

“You paid a bitcent, here's my thoughts”

2015-04-20 meetup

(no refunds)

- Privacy in Bitcoin Core 0.10.1 and 0.11
- Requirements for future multisignature,
 - ACE^{UP}: Accountable, Composable, Efficient, Usable, Private
- Some thoughts on “Selection Cryptography”

Greg Maxwell

<greg@xiph.org>

DE47 BC9E 6D2D A6B0 2DC6 10B1 AC85 9362 B041 3BFA

Privacy improvements in Bitcoin Core 0.10.1/0.11

Why Privacy?

- Privacy is an essential characteristic for a money-like good; without it:
 - Your business strategy and deals are made public
 - Your landlord hikes your rent when you get a raise
 - Nosey inlaws/neighbors nitpicking your spending
 - Thieves and con-artists can improve their targeting
- Privacy and fungibility go hand-in-hand
- Transparent and accountable systems can be built out of private and fungible ones; doing the reverse is much harder.

Why Privacy?

- Privacy is an essential characteristic for a money-like good, without it:
 - “Money is not speech, it is money. But the expenditure of money enables speech, and that expenditure is often necessary to communicate a message,[...]. A law that forbade the expenditure of money to communicate could effectively suppress the message.”
 - Your business strategy and your public
 - Your landlord hikes your rent and you raise
 - Nosey inlaws/neighbors nitpicking your spending
 - Government could repress speech by “attacking all levels of the production and dissemination of ideas,” for “effective public communication requires the speaker to make use of the services of others”
 - Justice Kennedy
(*Citizens United*)
quoting Justice Scalia
(*McConnell v. FEC*)
- Transparent and accountable systems can be built out of private and fungible goods, though the reverse is much harder.

Why Privacy in Bitcoin Core?

- Best practices implementation; “full node” model has strong and fundamental privacy advantages
- I've been asked by researchers and companies to “not fix” some privacy bugs
- We wouldn't add a privacy-harming phone home; I think not-fixing is morally equivalent
(or even worse!)
- Software should serve one interest: its users

Recent active attacks

- Surveillance nodes that track transactions have existed since at least 2011, resulting in reports of false allegations and other drama
- Farms of fake nodes attempt to capture more users' connectivity to get better visibility
- Basic protections longstanding in Bitcoin Core (e.g. netgroup limits)
- Weaknesses here encourage wasting network resources and cause outages

Recent active attacks

- Longstanding general advice is to run Bitcoin Core over Tor
 - Not perfect: Tor is weak against state actors, vulnerable to DOS, but usually strictly superior
- Some argue the active attacks are unlawful...
...but we already have to defend against attackers who don't care about the rule of law
(organized crime groups, NSA, GCHQ, etc.)
- Fortunately—room to improve the tech

Anti-“eclipse attacks”

- Sybils try to make your node forget about the honest network and mostly connect to them
- Consensus security needs only a single honest peer, but privacy is weakened by having a single dishonest one
- Recent simulation research showed ways attackers could gain more concentration in node memory than intended
- Expanded tables, removed some randomness, ignoring unsolicited addr floods, all improved for 0.10.1

Further connection anti-sybil

- Several proposals for hashcashy POW schemes to make using up lots of connections costly, tricky to balance against normal users
 - I have designs that are storage hard, and allow partial reusability but they're complex
- Tool to run alongside Bitcoin Core that uses private set intersection to ban mass connectors

Other network level improvements

- 0.11 will avoid reusing circuits on Tor
 - Prevents linking your connections with each other (partitioning risk), or with your web connections (privacy)
- 0.10.1 reduces leakage about the network topology from addr timestamps
 - Reduces risk from targeted DOS attacks, and knowledge gained from transaction timing
- Improvements to increase trickle batches in progress for 0.11

Wallet Relay control improvement

- Wallet rebroadcasts can give away a node's origin to an attacker watching the network
 - and if they don't, automatic rebroadcast probably does
- New flag for 0.11 disables wallet transaction broadcast; user can manually trigger
- Opens up the possibility to run an interface alongside Bitcoin Core to send transactions via Bitmessage or a mixmaster network
(but someone needs to write it!)

