A Tangled Curl

How we forged signatures in IOTA

Speaker:
Neha Narula (MIT Media Lab) **@neha**

Based on research performed with:
Ethan Heilman (Boston University, Commonwealth Crypto, advisor@DAGLabs),
Garrett Tanzer (Harvard University), **James Lovejoy** (MIT Media Lab, Vertcoin),
Michael Colavita (Harvard University), **Madars Virza** (MIT Media Lab, Zcash), **Tadge Dryja** (MIT Media Lab)

1 billion dollar marketcap

Custom hash function called **Curl**

IOTA Background: Terminology

	Bitcoin	**IOTA**
Payment	Transaction	Bundle
Currency 	1 Bitcoin ~ $3.6K	1M IOTA ~ $0.32

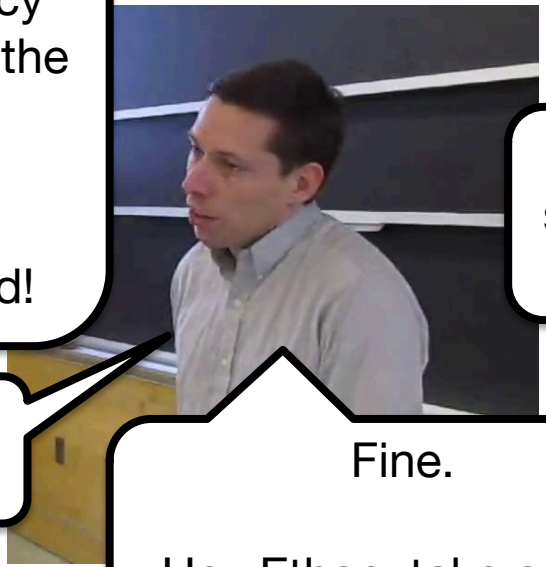
IOTA Background: Terminology

	Bitcoin	**IOTA**
Payment	Transaction	Bundle
Currency 	1 Bitcoin ~ $3.6K	1M IOTA ~ $0.32
Representation	Bits (0, 1) bytes (8 bits)	Trits (-1, 0, 1) trytes (3 trits)

Why did we look at IOTA?

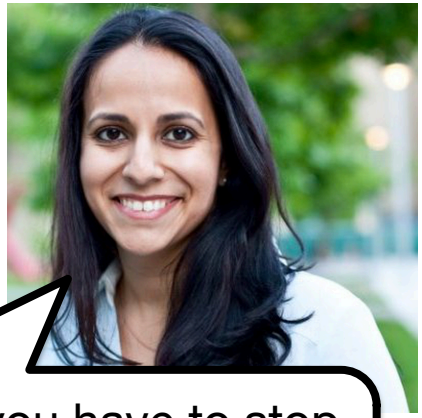
New cryptocurrency that solves all the problems!
Scalable!
No fees!
Decentralized!

No.



Fine.

Hey Ethan, take a look at this hash function…



Tadge, you have to stop saying everything sucks. Prove it.

There goes my weekend!

What is our attack?

- Bob signs a payment where he gets $2M and Eve gets almost nothing

- Eve forges Bob’s signature and instead sends a payment where she gets $2M and Bob gets almost nothing

- Chosen message setting: Eve gets to create the payment Bob signs

A note on impact and disclosure

- The signature forgery attacks presented here were **disclosed** to the IOTA developers
- The IOTA developers **deployed mitigations** for them
- The signature forgery attacks **no longer impact** IOTA's security

We never interfered with or sent anything to the IOTA network

In this talk…

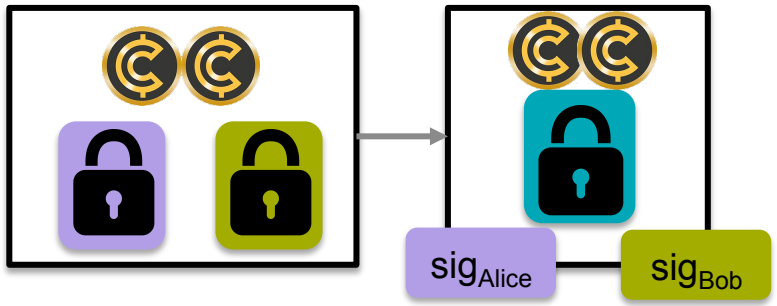
- **An attack on IOTA's multisig**
- Breaking the Curl-P-27 hash function
- Discussion

What is Multisig?

