mas.s62

lecture 3

signatures

2018-02-14

Tadge Dryja
signatures
pset01 was about lamport signatures
There are other signature schemes, some with cool features
Hash-based, RSA, ECDSA, EC schnorr
multiple use hash signatures

Problem with lamport signatures:
Multiple signatures from one key allows forgeries

Solution: use more public keys
multiple use hash signatures

Easy way:

make 2 public keys, concatenate and publish

For signatures, indicate use of key1 or key2
multiple use hash signatures
Signatures: same size
Public keys: 2X size
Private keys: 2X size..?
multiple use hash signatures

Signatures: same size

Public keys: 2X size

Private keys: 2X size...

make a root private key, and
hash(root, 1) for key 1, hash(root, 2)
for key 2.

In fact, private key can be 32 bytes
multiple use hash signatures

<table>
<thead>
<tr>
<th>0</th>
<th>0,0</th>
<th>0,1</th>
<th>0,2</th>
<th>0,3</th>
<th>0,4</th>
<th>0,5</th>
<th>0,6</th>
<th>0,7</th>
<th>... (256)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,0</td>
<td>1,1</td>
<td>1,2</td>
<td>1,3</td>
<td>1,4</td>
<td>1,5</td>
<td>1,6</td>
<td>1,7</td>
<td>... (256)</td>
</tr>
</tbody>
</table>

howto 32 byte privkey:

0row hash(seed,0,0), hash(seed,0,1)..

1row hash(seed,1,0), hash(seed,1,1)..
multiple use hash signatures from 16KB to 32B. That's quite nice! Can we do that with the pubkey..? 32B pubkey...?
multiple use hash signatures from 16KB to 32B. That's quite nice!

Can we do that with the pubkey..?

32B pubkey...?

Commit to pubkey with hash(0 0 0 0 0 0 0 0) (hash of all 16KB)
multiple use hash signatures

private key -> 32B

public key -> 32B

signatures still big; actually get bigger. Include full pubkey in signature.

Can do better: commit to many pubkeys
multiple use hash signatures

Hash tree, or Merkle tree
multiple use hash signatures

Prove Pub0 inclusion given root
multiple use hash signatures

Prove Pub0 inclusion given root
multiple use hash signatures

Prove Pub0 inclusion given root
multiple use hash signatures

only need 2 extra hashes, one per row
Merkle trees
Invented for Lamport sigs
add $O(n)$ elements in $O(1)$ sized root
prove an element in the set with $O(\log n)$ intermediate hashes
Merkle trees
commit to 1024
signing keys
use root as pubkey
signature includes proof that signing key is a leaf in the tree
10*32 = 320B overhead. cool!
Merkle trees
commits to 1024
signing keys
use root as pubkey
signature includes proof that signing key is a leaf in the tree
10*32 = 320B overhead. cool!
need more power
Hash based signatures are cool.
But we can do better. More powerful signature schemes.

RSA ECDSA ECBN
RSA

Invented by locals :)  

Not used in Bitcoin (or any currency)

Used for chaumian blinded cash

Basic setup: make 2 primes: p, q

n = p*q
RSA

Given $p, q$, computing $n$ is easy.

Given $n$, finding $p, q$ is hard!

A one way function... but not a hash function.
RSA

Can do some fun math with this.

Set $e = 3$ (or 65537)

set $d =$ some number you can compute if you know $p$ or $q$.

$d = e^{-1} \mod (p-1)*(q-1)$

$n$ is public.  $d$ is private.

$p$, $q$ not needed after setup.  $e$ always the same
RSA

Sign: \[ s = m^d \mod n \]

Verify: \[ s^e \mod n = m \]

Can sign many times. And do lots of cool stuff.
RSA

RSA key sizes are smaller than hash based signatures; often 2048 bits (256 bytes)

Somewhat tricky to implement! Lots of ways to lose your private key

but Bitcoin (& other coins) uses elliptic curve signatures
Intermission

3 min, walk around, ask questions about pset . . .

then start on elliptic curves
elliptic curves

Curves. Bitcoin's curve is

\[ y^2 = x^3 + 7 \]

