonsense.
* A Chosen Message Attack (CMA) is an attack in which the attacker is given complete control to choose any messages on which to perform the attack.

Thus, EU-CMA security guarantees that even if the attacker chooses two messages msg1, msg2 and the signer signs msg1, the attacker should not be able to find a valid signature for msg2.

Breaking the EU-CMA security of IOTA's signature scheme:
In our attack a malicious user, Eve, tricks a user Alice by asking Alice to sign a message msg1 and then later produces a different message, msg2, which also verifies under that signature.

<pre>

(1). Eve uses our collision attack on Curl to chooses two messages, msg1, msg2 such that:
    CURL_Hash(msg1) = CURL_Hash(msg2) and msg1 != msg2

(2). Eve sends msg1 to Alice and asks Alice to sign it.

(3). Alice sends Eve a signature on msg1:
    sig1 = IOTA_Sign(SK,msg1)

(4). Eve produces a valid signature,message pair (sig1,msg2) where msg1 is a message which Alice has not signed.
    msg1 != msg2 AND IOTA_Sign(SK, msg1) == IOTA_Sign(SK, msg2)
</pre>

By definition Alice has broken the EU-CMA security of IOTA's signature scheme.

Forging Signatures on IOTA Payments:

IOTA uses a bundle to represent a transfer of value; a bundle consists of multiple transactions. These transactions specify the inputs and outputs of the transfer. We were able to produce two bundles which differ by only one or two trits but hash to the same value, and so have the same signature. We can demonstrate a practical attack where we cause an IOTA user to lose funds.

A transaction in IOTA has the following fields:

<pre>
SignatureMessageFragment,
Address,
Value,
Tag,
Timestamp,
CurrentIndex,
LastIndex,
BundleHash,
TrunkTransactionID,
BranchTransactionID,
Nonce

</pre>

A bundle consists of multiple transactions, containing credits to the receiving addresses, debits from the spending addresses, and extra empty transactions which hold parts of the signature. Tags are only constrained by length, which we use to set up our attacks. The total value across all transactions in a bundle must sum to zero, and every transaction must contain two pointers to other valid transactions in the IOTA tangle (the trunk and branch transactions). A transaction ID is constructed by hashing the concatenation of every field in a transaction. The convention is that within a bundle, each transaction points to the transaction after it as its trunk transaction, and the last transaction in the bundle points to some other first transaction in a different valid bundle. The nonce must be calculated to provide sufficient proof of work for the transaction to get accepted into the tangle -- at the time of writing, this required that the transaction hash (including the nonce) ends with at least fifteen zero trits.

A user signs the hash of a bundle. A BundleHash, unlike a transaction hash, does not use all the fields in each transaction in the bundle. Instead, it is constructed by hashing the concatenation of only the following fields:

<pre>
Address,
Value,
Tag,
Timestamp,
CurrentIndex,
LastIndex
</pre>

of all transactions in the bundle.

Waste money attack:

We use our method for generating collisions to construct two valid IOTA bundles which collide on one trit in the Address fields.

<pre>
Alice wants to pay Eve:

(1). Eve creates two colliding bundles; bundle1 and bundle2. Bundle1 spends some of Alice’s funds and pays to Addr1. Bundle2 also spends some of Alice’s funds but differs in that it pays out to a slightly different address, addr2.

(2). Eve asks Alice to pay Eve by signing bundle1, that is paying out to addr1.

(3). Alice broadcasts the signature and bundle1.

(4). Eve takes the signature off of bundle1 and uses it on bundle2 which she then broadcasts and ensures is accepted by the network. To do this she can use eclipse attacks against Alice, have better network propagation, or use proof of work to rapidly confirm bundle2 over bundle1.

(5). Eve then tells Alice that Alice paid to the wrong addr and shows that Alice signed a transaction spending out to addr2 not addr1 as Eve asked. Eve then asks to be paid again.
</pre>

Similar attacks, based on [transaction malleability have been claimed to be used in Bitcoin to trick cryptocurrency exchanges into spending payments multiple times](https://www.coindesk.com/study-finds-mt-gox-lost-386-bitcoins-due-transaction-malleability/). This particular attack here is both more serious than transaction malleability attacks in Bitcoin since it exploits a deep cryptographic flaw in IOTA allowing signature forgeries, whereas transaction malleability does not allow signature forgeries.