“Two-person” rule for nuclear launch

Multisig payments

A valid payment requires **k**-of-**n** signatures. Example 2-of-2:



Why multisig? Added security.

- Attacker has to compromise **both** keys
- Can store keys in isolated locations (cold storage)
- Used by many exchanges

IOTA Background: Signatures

<u>IOTA's signature scheme:</u>

- IOTA builds on Winternitz One-Time Signatures (WOTS)
- IOTA modifies WOTS

  ...to hash messages with Curl-P-27 prior to WOTS

```
IOTA_Sign(sk, m):
    h_m = Curl-P-27(m)
    sig = WOTS_Sign(sk, h_m)
    return sig
```

IOTA Background: Signatures

IOTA's si

- IOTA
- IOTA

...to h

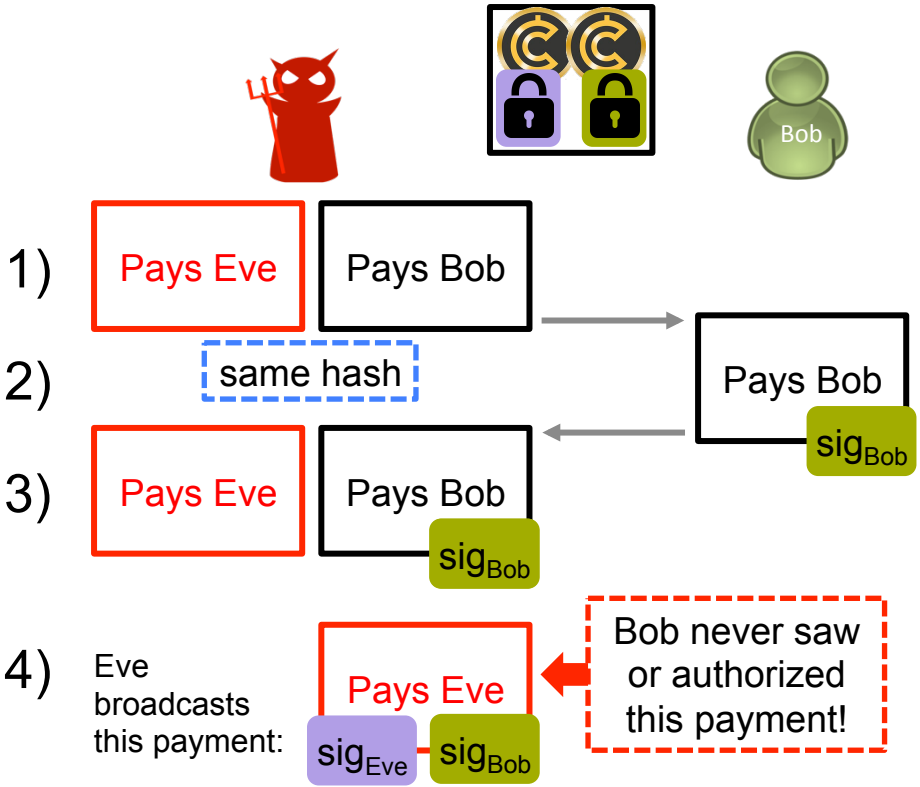
WOTS)

TS

The signature scheme details don't matter because in IOTA, payments are **hashed** before they are signed

If you can break the hash function, you can forge signatures!