Full-node usability \subset more privacy

- Privacy is more or less broken for SPV, bloom filters in practice are uniquely identifying, etc.
- Full nodes are completely private for receiving
- But 30+GB disk space is painful
- Pruning in the pipeline gets 0.11 down to ~1GB when enabled; just as private
- Bursty bandwidth irritates buffer-bloat, rate-limiting for 0.11; is automatic tuning possible?

Lots left to even get started on

- Wallet coin selection strongly assumes no address reuse; any reuse breaks privacy
 - Easy to fix: group addresses select groups; fixes slowed by inadequate wallet tests
- CoinJoin and other active privacy techniques
- “Stealth addresses” (I hate the name; should be “reusable addresses”); existing proposal is unmaintained; causes even greater SPV privacy loss
 - Huge cost in rolling out another address type, need to get it right

Requirements for the future of multisignature

ACE^{UP}

Accountable, Composable, Efficient, Usable, Private

Bitcoin Multisignature today

- “No tagbacks!”; turns out autonomy is hard, security is hard, nothing can be trusted, and everything is broken. More or less.
- Multisignature potential security improvement. Taken forever to get deployed.
- It's costly. 2 of 3 means roughly a 2.5x increase in transaction size! 2 of 3 is kind of a weak policy, also 2x slower to verify. Direct hit on decentralization.
- We can do better!

A Crypto-detour

- Schnorr is an alternative to ECDSA
 - but was patented until recently; can be implemented over the same groups
- Can do multisignature by ... addition:
 - $\text{Sign}(P_1, M) = \{R_1, S_1\}$
 - $\text{Sign}(P_2, M) = \{R_2, S_2\}$
 - $\text{Sign}(P_1 + P_2, M) = \{R_1 + R_2, S_1 + S_2\}$

(more or less: needs another round of communications to agree on R_n in advance of producing S_n ; care needed to make sure $P_2 \neq P_x - P_1$)

A Crypto-detour

- So a N of N is as network efficient as a 1 of 1.
 - and can be extended to arbitrary thresholds
- A panacea! The efficiency problem is solved!
- But the extension to thresholds require the signers to interact more
- Hmm... are there other requirements we need to worry about?

Accountability

- In the existing multiparty scheme, once a signature exists anyone who sees can tell which of the participants signed
- This is important because policy and security doesn't end at the blockchain
- If your vendor cheats, you want to know—and you want to be able to prove it to others
 - (including other customers or potentially the courts)
- Direct Schnorr thresholds are not accountable!

Usability

- In the existing multiparty scheme, someone can draft a transaction and pass it through enough signers once
 - direct Schnorr requires an additional round to agree on the nonce either at sign time or in advance
 - some other schemes need MANY additional rounds: a bummer if your keys are in a safe
- Bitcoin multisig pubkey can be made without interaction
 - direct Schnorr cannot except for N of N

Privacy

- Accountability requires the participants (or people they choose) to know who signed
- Privacy requires random third parties learn as little about your policy as possible
- If someone knows you are 2 of 3, they know they only need to kidnap two people to sign
 - good security suggests not revealing more than you must
- An unusual policy identifies your transactions; competitive repercussions

What about “threshold ECDSA”?

- Fancy crypto; first attempts didn't actually work without a trusted dealer, *maybe* it works without one now
- Requires many rounds of interaction to generate the pubkey and to sign; hurts usability
- Not accountable
- But it is efficient and maybe available, may find some applications where the round and non-accountability aren't an issue

TREECHECKSIG

- N-of-N schnorr is efficient, usable, but not accountable; less usable with N-of-M
- Enumerate all the M-choose-N possible N-of-N satisfactions; hash tree over this is the pubkey
- Signature reveals the $\log_2(\text{binomial}(m,n))$ sized proof for membership, plus a single signature
- Accountable, can be fairly private, usable
- Efficiency is better than Bitcoin Multisig
- Problem: tree starts becoming too big to compute

MULTICHECKSIG

- Avoid precomputing all the possible N-of-N?
- Give the verifier all the points, signature encodes which threshold subset to sum plus a single N-of-N signature
- Accountable, not private, but usability is good
- Linear in M , instead of M and N as in Bitcoin
 - verification is nearly constant speed (add is fast)
 - better than now, not as good as we'd like