Steal money attack:

We use our method for generating collisions to construct two valid IOTA bundles which collide twice on different trits. If Alice signs a bundle, bundle1, which pays Eve 100 IOTA from Alice's funds. Eve can use the signature on bundle1 as a valid signature on bundle2 which pays Eve 129140263 IOTA from Alice's funds.

<pre>
Alice wants to pay Eve:

(1). Eve creates two colliding bundles; bundle1 and bundle2. Bundle1 spends 939211930 IOTA from Alice’s address and pays 100 IOTA to Eve and 939211830 IOTA in change back to Alice. Bundle2 also spends some of Alice’s funds but differs in that it instead pays 129140263 IOTA to Eve and only 810071667 IOTA in change back to Alice.

(2). Eve asks Alice to pay Eve by signing bundle1, that is paying 100 to Eve and send the change 939211830 back to Alice.

(3). Alice broadcasts the signature and bundle1.

(4). Eve takes the signature off of bundle1 and uses it on bundle2 which she then broadcasts and ensures is accepted by the network. To do this she can use eclipse attacks against Alice, have better network propagation, or use proof of work to rapidly confirm bundle2 over bundle1.

(5). When Bundle2 is confirmed, Eve will have stolen 129140163 IOTA from Alice.
</pre>

This is an extremely serious attack as a signature attesting that "Alice pays Eve 100 IOTA", can also attest to "Alice pays Eve 129140263 IOTA" without Alice ever agreeing to pay Eve 129140263 IOTA. Signatures using Curl can no longer be trusted to authorize transactions.

6. Proof-of-Concept - Curl Attack Examples
We provide examples of colliding messages for the Curl hash function and non-randomness. Our first two examples can be replicated using [a short python proof-of-concept program we have made available](https://github.com/mit-dci/tangled-curl/blob/master/curlexamples.py). We also provide longer colliding messages which can be replicated using the ccurl-digest commandline tool.

Short colliding messages:

Two messages which collide and differ by a single position. Our python code for replicating this can be found here.

```
msg1 =
"RETHT9ES9HRCUITBHVCUHOBPUUUHT9PHLUNWRWGKBKF9YUMDWRXTRVGZ
HFZEHGATZXZAUPGVEKNMQXFVRXHF9QJQHUTILIPIXUYRVSJEIOJDRIUVWMUA
BSIKIBAKENE9KVFJUEQUHFRVGELFGJIDXQARWH99XTORHXRETHT9ES9HRCUI
TBHVCUHOBPUUUHT9PHLUNWRWGKBKF9YUMDWRXTRVGZHFZEHGATZXZAUP
GVEKNMQXFVR"
msg2 =
"RETHT9ES9HRCUITBHVCUHOBPUUUHT9PHLUNWRWGKBKF9YUMDWRXTRVGZ
HFZEHGATZXZAUPGVEKNMQXFVRXHF9QJQHUTILIPIXUYRVSJEIPJDRIUVWMUA
BSIKIBAKENE9KVFJUEQUHFRVGELFGJIDXQARWH99XTORHXRETHT9ES9HRCUI
TBHVCUHOBPUUUHT9PHLUNWRWGKBKF9YUMDWRXTRVGZHFZEHGATZXZAUP
GVEKNMQXFVR"

>hash1 =  H(msg1)
GIUNBQRBI9RJQPNDVSSMUFMTLAKWTGYDMGBUYZAJNOJSKXWTYBV9QO9LB
AIEUANAXAIUTHKMNGRZKLSZN

>hash2 =  H(msg2)
GIUNBQRBI9RJQPNDVSSMUFMTLAKWTGYDMG~~~~~~~~BUYZAJNOJSKXWTYB
V9QO9LBAIEUANAXAIUTHKMNGRZKLSZN

>print hash1 == hash2, msg1 == msg2
True False
```

Non-randomness:

Two messages which are bit rotations of each other when hashed result in outputs which are also bit rotations of each other. For example we choose the string KENNYLOGGINS.

```
msg1 =
b'9999KENNYLOGGINS99999999999999999999999999999999999999999999999999999999999999'
msg2 =
b'9KENNYLOGGINS99999999999999999999999999999999999999999999999999999999999999999999'

hash1 =  H(msg1)
OBIVYFRBFRPRVNYMJFUGYYYFHREPKJWGLWCUSKDBQVOZSFYOPPOYGFTPQQA9ZMSYE99CMCYEELGEFRDTU

hash2 =  H(msg2)
VYFRBFRPRVNYMJFUGYYYFHREPKJWGLWCUSKDBQVOZSFYOPPOYGFTPQQA9ZMSYE99CMCYEELGEFRDTUXMW

```

Notice that the hash2 is exactly the same as hash1 but shifted by three characters: hash1 = OBI**VYFRBFRPRV**..., hash2 = **VYFRBFRPRV**...