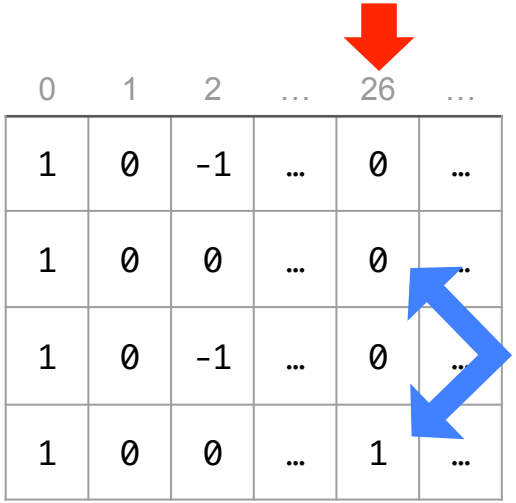
Exploiting colliding bundles: Unauthorized payments



1. Eve creates **two** special bundles which have the **same** hash
2. Eve asks Bob to sign the bundle paying him
3. Eve **copies** Bob's signature from the benign bundle to the evil bundle
4. Eve signs and broadcasts the evil bundle

Placing collisions to pay different amounts

Payee	Value
Alice:	100
Eve:	**1**
Carol:	100
Bob:	**2541865828330**

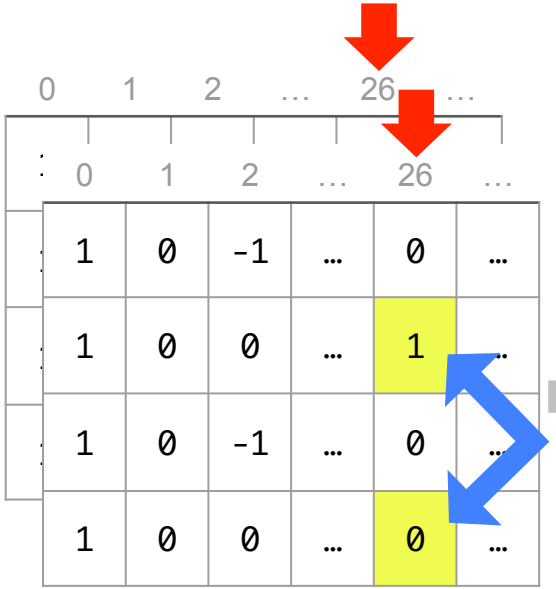


- Target Value fields for differing trits
- Create two colliding bundles which differ in 26th trit of two message blocks

Placing collisions to pay different amounts

Payee	Value
Alice:	100
Eve:	**1**
Carol:	100
Bob:	**2541865828330**

Bundle Bob sees



Payee	Value
Alice:	100
Eve:	**2541865828330**
Carol:	100
Bob:	**1**

Bundle Eve broadcasts

- Target Value fields for differing trits
- Create two colliding bundles which differ in 26th trit of two message blocks
- **Limitations: Can only play this trick in specific places**

In this talk…

- An attack on IOTA’s multisig
- **Breaking the Curl-P-27 hash function**
- Discussion

