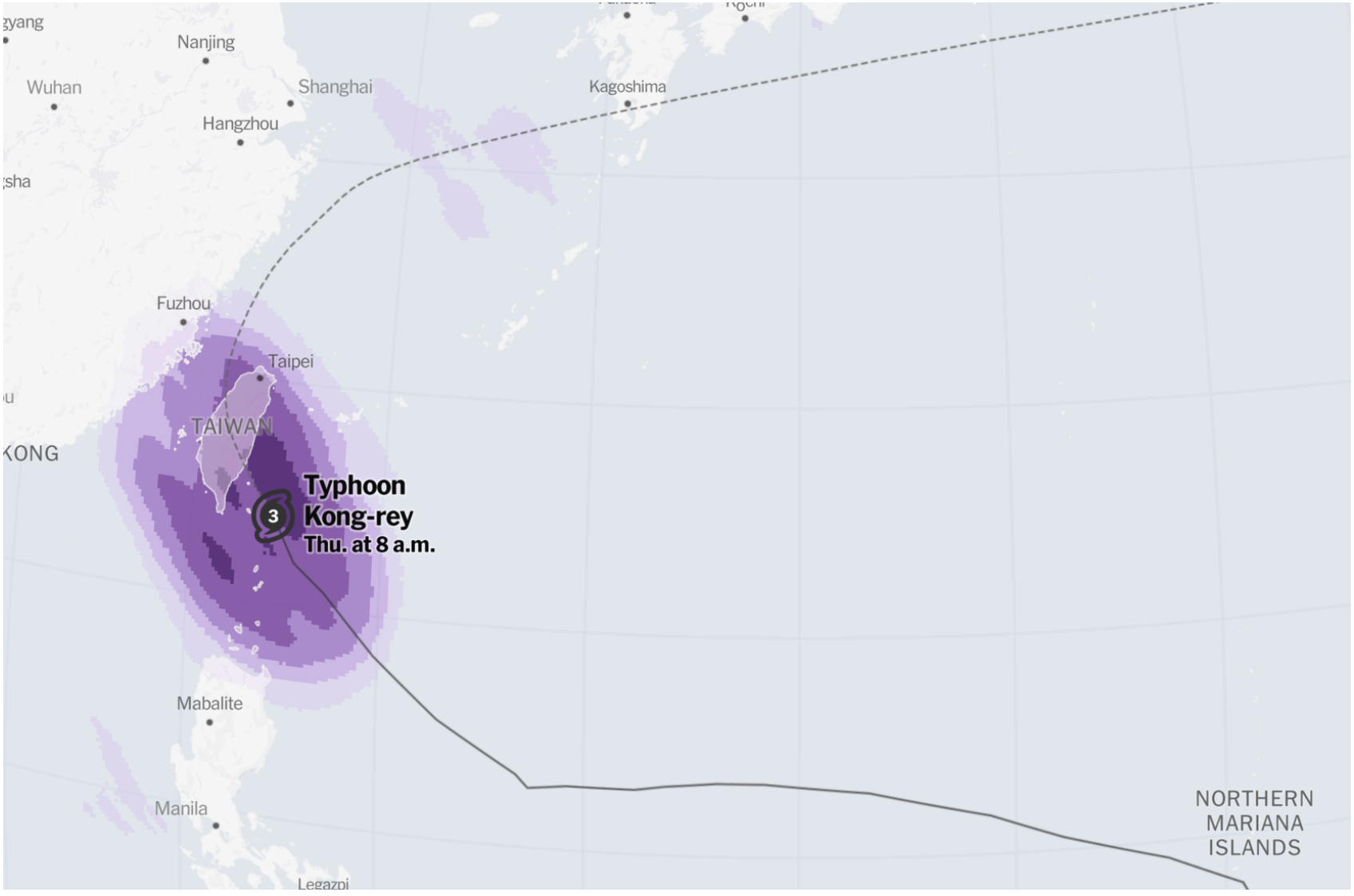
The State of the Art of ECC in Real-World Protocols

Nadia Heninger

UC San Diego

October 31, 2024

Parabolic curve cryptography?



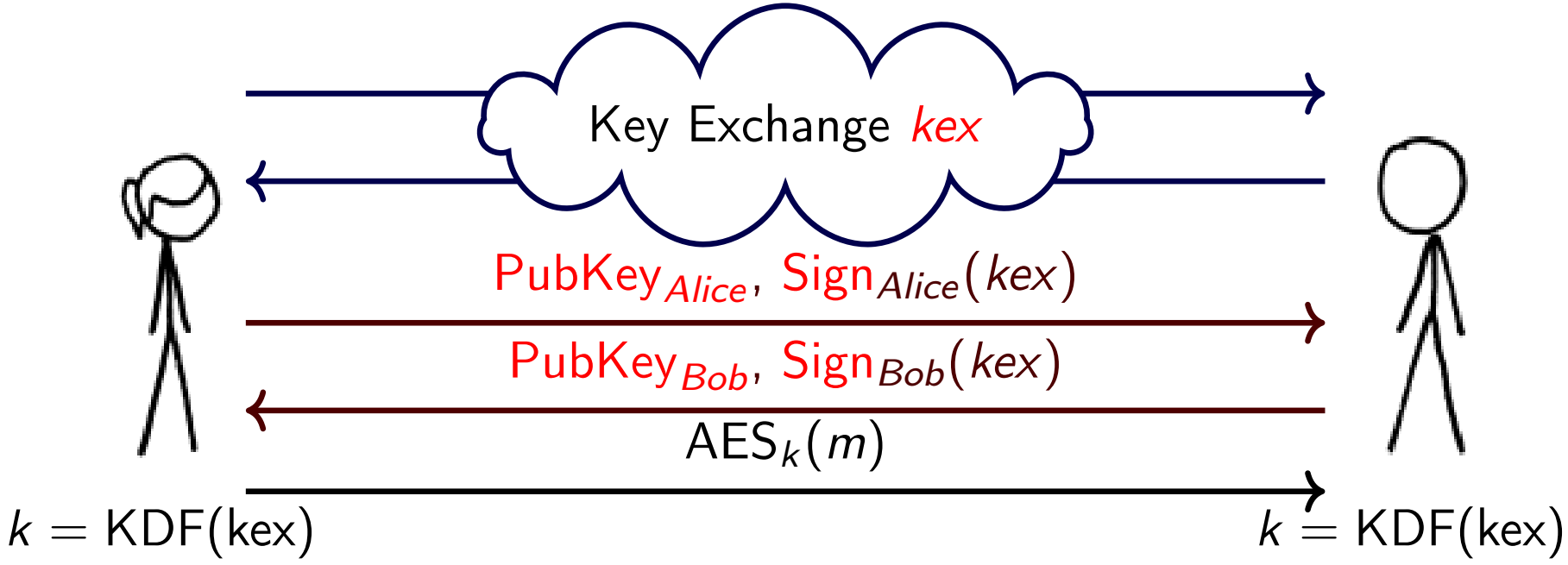
In this talk: non-“fancy” cryptography.

Part 1: Measurements.

Congrats elliptic curves, you've won!

Public key cryptography usage in communication protocols

Simplification of handshake for TLS, SSH, IKE, etc.



Elliptic curve cryptography used for:

1. Key exchange between client and server.
2. Signatures to authenticate key exchange.
3. Public key infrastructure signatures to authenticate signing keys.