POLYCHECKSIG

- If the points were already sums? [3 of 4 ex.]

$$P_1 = P_A + P_B + P_C + P_D$$

$$P_2 = P_A + 2P_B + 3P_C + 4P_D$$

$$P_v = 3P_1 - P_2 = 2P_A + P_B - P_D \quad \underline{\text{C is canceled!}}$$

- Works for any number; needs $M-N+1$ terms
- Encode in an unbalanced hash tree
- Accountable but if all sign it looks like 1-of-1

“Composable”?

- One reason someone can't use a multisig policy is because they need to be in someone else's multisig policy
 - e.g. a 2-of-3 of 2-of-3 users
 - Bitcoin Script and P2SH can support multisig of multisig; no wallet software does; the efficiency is a killer
- Access policies are monotone functions
- Build a higher-level construction that allows composition: sign the parts you understand

Comparison

Scheme	Accountable	Usable	Private	Comms
Bitcoin	Y	Y	N	0 + 0.5
Schnorr	N	~N	Y	Prop N, M + 1
TREE	Y	Y	~Y	0 + 1
MULTI	Y	Y	N	0 + 1
POLY	Y	Y	Y	0 + 1

Scheme	Size	2-of-3	13-of-15	50/100	990/1000	CPU
Bitcoin	34N+74M	250	1472	7100	107260	M
Schnorr	34+74	108	108	108	108	1
TREE	$\lg(B(M,N))*32+74$	172	332	-	-	1+0.01*N
MULTI	34N+74	176	584	3474	34074	1+0.01*N
POLY	$\leq(M-N+1)*34+74$	142	142 - 176	142 - 1808	142 - 448	1+(M-N)/2

The art of Selection Cryptography

What is cryptography?

- Most common definition: *(MW, similar in OED, WP, etc.)*
 - *(1) Secret writing*
 - *(2) the enciphering and deciphering of messages in secret code or cipher*
- What about:
 - digital signatures?
 - compact / zero-knowledge proofs?
 - private information retrieval?
 - hash functions, multiparty computation, cipher negotiation (e.g. in TLS), crypto-currency, etc...?
- These are all *clearly* cryptography

As a young cypherpunk in the 90s I heard the rallying cry—

“Information wants to be free”

—and I knew it to be true

Computers convert much of what *matters* to mankind into *pure* information. Networks can carry information everywhere without distortion.

With access to computers and networks we can equalize power imbalances and give people the freedom they need to fulfill their potential.

Let's not misunderstand a fundamental law of nature as a political aspiration:

Information *does* want to be free.

But the result isn't always pretty...

My email wants to be read by the NSA

My login attempts want to be indistinguishable from yours

My reading habits want to disclose my plans to my political opponents

Valuable information I seek wants to be equally available to your cheaply-sent spam flood

All spends of my digital cash want to be equally valid

... And when all information is free, the already powerful can afford to exploit these facts more completely than the disempowered

So, Cryptography?

I think a better definition might be

- ***Cryptography*** is the art and science we use to fight the fundamental nature of information, to bend it to our political and moral will, and to direct it to human ends against all chance and efforts to oppose it.

This is an expansive definition

- It encompasses a few things that aren't normally thought of as cryptography; e.g., parts of computer security, debatably some parts of law
- I don't offer it lightly; it leads to better intuitions than any definition I've encountered before

My email wants to be read by the NSA

- Apply: Cryptographic technique of strong encryption

My login attempts want to be indistinguishable from yours

- Apply: Cryptographic technique of digital signatures

My reading habits want to disclose my plans to my political opponents

- Apply: Cryptographic tool of private information retrieval

Valuable information I seek wants to be equally available to your cheaply sent spam flood

- Apply: Cryptographic tool of hashcash

All spends of my digital cash want to be equally valid

- Apply: Cryptographic tool of decentralized consensus

(or the cryptographic tool of a protocol with a trusted third party, and digital signatures)

Fighting a law of nature sounds hard

- Secure cryptography may not be possible
- Cryptosystems, today, are only ever secure given strong assumptions
 - A provably secure asymmetric cryptosystem is directly a proof that $P \neq NP$
- Attacks on cryptosystems are themselves also information that *wants to be free*
- Virtually every cryptographic idea is broken
- *Provable cryptography* is about showing constructions to not be insecure even when their assumptions hold
 - Easiest way to make a provable cryptosystem is to pick stronger assumptions! (Guess what happens?)