Long colliding messages: Since the [ccurl digest commandline tool](https://github.com/iotaledger/ccurl) made availaible by IOTA only works for msgs of exactly 2673 Trytes long, we provide colliding messages of length 2673. We used the lyrics to the [80's hit single "push it to the limit"](https://en.wikipedia.org/wiki/Scarface_(Push_It_to_the_Limit)in the colliding messages to demonstrate that we fully collide the internal state of curl and thus giving us arbitrary control over most of the message. We note that this internal state collision enables us to instantly create an unbounded number of other colliding messages.

```
$ ./ccurl-digest RET…<snip>…ES9

BUWSNIAGAYCVUPGUJMWIIYHR9DQAPVHRKHXPJKB9BTCKPUXTFSNPXIBHCIIYCJRAQJOGHXEYLXSJURUFS

$ ./ccurl-digest RET…<snip>…ES9

BUWSNIAGAYCVUPGUJMWIIYHR9DQAPVHRKHXPJKB9BTCKPUXTFSNPXIBHCIIYCJRAQJOGHXEYLXSJURUFS
```

Other Collisions: We also note that there are several trivial methods to generate collisions for different message lengths in Curl, but these do not appear to impact the

security of IOTA. These can also be found in [our python example code](https://github.com/mit-dci/tangled-curl/blob/master/curlexamples.py).

7. Proof-of-Concept - IOTA Signature Forgery Examples
We provide two examples of a signature forgery attack we executed using valid IOTA bundles. We provide instructions to replicate our signature forgery attacks [here](https://github.com/mit-dci/tangled-curl). Colliding bundles given below.

Waste Money Attack

These two bundles demonstrate the signature forgery "waste money" attack described earlier in the report. A signature on BURN_BUNDLE1 is also a valid signature for BURN_BUNDLE2.

```
var BURN_BUNDLE1 []string = []string{
        "999…<snip>…BKE",
        "T9E…<snip>…ROK",
        "AZZ…<snip>…TCO",
        "999…<snip>…CQR",
}

var BURN_BUNDLE2 []string = []string{
        "999…<snip>…BKE",
        "T9E…<snip>…ROK",
        "AZZ…<snip>…TCO",
        "999…<snip>…CQR",
}
```

Steal Money Attack

These two bundles demonstrate the signature forgery "steal money" attack described earlier in the report. A signature on STEAL_BUNDLE1 is also a valid signature for STEAL_BUNDLE2. We also provide the bundle structure below.

<pre>
In STEAL_BUNDLE1:
- tx0 50000 spend to Alice
- tx1 -939211930 fund, Alice's sig
- tx2 0, Alice's sig
- tx3 100 to Eve
- tx4 0
- tx5 939161830 to Alice
- tx6 0
</pre>

<pre>
In STEAL_BUNDLE2:
- tx0 50000 spend to Alice
- tx1 -939211930 fund, Alice's sig
- tx2 0, Alice's sig

- tx3 129140263 to Eve
- tx4 0
- tx5 810021667 to Alice
- tx6 0
</pre>

```

var STEAL_BUNDLE1 []string = []string{
        "999…<snip>…BKE",
        "NER…<snip>…ROK",
        "AZZ…<snip>…TCO",
        "999…<snip>…BKE",
        "999…<snip>…BKE",
        "999…<snip>…CQR",
        "999…<snip>…BKE",
}

var STEAL_BUNDLE2 []string = []string{
        "999…<snip>…BKE",
        "NER…<snip>…ROK",
        "AZZ…<snip>…TCO",
        "999…<snip>…BKE",
        "999…<snip>…BKE",
        "999…<snip>…CQR",
        "999…<snip>…BKE",
}
```