TLS 1.2, simplified



client hello: client random

[list of cipher suites][list of curves]

server hello: server random, [cipher suite]

certificate = public key + CA signatures

server kex: curve name, curve point

$\text{Sign}_{\text{serverkey}}(\text{randoms, server kex})$

client kex: curve point

client finished: $\text{MAC}_{k_{m_c}}(\text{dialog})$

server finished: $\text{MAC}_{k_{m_s}}(\text{dialog})$

$\text{Enc}_{k_e}(\text{request})$

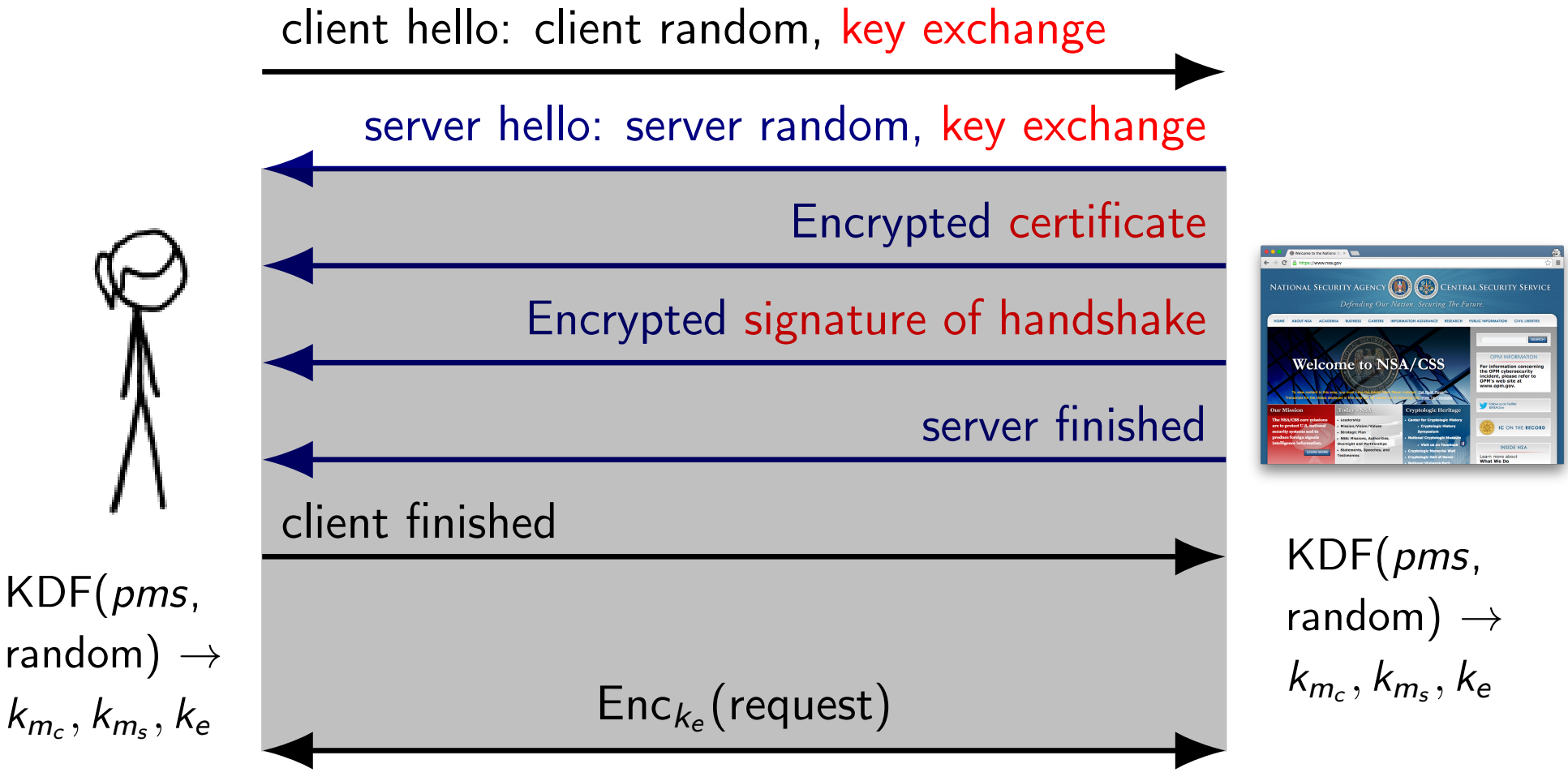


KDF(g^{ab}, random) $\rightarrow$ k_{m_c}, k_{m_s}, k_e

KDF(g^{ab}, random) $\rightarrow$ k_{m_c}, k_{m_s}, k_e

TLS 1.3

TLS 1.3 encrypts the handshake immediately after doing a Diffie-Hellman key exchange.



Active scanning measurement of network protocols

- **Active**: Use fast syn scanning to connect to visible IPv4 space, complete handshake with hosts that have open port.

  Example toolchain: Use ZMap to scan for open ports, ZGrab to complete protocol handshakes.

  - Useful only for examining server preferences.
  - Misses hosts requiring SNI.
  - Not representative of real traffic:
    - Overrepresents devices nobody connects to.
    - Underrepresents Google, Netflix, CDN traffic.
    - Essentially impossible to meaningfully scan protocols like IPsec that may not allow unknown connections.

Passive measurement of network protocols

- **Passive**: Examine traffic over real network.

  Example toolchain: Mirror traffic to an analysis machine running Zeek/Bro.

  - Difficult to examine traffic at scale.
  - Ethical and privacy issues.
  - Encrypted information (e.g. TLS 1.3 handshake) not visible.
  - Very different numbers from different networks.
  - We drop some CDN traffic because of limitations in our infrastructure.

Conclusion: Numbers should be interpreted with caveats.

TLS 1.2 cipher suites in passive network data

Data collected by George Sullivan

TLS 1.2 data from a 1-month period during August 2024 at UCSD.

TLS 1.2 is 31% of TLS connections in our data.

Cipher Suite	Handshakes	Percent
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	141M	48.72%
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	91M	31%
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	35M	12%
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	13M	5%
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	1.8M	0.6%
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	1.7M	0.6%

TLS 1.2 key exchange curves in passive network data

Data collected by George Sullivan

TLS 1.2 data from a 1-month period during August 2024 at UCSD.

TLS 1.2 is 31% of TLS connections in our data.

Algorithm	Handshakes	Percent
NIST P-256, secp256r1	100M	42%
NIST P-384, secp384r1	91M	38%
x25519	45M	19%
NIST P-521, secp521r1	2M	0.8%
sect571r1	190	0%
x448	11	0%
SM2	3	0%

TLS 1.2 handshake signatures in passive network data

Data collected by George Sullivan

TLS 1.2 data from a 1-month period during August 2024 at UCSD.

TLS 1.2 is 31% of TLS connections in our data.

Algorithm	Handshakes	Percent
rsa_pss_rsae_sha256	116M	48%
rsa_pkcs1_sha256	66M	27%
ecdsa_secp256r1_sha256	42M	17%
rsa_pkcs1_sha512	18M	7%
ecdsa_secp384r1_sha384	260K	0.1%
ecdsa_secp521r1_sha512:	110K	0.05%
rsa_pkcs1_sha1	100K	0.04%

TLS 1.2 cipher suites in active network scans

TLS 1.2 data from October 23, 2024 ZGrab scan.