Civil engineering is also fighting against nature.

- But has had a long time to mature: loss of life from design flaws was common in practice not so long ago (*and still is in some places*)
- Seldom would we be crazy enough to ask a building withstand a concerted effort of thousands to destroy it...
... but for cryptography that is usually what we must ask; it wouldn't be worth even trying without the power of software as a building material.

Software reliability today is *unfortunate*.

- **Very** complex mechanical engineering project: 200k parts
- Firefox? 17 million lines of code
- We have better tools for building software, which is why it's possible at all
- I find severe bugs in software almost every day, and cryptography is the hardest:

“Software testing is making sure your program does what it's supposed to, security testing is making sure that's *all* it does.”

“We're doomed, I get it...”

In response to this challenge the trope is:

“Never write your own cryptography”

Or, “abstinence only” cryptographic education

- People adopt an overly narrow definition of cryptography- one which excludes the parts that break most often
- *At best* it reduces the problem to *selecting* secure parts and then not invalidating their assumptions

If people count on a program to fight the nature of information, it's cryptography

Bad news

I cannot tell you how to practice safe cryptography in general

- No one knows, might not be possible

We do know some things that are unsafe

- They're usually complex and application specific

The best I can think we can do is

- Face the challenge frankly and understand the risks
- Learn from our mistakes
- Advance the art

Selecting cryptography –*is* cryptography

So what does good “Selection Cryptography” look like?

Norm in the Bitcoin industry to build tools out of primitives “found on github”

Unless you're a domain expert, you probably can't review it.

Step (1) Think for a bit and ask “If this code is broken or malicious how much trouble could it create?”

Step (2) If “not much” go back to step 1.

(Really: What if the installer roots your host? All code is dangerous)

Step (3) Given the risks, what do we know that mitigates the risk?

When selection fails–

(a case study)

A ubiquitous Javascript “Secure Random” deployed on tons of sites in the past

```
25 // Initialize the pool with junk if needed.
26 if(rng_pool == null) {
27   rng_pool = new Array();
28   rng_pptr = 0;
29   var t;
30   if(navigator.appName == "Netscape" && navigator.appVersion < "5" && window.crypto)
31   {
32     // Extract entropy (256 bits) from NS4 RNG if available
33     var z = window.crypto.random(32);
34     for(t = 0; t < z.length; ++t)
35       rng_pool[rng_pptr++] = z.charCodeAt(t) & 255;
36   }
37   while(rng_pptr < rng_psize) { // extract some randomness from Math.random()
38     t = Math.floor(65536 * Math.random());
39     rng_pool[rng_pptr++] = t >>> 8;
40     rng_pool[rng_pptr++] = t & 255;
41   }
42   rng_pptr = 0;
```

Crypto RNG unavailable? “No problem”

Read 16-bits at a time, from what is often a 48-bit LCG, seeded from the time at browser start

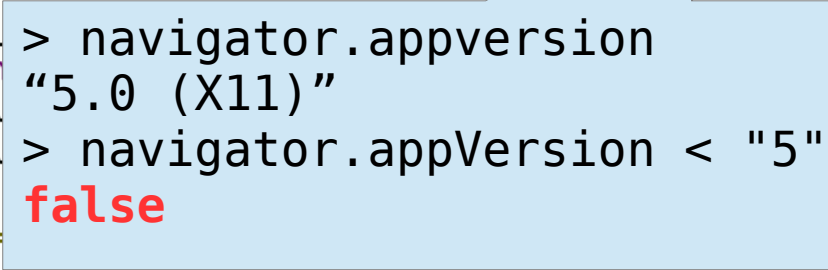
Feeding a highly biased RC4 RNG
Keep in mind: ECDSA *assumes* the nonce is uniform,
A 1-bit bias can result in key loss with enough reuse

When selection fails–

(a case study, cont.)