Simple, right?
elliptic curves define point addition
line of 3 points = 0
so $P+Q-R=0$
$P+Q=-R$
elliptic curves
point "multiplication"
take the tangent
to add a point
to itself
elliptic curves

note: actually "looks" more like this because it's modulo some big prime number

We don't compute it graphically so it's OK
point and scalar operations
(Note also works on exponents mod n)
a, b lowercase = scalar
A, B uppercase = point
what operations can we do?
point and scalar operations

scalars are regular unleaded numbers

\[ a+b \quad a-b \quad a\times b \quad a/b \]

everything is OK! just numbers!
point and scalar operations

Points have addition defined... but not multiplication and division

A+B  A-B  OK          A*B  A/B  NO

add & subtract OK, but can't multiply two points, or divide a point by a point. Not defined.
point and scalar operations

Mixed operations

A+b A-b NO A*b A/b OK

adding points and scalars is undefined

point times scalar OK; repeat the tangent doubling process. Division by scalar also possible.
point and scalar operations
roster of ops: what can we do

\[ a+b \ a-b \ a*b \ a/b \ (obvious) \]

\[ A+B \ A-B \ A*b \ A/b \]
point and scalar operations

roster of ops: what can we do

\[ a + b \] \[ a - b \] \[ a \times b \] \[ a / b \] (obvious)

\[ A + B \] \[ A - B \] \[ A \times b \] \[ A / b \]

Pick some random point \( G \)

That's the generator point \( G \)

Everyone agrees on \( G \)
EC private & public keys

private key $a = 256$ bit scalar

(same as one block from Lamport priv)

public key ?
EC private & public keys

private key $a = 256$ bit scalar

(same as one block from lamport priv)

public key $A = a \times G$

32 byte $x$ coord, 32 byte $y$ coord = 64B
EC private & public keys

private key \( a = 256 \) bit scalar
(same as one block from lamport priv)

public key \( A = a \times G \)

32 byte x coord, 32 byte y coord = 64B

since curve is symmetric about x-axis, can encode x-coord only and 1 bit for y. Down to 33 bytes.
ECDSA

What Bitcoin, other coins use today

It's ugly though...

Come up with another priv key $k$

$r = x$-coord of $k \times G$

$s = k^{-1}(\text{hash}(m) + a \times r)$
ECDSA

Made to avoid a patent on a better signature system

That patent has expired, we are free to use the simpler better algo that must not be named.
ECsig

Have message \(m\), privkey \(a\)

make \(k\), a new random private key

\[ R = k \times G \]

\( s = k - \text{hash}(m, R)a \)

signature = \(R, s\)
ECsig

given $R, s$ verify:

\[ s = k - \text{hash}(m, R)a \]

\[ sG = kG - \text{hash}(m, R)aG \]

\[ sG = R - \text{hash}(m, R)A \]

\[ R == sG + \text{hash}(m, R)A \]
ECsig

Make up k and compute s, R? but need a

\[ s = k - \text{hash}(m, R)a \]

without a, can't make a valid s
ECsig

Make up an s, solve for R?

\[ sG = R - \text{hash}(m, R)A \]
ECsig

Make up an s, solve for R?

\[ sG = R - \text{hash}(m, R)A \]

\[ R = \text{hash}(m, R)A + sG \]

\[ R = \text{hash}(R)! \text{ Can't compute, can't cancel out} \]
Fun with points

$A = aG$

$B = bG$

$aB = bA = (aG)b = (bG)a = (ab)G = C$

C is a "Diffie Hellman" point.

Super useful! If you know either a or b, you can compute C.
Fun with points

A = aG

B = bG

aB = bA = (aG)b = (bG)a = (ab)G = C

C is a "Diffie Hellman" point.
Fun with points

\[ A = aG \quad B = bG \]

\[ D = A+B = (a+b)G \]

sign with \( D \)? Can with \( d = a+b \) make a combined key \( D = A+B \); either party can reveal their side to the other to give signing ability
What do we do with this??
Nothing yet. Hard to program this stuff.

Ground work for cool stuff you can do with keys, transactions, signatures.

Fun new area! Non-experts (like me) can come up with new stuff!
Next pset: NameChain
Mine your name; get a high score
(hax0r names also OK)
pls don't DDoS the server :)