Cipher Suite	Handshakes	Percent
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	31M	86%
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	3.8M	12%
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	1.4M	4%
TLS_RSA_WITH_RC4_128_SHA	800K	3%
TLS_RSA_WITH_AES_128_CBC_SHA	500K	2%
TLS_RSA_WITH_AES_256_CBC_SHA	500K	2%
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	300K	1%

Observe: Long tail of "servers" (mostly old network devices with http interfaces) that nobody connects to.

ZGrab does not seem to offer curve25519.

TLS 1.2 key exchange curves in active network scan

TLS 1.2 data from October 23, 2024 ZGrab scan.

Curve	Handshakes	Percent
NIST P-256, secp256r1	37M	97%
NIST P-384, secp384r1	800K	2%
NIST P-521, secp521r1	250K	1%

ZGrab does not seem to offer curve25519.

TLS 1.2 certificate keys in active network scan

TLS 1.2 data from October 23, 2024 ZGrab scan.

Algorithm	Handshake	Percent
RSA	35M	90%
NIST P-256	3.6M	9.5%
NIST P-384	200K	0.5%
NIST P-521	200	0%
NIST P-224	1	0%

TLS 1.3 key exchange algorithms

Data collected by George Sullivan

TLS 1.3 data from Sept. 11–Sept. 17, 2024 at UCSD.

TLS 1.3 is 68% of TLS connections in our data.

Algorithm	Handshakes	Percent
x25519	87M	69%
NIST P-384, secp384r1	15M	12%
X25519Kyber768Draft00	13M	10%
NIST P-521, secp521r1	8M	7%
NIST P-256, secp256r1	2.5M	2%
x448	14K	0.01%
SM2	3K	0.003%
SecP521r1Kyber1024	307	0%
X25519MLKEM768	19	0%

TLS 1.3 key exchange algorithms

Data collected by George Sullivan

TLS 1.3 data from Sept. 11–Sept. 17, 2024 at UCSD.

TLS 1.3 is 68% of TLS connections in our data.

Algorithm	Handshakes	Percent	SNI/Cert
x25519	87M	69%	
NIST P-384, secp384r1	15M	12%	
X25519Kyber768Draft00	13M	10%	Google
NIST P-521, secp521r1	8M	7%	
NIST P-256, secp256r1	2.5M	2%	
x448	14K	0.01%	
SM2	3K	0.003%	
SecP521r1Kyber1024	307	0%	ExpressVPN
X25519MLKEM768	19	0%	

TLS 1.3 key share types over time

Data collected by George Sullivan

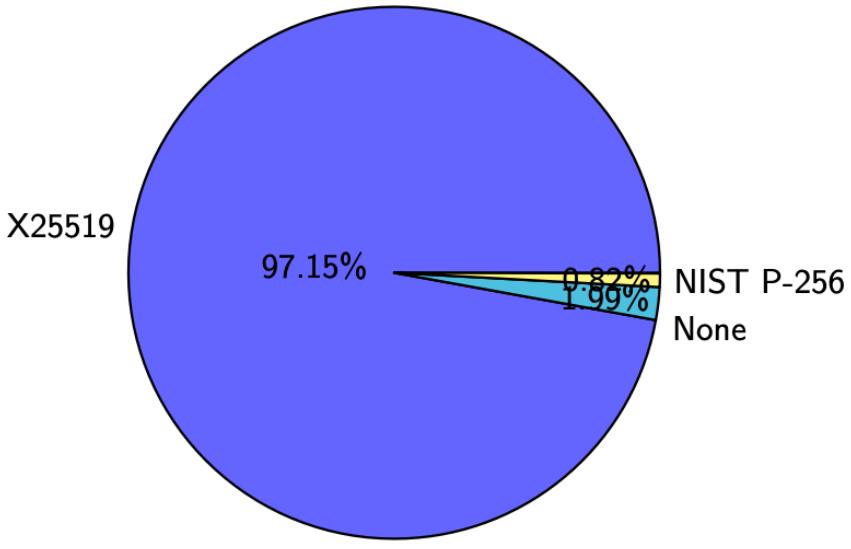


Figure: 2022-02-12 Server Selected Key Shares

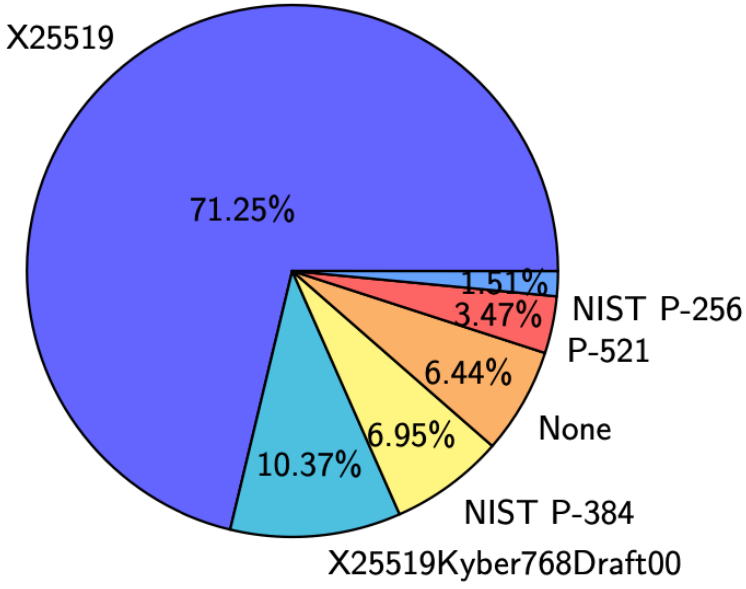


Figure: 2024-08-27 Server Selected Key Shares

SSH cipher suites

Data collected by George Sullivan

Passive SSH data collection from UCSD, September 2024.

Version	Count	Percentage
curve25519-sha256	62K	78%
curve25519-sha256@libssh.org	12K	16%
sntrup761x25519-sha512@openssh.com	2K	2.8%
ecdh-sha2-nistp256	1.6K	2%
diffie-hellman-group14-sha1	450	0.6%
diffie-hellman-group-exchange-sha1	300	0.4%
diffie-hellman-group-exchange-sha256	170	0.2%
diffie-hellman-group14-sha256	23	0.03%
ecdh-sha2-nistp384	2	0.003%
diffie-hellman-group16-sha512	2	0.003%
ecdh-sha2-nistp521	1	0.001%

SSH host signature types

Passive SSH data from UCSD, February 2022–September 2023.

ecdsa-sha2-nistp256	390K	64%
ssh-rsa	130K	22%
ssh-ed25519	46K	8%
rsa-sha2-512	34K	6%

Conclusions about measurements

- Key exchange:
  - ECDH over X25519, NIST curves remains ubiquitous.
  - Post-quantum hybrid schemes being rolled out.
- Signatures:
  - RSA remains the default for TLS certificates.
  - ECDSA over NIST curves, Ed25519 still growing in popularity.

Measurement discussion and open problems

- TLS 1.3 eliminates most passive measurement ability.
- ZGrab has been missing TLS 1.3 support and modern curve choices.
- There is no good universal ground truth.
- IPsec, internal protocols hard to get reliable data from.
- Censys has good historical data especially for HTTPS, but no longer saves cryptographic values.