Any remotely modern browser has `window.crypto` ...*except in web-workers*; but that doesn't even matter here:

```
25 // Initialize the pool with junk if needed.
26 if(rng_pool == null) {
27   rng_pool = new Array();
28   rng_pptr = 0;
29   var t;
30   if(navigator.appName == "Netscape" && navigator.appVersion < "5" && window.crypto)
31     {
32       // Extract entropy (256 bits) from NS4 RNG if available
33       var z = window.crypto.random(32);
34       for(t = 0; t < z.length; ++t)
35         rng_pool[rng_pptr++] = z.charCodeAt(t) % 255;
36     }
37   while(rng_pool.length < 256)
38     rng_pool[t++] = Math.random() * 255;
39   rng_pptr = t;
40 }
41 rng_pptr = 0;
```



```
> navigator.appVersion
"5.0 (X11)"
> navigator.appVersion < "5"
false
```

When selection fails—

(another case study)

Wallet deploys “ECIES” encryption using third party code

Wallet software fixed the use of a non-cryptographic RNG

But it wasn't actually ECIES and they didn't fix:

- 2^{16} calls to oracle let you decrypt arbitrary messages
- Directly leaks 7 bits of plaintext per 256 bits of input
- Corrupts many possible messages (e.g. all 1s message)
- Plus lots of less cryptographic suitability-for-purposes issues

Due to domain expertise I found these issues in ~10 minutes.

Yet I consider the wallet authors highly competent—another (competent) vendor didn't even fix the RNG in code from the same source.

Absent domain expertise what can we do better?

Risk Mitigation (1 of 3)

(0) Is the software intended for your purposes?

(1) Are the cryptographic considerations being taken seriously?

- What is the response to security concerns / issues?

(2) The review process... is there one?

- Cryptography is a team sport. *No one* can do it alone. Review isn't always visible, but it can be; we should demand that it be made visible.
- Widespread use is often taken to be a proxy for review, but that hardly works

(3) What is the experience of the authors?

- It's unlikely that cryptography written without deep understanding by just following a recipe will turn out secure. Understanding leaves evidence, but don't mistake punditry for expertise.
- Are the authors striving for domain expertise? (You probably can't tell actual expertise from technobabble if you're not an expert yourself; look for the process)

Risk Mitigation (2 of 3)

(4) Is the software documented?

- Externally and internally? Are its assumptions clearly documented?
If not, why do you think you aren't violating them?

(5) Is the software portable?

(6) Is the software tested?

- Automated testing is one of the most powerful tools available
- If there are tests: try introducing a bug, do the tests catch it?

(7) Does the software adopt best practices?

- Look for evidence in review even if you don't know them
- Does it “fail safe”? Is it intentionally hard to misuse?
- “What have the authors done to mitigate risk?” If it's not clear, ask;
if they don't have answers they haven't thought about it too hard

Risk Mitigation (3 of 3)

Much of this reduces to looking for strong evidence of conscientiousness

In my experience the number of people recommending something is inversely correlated with the rigorousness of its development

- It's easier to offer lots of whiz-bang features if you don't care about being secure
- Iteration doesn't work well here: privacy lost cannot be recovered (nor Bitcoins!)

All of these measures are imprecise...

Beware crypto-laundering: cryptography is built out of cryptography; your vendors face the same challenges you do

You can trust, but verify--without verifying there is less reason to do the work to get it right

What happens when what you want *itself* violates good practices?

- Pragmatic has its place, but beware of biasing against competence

There is a lot left to learn

I hardly think I know more than the most basic essentials for selection cryptography

If we don't make a conscious effort to advance the art and demand better, nothing will improve

(or it'll be “improved” by regulatory intervention...)

With effort we can someday evolve this art into a science

I'm very interested in the best techniques you've found to select stronger systems

(not to mention, techniques to build stronger systems—but that's its own talk)

Thanks for your time

Greg Maxwell

<greg@xiph.org>

DE47 BC9E 6D2D A6B0 2DC6 10B1 AC85 9362 B041 3BFA