Letter #74
Author: Sergey Ivancheglo
Date: 6th of September

Hi,

There are some corrections and suggestions to your paper:
> We have developed practical attacks on IOTA's cryptographic hash function Curl
"Curl" should be replaced with "Curl-P". "Cryptographic" should be removed because Curl-P was designed to allow practical collisions.
> Exploiting these weaknesses in Curl, we break [the EU-CMA security](http://www.cs.tau.ac.il/~canetti/f08-materials/scribe8.pdf) of the IOTA signature scheme.
A better definition is required, the one you use allows us to break EU-CMA security after just 10 queries to the signing oracle thus making your "attack" lose any value. The definition also doesn't take into account existence of distributed/decentralized ledgers.
> We present and demonstrate a practical attack whereby an attacker could exploit this vulnerability in IOTA to steal funds from an IOTA user.
This is a wrong claim because the attacker can't make a user sign an arbitrary hash.
> This report is the product of a responsible disclosure process.
This reminded me a question I wanted to ask some time ago: How did it happen that a RESPONSIBLE disclosure led to 10s people knowing the details which were supposed to be kept in secret?
> The cryptographic hash function Curl designed by the IOTA project has serious weaknesses.
Should be changed to "The hash function Curl-P designed by the IOTA project has properties used as protection against scam copycats."
> We have demonstrated practical attacks which break Curl's collision resistance for messages of different length and the same length.
"Different length" part should be removed, that was incorrect usage of Curl-P caused by absence of complete documentation.
> We also demonstrate additional non-random behavior between certain classes of related messages, namely messages which are bit rotations of other messages.
This part should be removed completely, that was incorrect usage of Curl-P caused by absence of complete documentation.
> IOTA is still using Curl for transaction ID generation and for proof of work, we present no practical attacks against these uses.
Are you planning to provide them in the nearest future? We'd like to reference them to show that illegal copies of IOTA software are vulnerable to attacks.
> The signature scheme used by IOTA prior to recent updates relied on the collision resistance of Curl for hashing messages as part of the signature algorithm.
This is a wrong statement, some our letters weren't probably received by your team, could we do some kind of a comparison to find the discrepancies?
> The attacks we developed against Curl lead to practical signature forgery attacks against payments in IOTA.
"Practical" should be removed, probability of success should be mentioned. If I recall correctly it's something close to "one in a trillion".
> We provide an example forged signature on a valid IOTA payment (before the fix was deployed) at the of the report.

That example lacks explanation of the networking part of the attack scenario.
> In order to perform the upgrade, trading was [halted on Bitfinex](https://www.bitfinex.com/posts/215) for [roughly 3 days](https://twitter.com/bitfinex/status/895915034102177792).
“Trading” should be replaced with “deposits/withdrawals”.
> The signature forgery vulnerability was fixed in [IOTA Reference Implementation (IRI) version 1.3](https://github.com/iotaledger/iri/releases/tag/v1.3).
That upgrade was about other things. The migration to Kerl was included only because it was planned more than a year ago that we would upgrade it if a cryptographer found signs of the copy-protection mechanism before final Curl is ready.
> This disclosure included example collisions on messages of different lengths and an example of non-pseudo-randomness.
Should be removed completely because, as we already wrote, the both examples were based on an incorrect usage of Curl-P.
> We also outlined how we could use these collisions to break the EU-CMA security of the IOTA signature scheme.
You used an incorrect definition of the EU-CMA security, this should, probably, be emphasized.
> We sent the IOTA developers valid IOTA payments (bundles in IOTA terminology) with different output addresses which hash to the same value under Curl as used in IOTA signatures (prior to the Aug 7th fix). This showed that not only could we forge signatures for some messages, but that these messages could be valid payments in IOTA.
The payment was valid only syntaxically, without the networking part of the attack scenario we can’t say that payment was valid.
> This showed that under certain circumstances an attacker could exploit this cryptologic vulnerability in Curl to steal funds (prior to the Aug 7th fix).
Should be mentioned that probability of “certain circumstances” is negligible.

The above covers only the first 2 sections. Let’s first come to consensus on them and then move to the remaining part.
PS: I know that David and Dominik also have some commentary that they will include tomorrow, for example, the timeline regarding Ethan being approached by IOTA team in May to do precisely this but declined due to working on Paragon Foundation. For us it is very important that everything is transparent and correct in this.

Sergey

Letter #75
Author: Neha Narula
Date: 6th of August

Hi! A quick response:

On Wed, Sep 6, 2017 at 3:59 PM, Come-from-Beyond <come-from-beyond@mail.ru> wrote:
Hi,

There are some corrections and suggestions to your paper:
> We have developed practical attacks on IOTA's cryptographic hash function Curl
"Curl" should be replaced with "Curl-P". "Cryptographic" should be removed because Curl-P was designed to allow practical collisions.