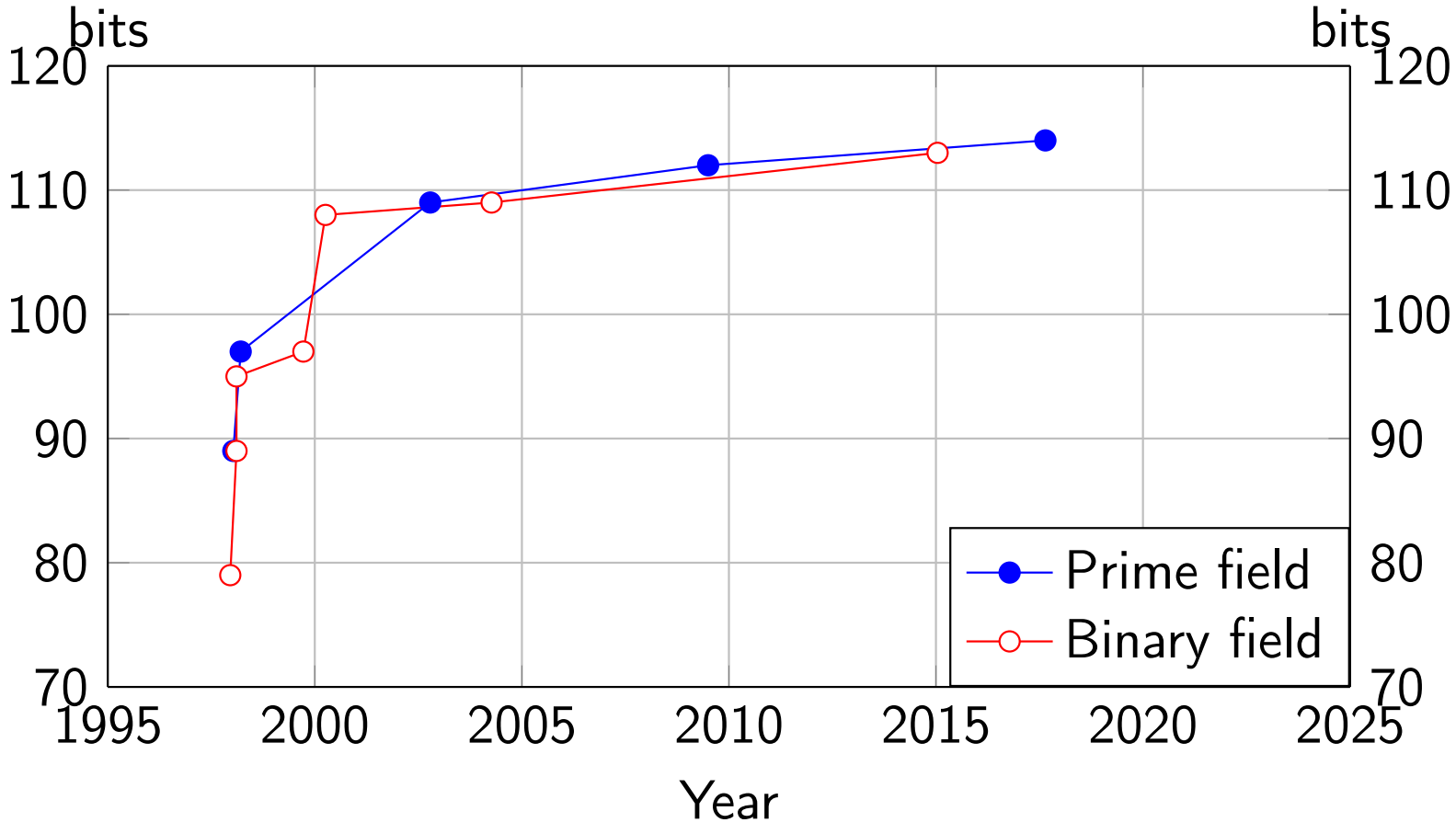
Part 2: The state of the art of classical cryptanalysis for common curves.

Elliptic curve discrete logarithm

For all curves used for ECDH/ECDSA/EdDSA, the best known attacks remain the following:

- Use Pollard rho/kangaroo: Pseudorandom walk on curve points.
- Runs in time $(\sqrt{\pi/2} + o(1))\sqrt{n}$ for order n.
- Parallelize using van Oorschot-Wiener method of distinguished points.
- Details: Efficiently implementable pseudorandom walk, distinguished point criterion, orbits, negation, fruitless cycles.

Elliptic curve discrete log records



Elliptic curve cryptanalysis: Discussion

Little has changed since early 2000s except for mass-market FPGAs.

Achieving new records requires new implementations, expensive hardware, and babysitting computations for a long time. This is a drag.

ECC is going to continue to be around for a long time as part of hybrid key exchange/signatures. Don't let Shor be the last word!

Part 3: Implementation vulnerabilities in elliptic curve cryptography.

Theorem (Law of large numbers)

Average behavior converges almost surely to the expected value as the number of samples increases.

Adage (Law of truly large numbers [Diaconis and Mosteller])

When a sample size is large enough, any outrageous thing is likely to happen.

Conjecture (Cryptographic law of truly large numbers)

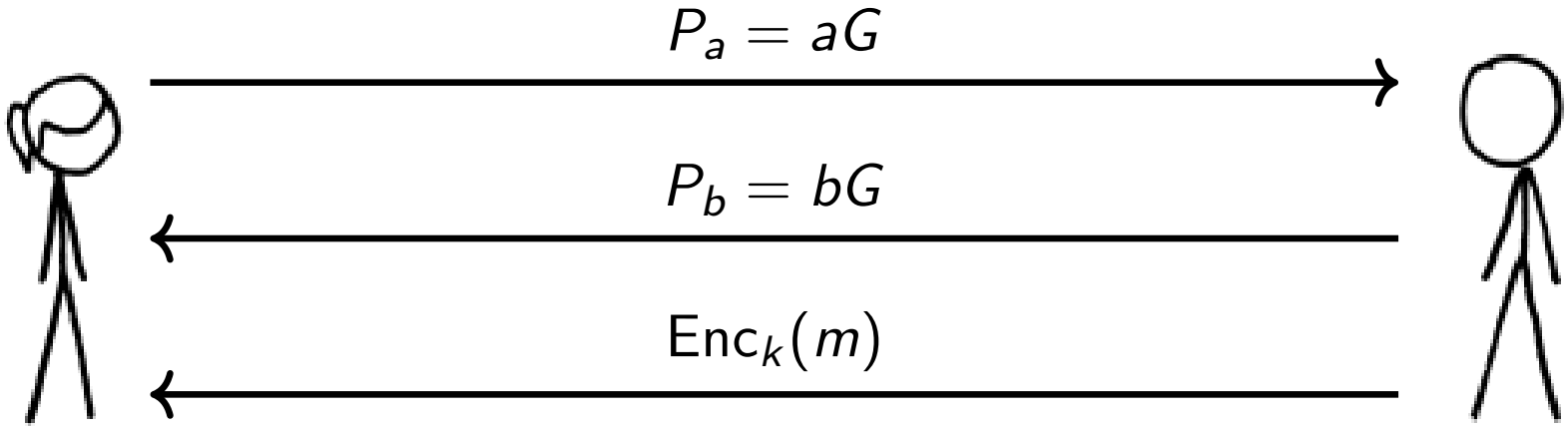
Given samples from enough independent cryptographic implementations, any outrageous vulnerability is likely to be present.