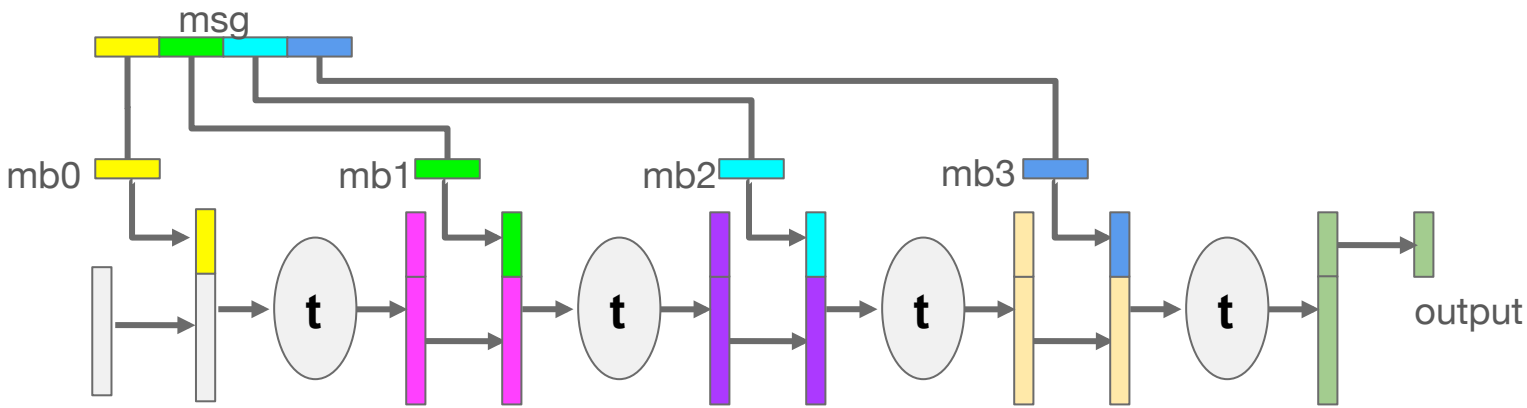
Curl-P-27: A Cryptographic Hash Function

To forge signatures we need to find
colliding msgs for Curl-P-27:

Curl-P-27(`-1011010...-1`) == Curl-P-27(`01000100...0`)

Curl-P-27 uses a Sponge-like Construction

Curl-P-27 is built on the sponge construction



Security depends on the transform function **t**

Curl-P-27: Transformation function is very simple

The transformation function in Curl-P-27 is just
the repeated application of a permutation + a simple S-Box

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

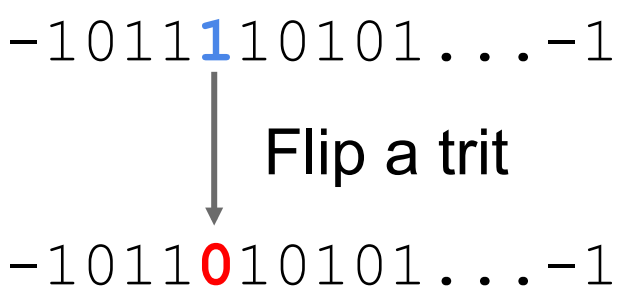
AES S-Box

	-1	0	1
-1	1	1	-1
0	0	-1	1
1	-1	0	0

Curl-P-27 S-Box

Curl-P-27: Reducing collision resistance

Choose a random bundle



If we flip the 26th trit the prob. of a collision is:

$>1/(2^{42.40})$

If we are clever about choosing the message this increases to

$>1/2^{22.87}$ = **1 out of 7.6 million**

In cryptographic terms this is **23-bit collision resistance**

Curl-P-27: Transformation function is very simple

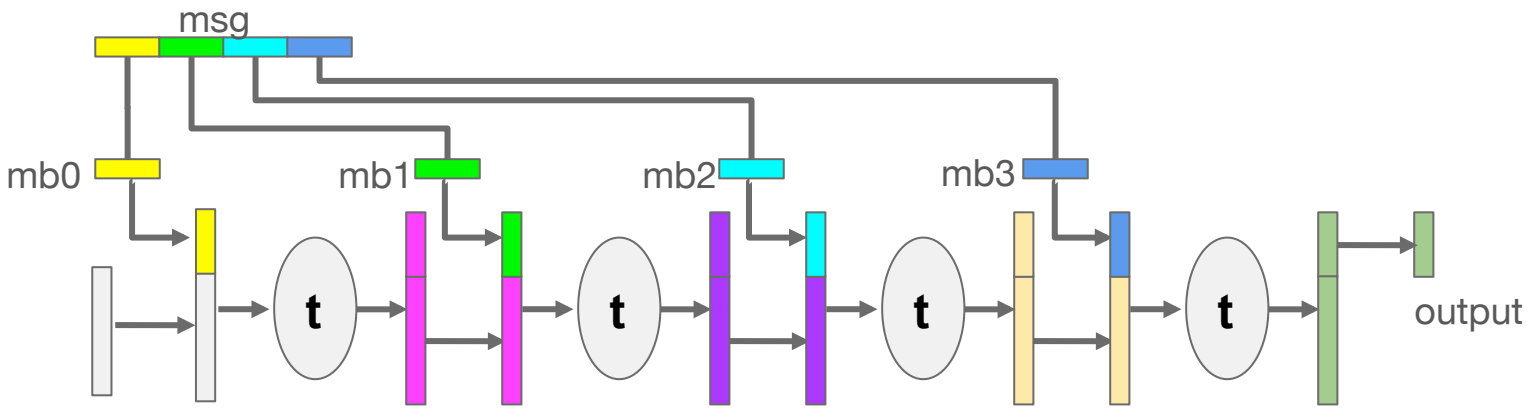
As the likelihood of a collision is at least 1 out of 7.6 million we need to try many messages (bundles) before we are successful

address	tag		value
DKSDJFLS...R	99999…JKA99		22000000...
QWEWEABZ...9	99999…LK999		00000010...
ABEPCMQQ...Z	99999…VBN99		00050000...

We can change the 81-trit tag field in IOTA bundles
Tags have no impact on transaction validity

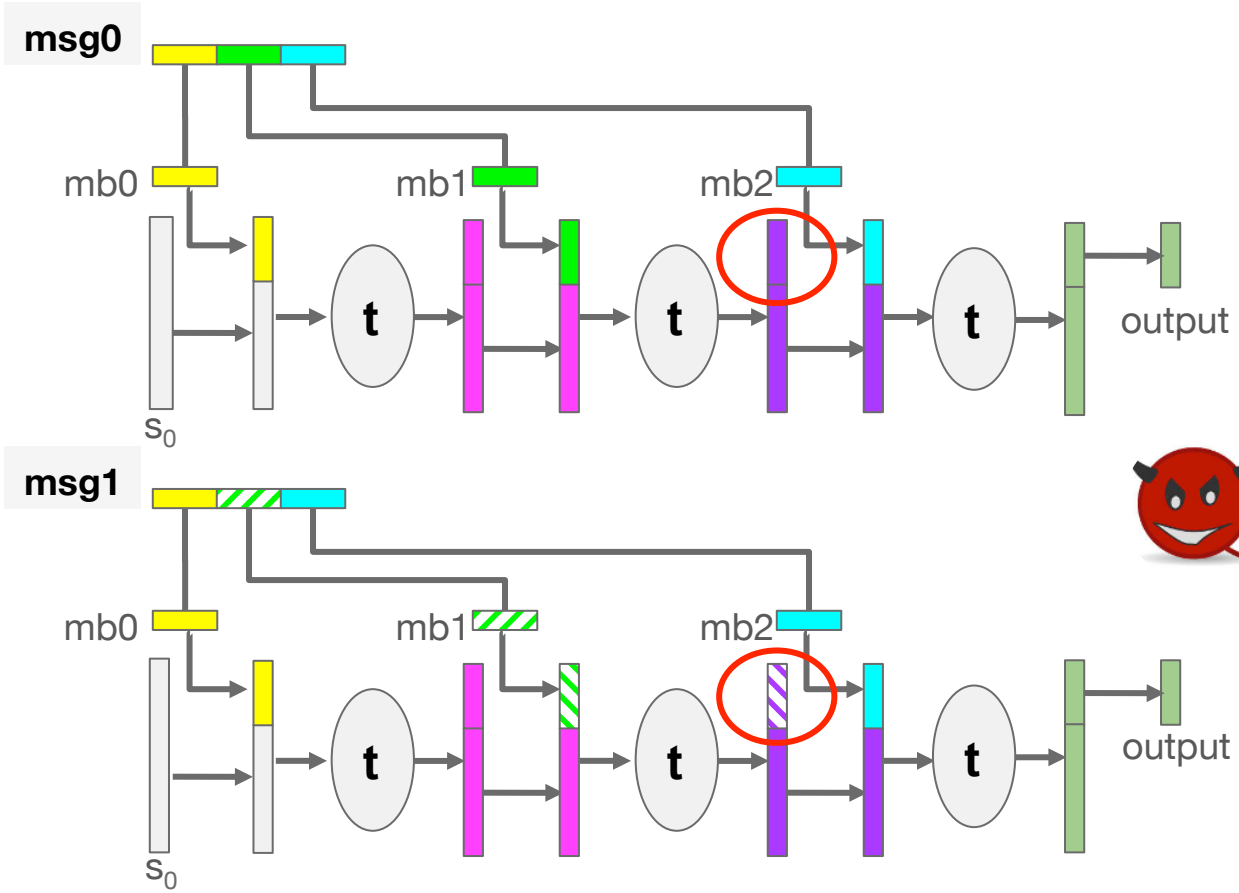
How do we create collisions in Curl-P-27?