You refer to it as "Curl" and as a cryptographic hash function in your own post:

https://blog.iota.org/upgrades-updates-d12145e381eb

> Exploiting these weaknesses in Curl, we break [the EU-CMA security](http://www.cs.tau.ac.il/~canetti/f08-materials/scribe8.pdf) of the IOTA signature scheme.
A better definition is required, the one you use allows us to break EU-CMA security after just 10 queries to the signing oracle thus making your "attack" lose any value. The definition also doesn't take into account existence of distributed/decentralized ledgers.

I don't understand this statement.

> We present and demonstrate a practical attack whereby an attacker could exploit this vulnerability in IOTA to steal funds from an IOTA user.
This is a wrong claim because the attacker can't make a user sign an arbitrary hash.

I don't understand this statement.

> This report is the product of a responsible disclosure process.
This reminded me a question I wanted to ask some time ago: How did it happen that a RESPONSIBLE disclosure led to 10s people knowing the details which were supposed to be kept in secret?

Who are the 10s of people on our side? Our working group is the four people listed on this report. In the past week, we have shared drafts of the report to get feedback, as you have completed your updates. Everyone we shared with is under embargo.

> The cryptographic hash function Curl designed by the IOTA project has serious weaknesses.
Should be changed to "The hash function Curl-P designed by the IOTA project has properties used as protection against scam copycats."

I do not agree with this statement. Curl was used for signatures.

> We have demonstrated practical attacks which break Curl's collision resistance for messages of different length and the same length.
“Different length” part should be removed, that was incorrect usage of Curl-P caused by absence of complete documentation.

We were using Curl as a hash function.

> We also demonstrate additional non-random behavior between certain classes of related messages, namely messages which are bit rotations of other messages.
This part should be removed completely, that was incorrect usage of Curl-P caused by absence of complete documentation.

We were using Curl as a hash function.

> IOTA is still using Curl for transaction ID generation and for proof of work, we present no practical attacks against these uses.
Are you planning to provide them in the nearest future? We’d like to reference them to show that illegal copies of IOTA software are vulnerable to attacks.
No plans at the moment. I do not know what "illegal copies of IOTA" are.

> The signature scheme used by IOTA prior to recent updates relied on the collision resistance of Curl for hashing messages as part of the signature algorithm.
This is a wrong statement, some our letters weren’t probably received by your team, could we do some kind of a comparison to find the discrepancies?
What is wrong about this?

> The attacks we developed against Curl lead to practical signature forgery attacks against payments in IOTA.
“Practical” should be removed, probability of success should be mentioned. If I recall correctly it’s something close to “one in a trillion”.
We were able to create collisions within a few minutes on commodity hardware. This is "practical". We will make it clear that is what we mean when we say "practical".

> We provide an example forged signature on a valid IOTA payment (before the fix was deployed) at the of the report.
That example lacks explanation of the networking part of the attack scenario.
This example is independent of networking. It is simply two colliding bundles.

> In order to perform the upgrade, trading was [halted on Bitfinex](https://www.bitfinex.com/posts/215) for [roughly 3 days](https://twitter.com/bitfinex/status/895915034102177792).
“Trading” should be replaced with “deposits/withdrawals”.

I have no issue with this change.

> The signature forgery vulnerability was fixed in [IOTA Reference Implementation (IRI) version 1.3](https://github.com/iotaledger/iri/releases/tag/v1.3).
That upgrade was about other things. The migration to Kerl was included only because it was planned more than a year ago that we would upgrade it if a cryptographer found signs of the copy-protection mechanism before final Curl is ready.

The statement still stands, the specific attack we found was fixed in that update.

> This disclosure included example collisions on messages of different lengths and an example of non-pseudo-randomness.
Should be removed completely because, as we already wrote, the both examples were based on an incorrect usage of Curl-P.
We think the examples are useful.

> We also outlined how we could use these collisions to break the EU-CMA security of the IOTA signature scheme.
You used an incorrect definition of the EU-CMA security, this should, probably, be emphasized.

We do not agree with this.