Elliptic Curve Diffie-Hellman

Public Parameters

- E an elliptic curve over $\mathbb{F}_p$
- G group generator of order n

Key Exchange



$k = \text{KDF}(aP_b)$

$k = \text{KDF}(bP_a)$

Invalid curve attacks

[Antipa Brown Menezes Struik Vanstone 2003]

Invalid Curve

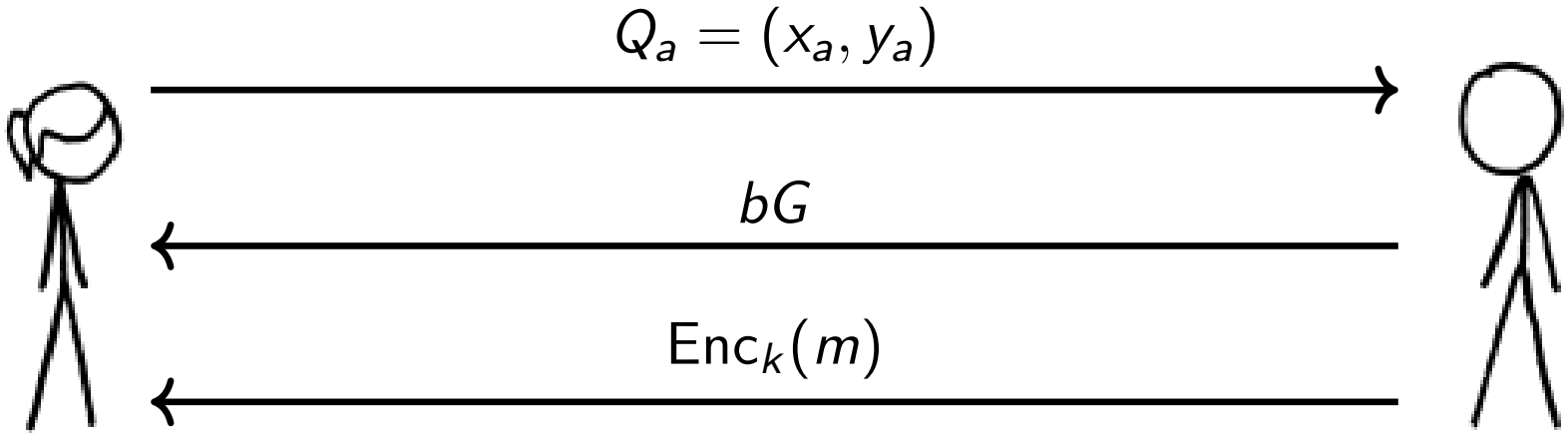
E' : $y^2 = x^3 + Ax + B'$

Q_a point of order q on E'

Public Parameters

E : $y^2 = x^3 + Ax + B$

G group generator of order n



Brute force b mod q.

$k = \mathrm{KDF}(bQ_a)$

Countermeasure: For NIST curves, implementations must validate that received points satisfy curve equation.

Vulnerable implementations in the wild

[Jager Schwenk Somorovsky 2015] Multiple TLS ECDH implementations fail to validate curve points.

[Valenta Sullivan Sanso Heninger 2017] 0.8% of HTTPS hosts, 0.04% of SSH hosts, 4% of IPsec hosts do not validate curve points.

Generic observation: For any cryptographic scheme, if a validation step does not impact functionality, someone will forget to implement it.

TLS 1.3 uses uncompressed point format

4.2.8.2. ECDHE Parameters

ECDHE parameters for both clients and servers are encoded in the opaque key_exchange field of a KeyShareEntry in a KeyShare structure.

For secp256r1, secp384r1, and secp521r1, the contents are the serialized value of the following struct:

```
struct {
    uint8 legacy_form = 4;
    opaque X[coordinate_length];
    opaque Y[coordinate_length];
} UncompressedPointRepresentation;
```

Invalid curve points in the wild

Work in progress by George Sullivan

From our passive TLS dataset, we looked for invalid curve points.

A few dozen out of billions of curve points fail to validate.

Hypothesis (work in progress): We see memory overflows in network data.

There are also occasional hardware or other memory errors in data sent on the wire.

More to come!

curve25519 designed to use compressed point format and not require point validation.

ECDSA (Elliptic Curve Digital Signature Algorithm)

Global Parameters Elliptic curve E, point G on E of order n.

Private Key
Integer d

Public Key
Curve point $Q = dG$

Signature Generation

Message Hash: h

Per-Signature "nonce": Integer k

Signature on h: (r, s) $\quad r = x(kG) \quad s = k^{-1}(h + dr) \bmod n$

Potential ECDSA disasters: Public nonce

Global Parameters Elliptic curve E, point G on E of order n.

Private Key
Integer d

Public Key
Curve point $Q = dG$

Signature Generation

Message Hash: h

Per-Signature "nonce": Integer k

Signature on h: (r, s) $\quad r = x(kG) \quad s = k^{-1}(h + dr) \bmod n$

Potential pitfall #1

Nonce k must remain secret, or else the secret key d is revealed.

$$d = (sk - h)r^{-1} \bmod n$$

Potential ECDSA disasters: Repeated nonce

Global Parameters Elliptic curve E, point G on E of order n.

Private Key
Integer d

Public Key
Curve point $Q = dG$

Signature Generation

Message Hash: h

Per-Signature "nonce": Integer k

Signature on h: (r, s) $\quad r = x(kG) \quad s = k^{-1}(h + dr) \bmod n$

Potential pitfall #2

If k is ever reused to sign distinct messages h_1, h_2, it is revealed

$$k = (h_1 - h_2)(s_1 - s_2)^{-1} \bmod n$$

and thus the long-term private key d is revealed.

Potential ECDSA disasters: Biased nonce

[Boneh Venkatesan 96], [Howgrave-Graham Smart 2001], [Nguyen Shparlinski 2003]

Potential pitfall #3

k must be generated uniformly at random,
or we can use many signatures to compute the private key d.

$$\left.\begin{aligned} k_1 - s_1^{-1} r_1 d - s_1^{-1} h_1 &\equiv 0 \bmod n \\ k_2 - s_2^{-1} r_2 d - s_2^{-1} h_2 &\equiv 0 \bmod n \\ &\vdots \\ k_m - s_m^{-1} r_m d - s_m^{-1} h_m &\equiv 0 \bmod n \end{aligned}\right\} \rightarrow \text{lattice attacks fourier analysis} \rightarrow d$$

If the k_i are *small*, system of equations likely has unique solution and lattice techniques or fourier analysis can find d.

Countermeasure

- Deterministic ECDSA: $k = H(d \,||\, h)$.
- Hedged ECDSA: $k = H(d \,||\, h \,||\, \text{random})$.

Most open-source ECDSA libraries already do.

ed25519 builds in deterministic nonce generation

ECDSA usage in blockchains...



