mas.s62
lecture 17
coinjoin, signature aggregation

2018-04-11

Tadge Dryja

today

privacy

- coinjoin

aggregate signatures

- schnorr multi-signatures
- aggregation and attacks

privacy

related terms: anonymity, fungibility

whatever the term, I don't need any.

I've got nothing to hide.

privacy

if you don't have anything to hide, you don't have any bitcoin

literally; the instant your private key is publicly known, someone will take your bitcoins

fungibility

often used as a euphemism for privacy / anonymity

means every bitcoin is "the same"

important for things that strive to be money

gold is fungible, diamonds are not

fungibility

currency legally considered fungible

Crawfurd v The Royal Bank (1749)

guy writes his name on a £20 note, loses it. The note shows up later at the bank, he demands it back.
Court says nope, that's not how money works.

fungibility

bitcoin does not enjoy the same legal protections; not considered currency by most govts

so it's up to the software to enforce fungibility of the coins

real world case

customer buys coins

customer transfers coins to UK betting shop. bets on game & wins

customer transfers winnings back to US exchange to sell. exchange closes the account as violation of ToS

problem...?

real world case

those coins are "worth less" than other coins, because of where they're from

very un-money-like!

if we want to make bitcoin money, how to fix this?

address re-use

simplest loss of privacy

persistent use of a pubkey

web explorers treat addresses as having balances

guessing change outputs

tx with 1 in, 2 out

input: 10 coins

output a: 1 coin

output b: 8.9997 coin

guess which goes to the same person

anonymity set

terms: anonymous, pseudonymous

anonymity set

even if I can't trace the bitcoins, I know it belongs to someone who has bitcoins! Which most people don't.

try to increase anonymity set

how to lose the trail

bitcoin mixers!

coins at address A

send 10 coins to the mixer, addr B

later, 4 coins to addr C

later, 6 coins to addr D

mixers

mixers work well

potential anonymity set is all other users of mixer

problem: mixers disappear with everyone's money. consistently.

coinjoin

I taint rich (maxwell, bitcointalk.org, 2013)

mixing multiple users within a single transaction

coinjoin tx

two different people in the same tx

input 0 user A signature 10 coins	output 0 address C 2 coins
input 1 user B signature 2 coins	output 1 address D 10 coins

coinjoin tx

fun first (2nd?) tx

69d9d66aae4812b6cf156f32267b773fb2118db696bb847ebd3454a198b59fbd

input 0 user A signature 10 coins	output 0 address C 2 coins
input 1 user B signature 2 coins	output 1 address D 10 coins

coinjoin tx

problems with this model?

any way to tell who's who?

input 0 user A signature 10 coins	output 0 address C 2 coins
input 1 user B signature 2 coins	output 1 address D 10 coins

coinjoin tx

gee, maybe A->D, B->C

amounts are different

input 0 user A signature 10 coins	output 0 address C 2 coins
input 1 user B signature 2 coins	output 1 address D 10 coins

coinjoin tx

how about this?

A signature 10 coins	address C 1 coin
B signature 2 coins	address D 7 coins
	address E 1 coin
	address F 3 coins

coinjoin tx

how about this?

... nice try but still no

A signature 10 coins	address C 1 coin
B signature 2 coins	address D 7 coins
	address E 1 coin
	address F 3 coins

coinjoin tx

now?

A signature 10 coins	address C 2 coins
B signature 2 coins	address D 2 coins
	address E 8 coins

coinjoin tx

this actually works; unclear if output C is from user A or B

A signature 10 coins	address C 2 coins
B signature 2 coins	address D 2 coins
	address E 8 coins

improving on coinjoin

have more users, bigger anonymity set

problem: users themselves know the mapping of inputs to outputs, can leak this info, hurtning anonymity

improving on coinjoin

coinshuffle: pre-coinjoin messaging to shuffle inputs and outputs

if at least 2 participants are honest, mapping is private

coinshuffle

everyone make public keys, send to everyone else. everyone also broadcast inputs

encrypt your output with everyone's pubkeys sequentially

$enc_c(enc_b(enc_a(output)))$ -> hand to a

coinshuffle

user a receives encrypted outputs, shuffles and decrypts

hands still encrypted outputs to next user, who decrypts, shuffles

final user gets the outputs, but can't tell which belong to whom

everyone signs this tx

real world issues

some people use this!