> We sent the IOTA developers valid IOTA payments (bundles in IOTA terminology) with different output addresses which hash to the same value under Curl as used in IOTA signatures (prior to the Aug 7th fix). This showed that not only could we forge signatures for some messages, but that these messages could be valid payments in IOTA.
The payment was valid only syntaxically, without the networking part of the attack scenario we can't say that payment was valid.
I will try to edit this to make it very clear that we do not know exactly what would have happened had we submitted the transactions to the IOTA network, and we did not do so.

> This showed that under certain circumstances an attacker could exploit this cryptologic vulnerability in Curl to steal funds (prior to the Aug 7th fix).
Should be mentioned that probability of "certain circumstances" is negligible.
We do not agree with this. We think the probability is high enough for concern.

Letter #76
Author: David Sønstebø
Date: 7th of September

Hey Neha,

I was preparing a thorough response to your publication, then something almost incomprehensible occurred.
We are beyond baffled and frankly shocked at the moment. We were just reached out to by a CoinDesk journalist that Ethan contacted in an attempt to rush out this publication. This may be the biggest scandal I have ever heard of from what has been portrayed as a professional 'responsible disclosure'. Ethan is clearly in complete conflict of interest and pushing this for his own gain, this is no longer about academic merits, but a desperate attempt by Ethan to make money. We will use all resources to elucidate this as publicly as possible if Ethan does not effective immediately contact all the people he has been spreading this premature story to and retract all his statements.

Letter #77
Author: Neha Narula
Date: 7th of September

Hi,

The responsible disclosure time period is over; you fixed the vulnerability we found and deployed the fix. Our original agreement specified that we were bound until August 12th. It is quite well past that! But it is very important to us to hear what you think might be mistakes in the report, so we can fix them.

I'm afraid I don't agree with you on the topic of conflict of interests. At any rate, I'm the one who first contacted a journalist on this topic, so you should direct your ire towards me, not Ethan.

Thanks,
Neha

Letter #78
Author: David Sønstebø
Date: 7th of September

Neha, are you sober?

The repeated bugs in your code lead to weeks of postponements, and you still have not answered even half of our questions. This is the most unprofessional behavior I have ever witnessed by an 'academic'. Who is your supervisor?

Letter #79
Author: Neha Narula
Date: 7th of September

Hi,

I think these personal attacks are unprofessional and unwarranted. I'm going to stop responding now.

Thanks,
Neha

Letter #80
Author: David Sønstebø
Date: 7th of September

Neha, you rushed to the press with a preprint, as per your last communication with Sergey just an hour ago there is still a ton of unresolved issues. What kind of academic rushes to the press before peer review?

Letter #81
Author: Neha Narula
Date: 7th of September

Hi IOTA team,

We have been in working with you for over a month now developing this and answering your questions, and you said that you deployed this fix a month ago. We will take Sergey's comments under advisement. Tell us of any other factual issues you have with the report, and we will take those under advisement as well. We will be publishing tomorrow.

Thanks,
Neh

Letter #82
Author: Sergey Ivancheglo
Date: 7th of September

Hi Neha,

I think if your team fixes all issues in Ethan's findings there will be nothing to publish left. 1 day is not enough for you to extend the "attacks" to something possible in practice, so let's just continue our dispute in public.

I'm still expecting to get an explanation of how a RESPONSIBLE disclosure led to numerous people outside of our teams knowing the details before the update was even scheduled. This is especially important taking into account your own words:

> On August 5th, you will need to inform users to upgrade and change addresses. How do you plan on messaging this? At this point people will start looking for vulnerabilities.

Personally, I just can't get if you were knowing from the very beginning that the "vulnerability" found by Ethan wasn't critical or that your disclosure wasn't actually responsible given how many independent people, most of whom are not cryptographers or security researchers, has reached out to us about it (and that was only a part of those who had known about Ethan's findings).

Sergey

Letter #83
Author: Sergey Ivancheglo
Date: 21st of October

Hi Ethan,

I can't get a single reply from you, looks like you put me on ignore on Twitter. I don't blame you, sometimes I'm pretty annoying, I spend too much time with computers and lack some skills required for proper interaction with humans. I'm writing this letter to inform you that I'm going to contact Boston University administration to make some things (related to your report on Curl-P) clear. I'm doing it via a lawyer, not personally. I'm informing you now because I like transparency and feel that I shouldn't do that behind your back.

No need to reply to this letter if you don't want to, of course.

Best regards,
Sergey