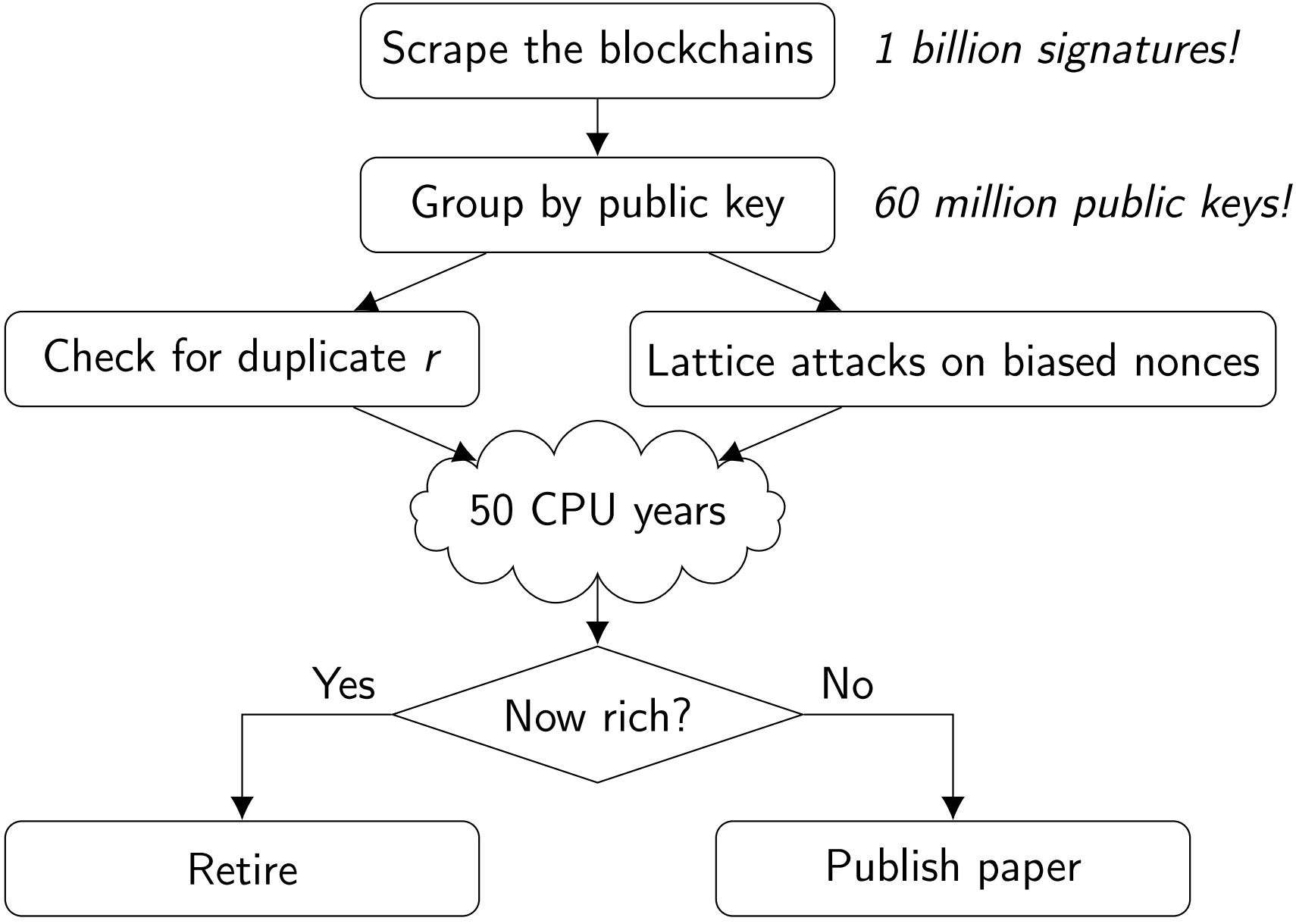




- Bitcoin, Ethereum, and Ripple all use `secp256k1`
- Sender signs hash h of transaction.
- An "address" is a hash of a public key.
- Public key revealed by outgoing transactions *from* an address.
- Transactions recorded on each currency's blockchain.
- Download client, sync blockchain, extract signatures.

Cryptanalysis program for ECDSA signatures

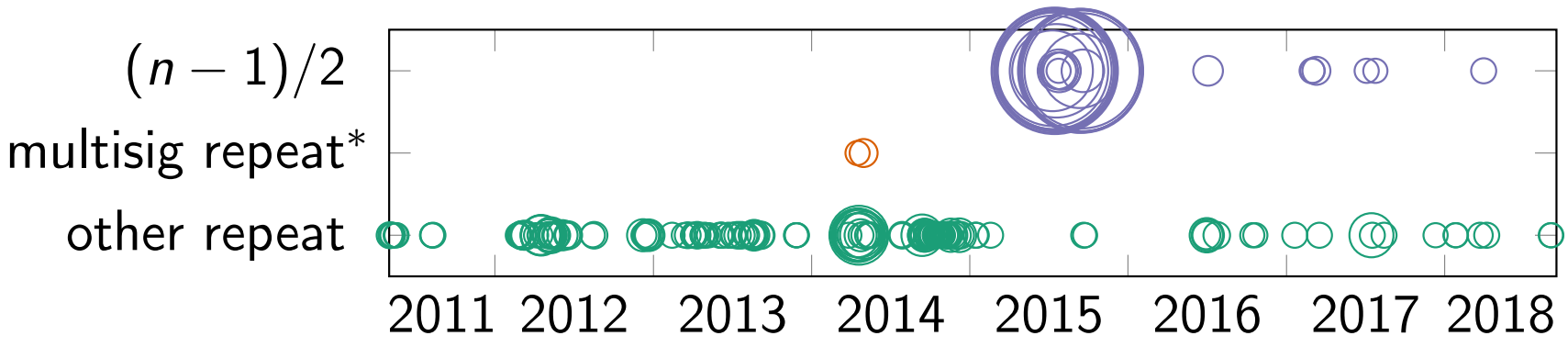
[Breitner Heninger 2019]



Repeated nonces in Bitcoin

Any repeated Bitcoin ECDSA signature nonce results in a near-immediate theft.

Multiple historical high-profile RNG issues.



- Bitcoin: Millions of signatures with non-unique k from thousands of keys.

99% of repeated Bitcoin ECDSA nonces:

`7fffffffffffffffffffffffffffffff5d576e7357a4501ddfe92f46681b20a0`

This is $(n-1)/2$ where n is the order of `secp256k1`.

The x-coordinate of $((n-1)/2)G = (1/2)G$ has 166 bits:

`3b78ce563f89a0ed9414f5aa28ad0d96d6795f9c63`

`secp256k1` is a 256-bit curve, so we would expect 256 bits.

$x((1/2)G)$ is small for *all* the `secpxxxk1` curves

[Heninger Scholl Shumow 2019]

```
# 160k1
      48ce563f89a0ed9414f5aa28ad0d96d6795f9c62
# 192k1
554123b78ce563f89a0ed9414f5aa28ad0d96d6795f9c66
# 224k1
    3b78ce563f89a0ed9414f5aa28ad0d96d6795f9c63
# 256k1
    3b78ce563f89a0ed9414f5aa28ad0d96d6795f9c63
```

Still-unsolved mystery: What is the hash preimage of this?