... which people use this?

limited anonymity set of people who really want anonymity.

which is not the anonymity set the people who want anonymity want.

make coinjoin cheaper

people don't care about privacy

other people's privacy = externality

everyone likes cheaper txs though

make coinjoin cheaper

privacy and scalability can work together

less information to store, less information to link to users

aggregate signatures

current signatures

input 0 user A signature 10 coins	output 0 address E 2 coins
input 1 user B signature 2 coins	output 1 address F 10 coins

aggregate signatures

aggregate signatures

input 0 10 coins	output 0 address E 2 coins
input 1 C = A+B signature 2 coins	output 1 address F 10 coins

aggregate signatures

how to make this signature?

Given

pubkeys A, B

message m

need one signature R, s

aggregate signatures

signature equation

$s = k - h(m, R)c$

$sG = R - h(m, R)C$

make c = a+b, but need to not share private keys

aggregate signatures

first, share R

alice: make k_a, compute R_a, share R_a

bob: make k_b, compute R_b, share R_b

aggregate signatures

next, add R

both: compute $R = R_a + R_b$

aggregate signatures

next, compute s's

alice: $s_a = k_a - h(m, R)a$

bob: $s_b = k_b - h(m, R)b$

share s_a and s_b

aggregate signatures

finally, compute s sum

$s = s_a + s_b$

$= k_a + k_b - h(m, R)a - h(m, R)b$

$= k - h(m, R)(a+b)$

$sG = R - h(m, R)C$

works!

aggregate signatures

now users can save space, only 1 signature for n inputs

input 0 10 coins	output 0 address E 2 coins
input 1 C signature 2 coins	output 1 address F 10 coins

key attacks

problem:

wait, I didn't sign that...

input 0 40000 coins	output 0 address E 40002 coins
input 1 user A&B signature 2 coins	

rogue key attacks

observe (rich) key A on network

make q, compute qG = Q

compute B = Q - A

send some coins to key B

note that you don't know b, and can't sign

rogue key attacks

spend from B and A

you don't know b, you don't know a

even though you don't know the private key for either, you know the private key for both!

c = a+b = a+(q-a) = q

rogue key attacks

require proof of knowledge of b

make b sign a message before combining keys

rogue key attacks

require proof of knowledge of b

make b sign a message before combining keys

... but the whole point was to aggregate signatures!

delinearization

redefine signatures - still send to C

instead of singing with C=A+B, sign with C=(A*h(A))+(B*h(B))

delinearization

sign with C = A*h(A)+B*h(B)

c = a*h(A) + b*h(B)

I know b = q - a, I know q

c = a*h(A) + (q-a)*h(Q-A)

delinearization

sign with C = A*h(A)+B*h(B)

c = a*h(A) + b*h(B)

I know b = q - a, I know q

c = a*h(A) + (q-a)*h(Q-A)

can't get rid of a*h(A) term

Wagner's birthday

this actually isn't enough!

Wagner: A Generalized Birthday Problem

Finding a collision is hard, right?

$2^{n/2}$ time

but that's a 2-collision

Wagner's birthday

2 collision: find A, B s.t. A = B

general collision

find $A_0, A_1 ... A_i$, $B_0, B_1 ... B_j$ s.t.

$\Sigma A = \Sigma B$

if you have lots of As and Bs, gets easier

improved delinearization

take the hash of all the keys together z = h(A,B)

sign with C = A*h(z,0)+B*h(z,1)

c = a*h(z,0) + b*h(z,1)

this works, paper calls it "MuSig"

aggregate signatures

first use: within my own wallet

saves space

input 0 (mine) 2 coins	output 0 address E 4 coins
input 1 (mine) C signature 3 coins	output 1 address F 1 coin

aggregate signatures

cooler use: with coinjoin

A 3 coins	address E 3 coin
B 3 coins	address F 3 coins
C 3 coins	address G 3 coin
D 3 coins signature	address H 3 coins

aggregate signatures

helps scalability and privacy

coinjoin tx is cheaper than solo tx

one giant tx per block, with 1 sig?

what about amounts... still an issue

(next time: how to mix amounts)

MIT OpenCourseWare
https://ocw.mit.edu/

MAS.S62 Cryptocurrency Engineering and Design

Spring 2018

For information about citing these materials or our Terms of Use, visit: https://ocw.mit.edu/terms.