Curl-P-27 is built on the sponge construction



Differences in the first third of the state are erased as new message blocks are copied

How do we create collisions in Curl-P-27?



Plan: ensure all the diffs are in first 3rd of the state

```
neha@ben:$
```

In this talk…

- An attack on IOTA’s multisig
- Breaking the Curl-P-27 hash function
- **Discussion**

IOTA Fixes Our Signature Forgery Vulnerability

- In July 2017 we disclosed this to the IOTA devs
  ...in response the IOTA devs replaced Curl-P-27 with Kerl

Kerl is used in IOTA for the following tasks:

Functionality	Curl-P-27	Curl-P-81	Kerl
Address generation			V^
Signature generation			V
Signature verification			V
Essence calculation (bundleHash)			V
Proof of Work		V	
Transaction Hash		V	
Milestone verification	V		

Curl-P-N: N number of rounds

https://github.com/iotaledger/kerl

IOTA claims this was a backdoor

"[..] *Curl-P was indeed deployed in the open-source IOTA protocol code as a copy-protection mechanism to prevent bad actors cloning the protocol and using it for nefarious purposes. Once the practical collisions were uncovered, its purpose as a copy-protection mechanism was of course rendered obsolete*"

In response to Ethan's question "*Did we discover a copy-protection backdoor in IOTA?*"
they write: "*The answer to the first question is of course, yes, as we have explained above.*"

Read IOTA's full statement at blog.iota.org/11fdccc9eb6d

Takeaways

1. We exploited weaknesses in Curl-P-27 to create chosen message signature forgery attacks
2. Don’t roll your own crypto
3. Cryptocurrencies have many interesting security and cryptographic challenges!

github.com/mit-dci/tangled-curl

Troika:

a ternary hash function

Reference document

Version 1.0.1

December 21, 2018

Epilogue: A new hash function appears

- In December 2018 IOTA announced the creation of a new ternary hash function Troika designed by Cybercrypt A/S
- €200,000 prize pool to break round-reduced variants

> *"Currently IOTA uses the relatively hardware intensive NIST standard SHA-3/ Keccak for crucial operations for maximal security."*
>
> *"[…] we […] started tackling the hardware side with new thinking in computational processing. A next generation of microprocessor architecture based on ternary logic for ultimate efficiency in IoT is the result. (A deep dive blog post on trinary's superiority over binary will come soon)."*

Read IOTA's full statements at blog.iota.org/678e741315e8 and blog.iota.org/615d2df79001

A note on cryptocurrency security…

- Increasing number of cryptocurrencies and codebases
- Attackers can easily and anonymously exploit bugs for financial gain
- Challenging space to determine best practices for reporting, disclosure, deploying fixes, and communication

narula@mit.edu