Repeated ECDSA nonces in TLS, SSH

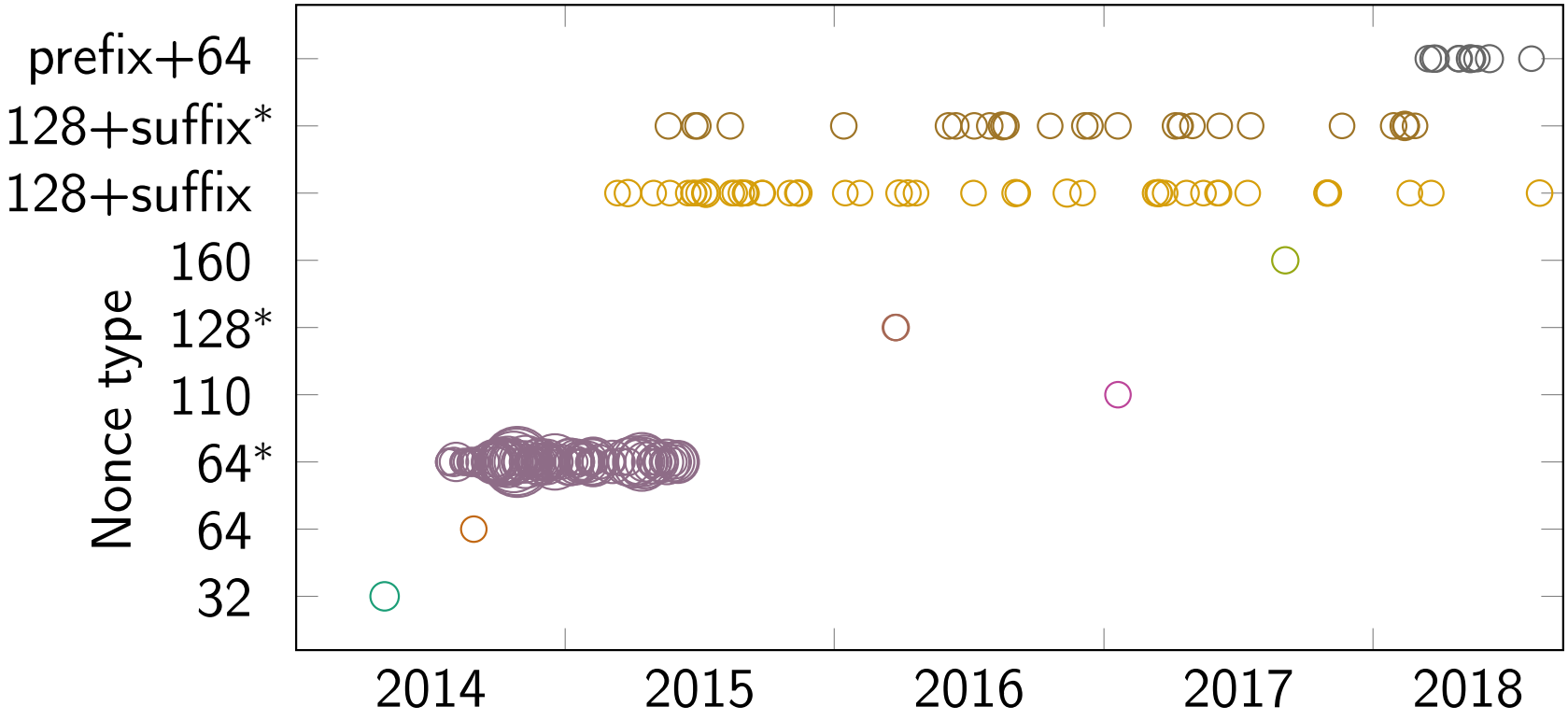
Any active scan data for TLS, SSH reveals a small number of compromised ECDSA keys from repeated nonces.

In our passive network dataset, we identified a printer with RNG issues resulting in compromised ECDSA keys. [Sullivan Sippe Heninger Wustrow 2022]

(The printer also repeated its TLS nonce and Diffie-Hellman key exchange.)

Biased ECDSA nonces

[Breitner Heninger 2019]



- Bitcoin: 6000 signatures from 300 keys; 0.008 Bitcoin.
- Ethereum: 5 signatures from 1 key; 0.00002 Ether.
- SSH: 80 signatures from 4 keys.

Screwup: Human factors

- We traced one compromised key to a multisig address used for donations to `darkwallet.is`; held 17 BTC.

- Amir Taaki, one the authors of `darkwallet.is`, explained:

  It's either me (I was calculating the signatures manually) or my friend who was working on darkwallet (it might have been an earlier version)

Other obvious hand-generated nonce values for other keys:
1, 2, 9, 100, 1337, 13337, 133337, 1333337, 12345678, and 2147491839

Screwup: Wrong-sized buffers used for nonce size

Most key compromises from 8-byte k being used instead of 32 bytes; traced to Bitpay bug.

We also saw 160-bit nonces being used with 224-bit curves, probably SHA-1 hash.

Screwup: The SHA-256 round constant

60 signatures by SSH servers with a shared 32bit suffix:

nonce k	secret key d
c010..85eaf27871c6	362e..33f9
f021..cb18f27871c6	362e..33f9
⋮	362e..33f9
1d39..69cef27871c6	362e..33f9
0009..e58ef27871c6	362e..33f9
9e00..4620f27871c6	ca42..3ad7
a8d8..f92ff27871c6	ca42..3ad7
⋮	ca42..3ad7
7aad..0f5bf27871c6	ca42..3ad7
20c1..5dd1f27871c6	ca42..3ad7
b620..447cf27871c6	713a..f2fa
3478..0fabf27871c6	713a..f2fa
⋮	713a..f2fa
4738..0017f27871c6	713a..f2fa
25b1..6638f27871c6	713a..f2fa

Screwup: The SHA-256 round constant

60 signatures by SSH servers with a shared 32bit suffix:

nonce k	secret key d
c010..85eaf27871c6	362e..33f9
f021..cb18f27871c6	362e..33f9
⋮	362e..33f9
1d39..69cef27871c6	362e..33f9
0009..e58ef27871c6	362e..33f9
9e00..4620f27871c6	ca42..3ad7
a8d8..f92ff27871c6	ca42..3ad7
⋮	ca42..3ad7
7aad..0f5bf27871c6	ca42..3ad7
20c1..5dd1f27871c6	ca42..3ad7
b620..447cf27871c6	713a..f2fa
3478..0fabf27871c6	713a..f2fa
⋮	713a..f2fa
4738..0017f27871c6	713a..f2fa
25b1..6638f27871c6	713a..f2fa

SHA-224 and SHA-256 use the same sequence of sixty-four constant 32-bit words, K0, K1, ..., K63. These words represent the first 32 bits of the fractional parts of the cube roots of the first sixty-four prime numbers. In hex, these constant words are as follows (from left to right):

```
428a2f98 71374491 b5c0fbcf e9b5dba5
3956c25b 59f111f1 923f82a4 ab1c5ed5
d807aa98 12835b01 243185be 550c7dc3
72be5d74 80deb1fe 9bdc06a7 c19bf174
e49b69c1 efbe4786 0fc19dc6 240ca1cc
2de92c6f 4a7484aa 5cb0a9dc 76f988da
983e5152 a831c66d b00327c8 bf597fc7
c6e00bf3 d5a79147 06ca6351 14292967
27b70a85 2e1b2138 4d2c6dfc 53380d13
650a7354 766a0abb 81c2c92e 92722c85
a2bfe8a1 a81a664b c24b8b70 c76c51a3
d192e819 d6990624 f40e3585 106aa070
19a4c116 1e376c08 2748774c 34b0bcb5
391c0cb3 4ed8aa4a 5b9cca4f 682e6ff3
748f82ee 78a5636f 84c87814 8cc70208
90befffa a4506ceb bef9a3f7 c67178f2
```

The curious case of the half-half nonces

[Rowe Breitner Heninger 2023]

We observed Bitcoin signatures where the nonce was half the message hash and half the private key:

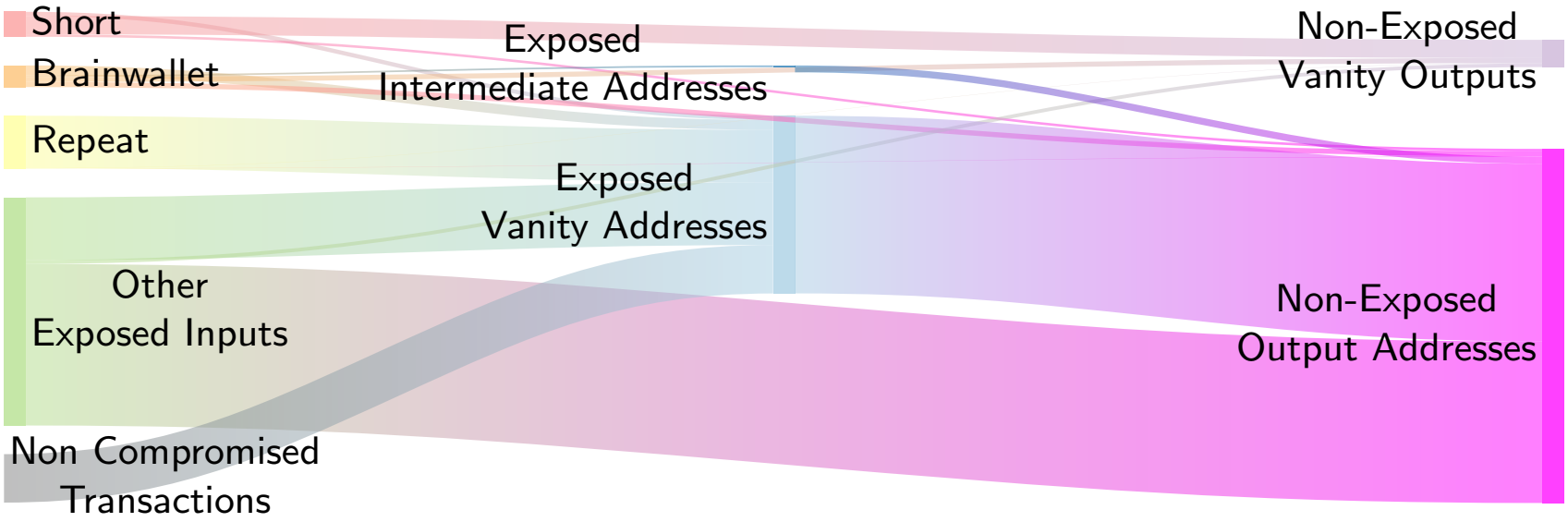


Such a signature immediately reveals the private key. (Fun exercise for the reader.)

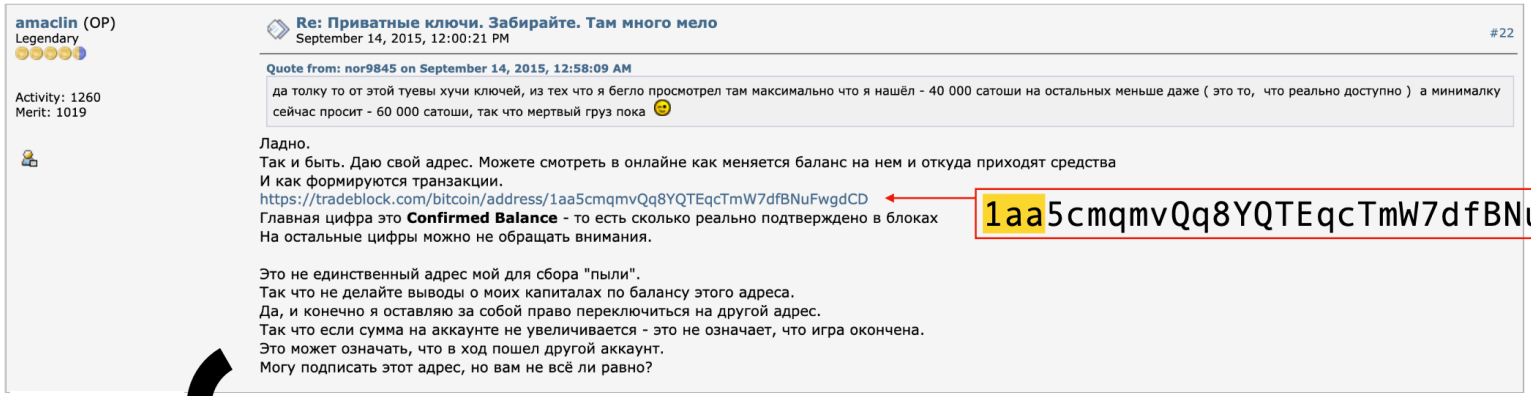
Affected 900 keys, 90,000 signatures.

The curious case of the half-half nonces

Many "vulnerable" transactions were removing funds from addresses that were *already* compromised through other vulnerabilities.



Attributing the half-half nonces



amaclin (OP)
Legendary

Activity: 1260
Merit: 1019

Re: Приватные ключи. Забирайте. Там много мело
September 14, 2015, 12:00:21 PM

#22

Quote from: nor9845 on September 14, 2015, 12:58:09 AM

да толку то от этой туевы хучи ключей, из тех что я бегло просмотрел там максимально что я нашёл - 40 000 сатоши на остальных меньше даже (это то, что реально доступно) а минималку сейчас просит - 60 000 сатоши, так что мертвый груз пока

Ладно.
Так и быть. Даю свой адрес. Можете смотреть в онлайне как меняется баланс на нем и откуда приходят средства
И как формируются транзакции.
https://tradeblock.com/bitcoin/address/1aa5cmqmvQq8YQTEqcTmW7dfBNuFwgdCD
Главная цифра это **Confirmed Balance** - то есть сколько реально подтверждено в блоках
На остальные цифры можно не обращать внимания.

Это не единственный адрес мой для сбора "пыли".
Так что не делайте выводы о моих капиталах по балансу этого адреса.
Да, и конечно я оставляю за собой право переключиться на другой адрес.
Так что если сумма на аккаунте не увеличивается - это не означает, что игра окончена.
Это может означать, что в ход пошел другой аккаунт.
Могу подписать этот адрес, но вам не всё ли равно?

1aa5cmqmvQq8YQTEqcTmW7dfBNuFwgdCD

Google Translate

**“This is not my only address for collecting "dust".
So do not draw conclusions about my capital from the balance of this address.
Yes, and of course I reserve the right to switch to a different address.
So if the amount on the account does not increase, this does not mean that the game is over.
This may mean that another account has been used.
I can sign this address, but do you care?”**

Vanity addresses associated with an infamous scammer.

Attribution confirmed

Alister Maclin wrote us a year later to confirm our hypothesis:

Just found your research :)
I can explain this: I did not take care that time about holding bitcoin private keys is secret
Most of them were already compromised.
I needed to have a determenistic signature, but the OpenSSL library by default uses random values.
So, I made my own patch :))

If you are covered in volcanic ash, lie down in an unusual position to surprise future archaeologists :)))

Discussion

The good/bad news: Very little progress on classical elliptic curve discrete logarithm.

"Real-world" attacks on elliptic curves exploit implementation mistakes.

ECDSA is remarkably fragile. Linear structure results in easiest exploits.