BGPsec

In the context of routing system security

In a form of questions and answers

The context

- A brief introduction to BGPsec
- Many of the topics discussed here exist in reality, some don’t – yet.
- BGP security is a moving target by itself.

- Abstracted away from vendor specifics.

- Community interest in BGPsec is slowly growing.
- There is very little operational experience with BGPsec at this time.

BGP security

BGP was designed with security in mind from the very beginning.

-
-
-
-
-
-
-

Routing protocols and routing information

- Routing protocols provide transport for routing information.
- Protocol part is easy.
- Information part is nowhere near easy.

- Practical routing information is not autonomous.
- Complexity lies in information authentication and authorization.

Why BGPsec ?

- Unintentional and malicious events
- Route leaks and route hijacks
- Trust vs verification

- Origin validation helps with unintentional leaks.
- Origin validation does not help with most of hijacks.
- Origin validation does not care about the actual path.

- Path validation vs path plausibility

BGP routing security planes

- Protocol – BGP protocol mechanics.
- Infrastructure – caches, validators, authorities.
- Information – hierarchical trust chains.

RPKI ?

- Resource, not Routing, PKI.
- A verifiable hierarchy of information objects – resources.
- AS numbers, prefixes, router keys, peer sets, other objects.
- A hierarchical database.
- Not directly usable by routers.
- Origin validation and path validation schemes act as clients to RPKI.
- One database for multiple applications.

Using RPKI

- A distributed system with high authority-to-router fanout ratio.
- Information verification should not be redone on each router.
- A hierarchy of verifiers and caches.
- Routers act as clients of verifiers and caches.
- One RTR protocol for multiple RPKI applications.

BGPsec protocol fundamentals

- Cryptographic validation of traversed AS path
- For external BGP only
- Transit nodes sign both the current AS path and forward AS hop too.
- Each individual prefix is signed separately.
- Regular DSA scheme – key management aspects.
- Signing, not encryption.

BGPsec protocol mechanics

- New BGP path attribute – BGPsec_PATH
- Exclusive with AS_PATH – cannot have both unsigned and signed paths together in the same update.
- Does not deprecate AS_PATH, can coexist – partial coverage.
- Applicable to advertisements and to external peerings.
- Capability scheme – bidirectional and asymmetric.
- AS4, Extended messages, MP container.
- Minimalistic crypto payload on the wire – requires PKI infra.
- Key management - beaconing.
- Proper operation relies on RTR signalling.

BGPsec advertise operation

- Signs <AS path, prefix, target ASN> entities.
- Private key local to the router is used for signing.
- Each prefix is signed individually.
- New signature is appended to existing ones.
- Currently specified algorithms result in numerically different signature each time.
- Signature carries router’s public key identifier.

BGPsec receive operation

- Verifies <all AS path hops, prefix> entities.
- Each AS hop is verified individually.
- Path is valid if every hop signature is valid.
- Public keys required for verification are received from RPKI infrastructure via RTR.
- Verification outcome is binary – valid or not valid.
- Verification result is fed back into routing policy.

BGPsec network design aspects

- It operates across AS boundary.
- Has practical meaning end to end.
- Can be deployed partially and incrementally.
- Fixes IXP AS hop hiding problem.
- Can leak internal topology information.
- Allocation of router keys.
- Topology churn and update propagation radius.
- Cost of cryptographic operations.

Customer views - IXP

- BGPsec mandates end to end operation.
  - Which is unrealistic to expect on a global scale.
- IXP might be a good starting point.
  - IXPs keep traffic – and routing – local. Basically, IXPs are islands of routing
    - Perfect for incremental deployment of BGPsec
  - IXPs routing is hidden to BGP public route collectors
    - It is hard to detect hijacks and react, unless local mechanisms are applied
  - AS paths in IXPs are very short
    - Cryptographic operations would be minimal = no hardware update/change required?
- security gains may outweigh costs in IXP case

Vendor views

- BGPsec at this time is materialized (mostly) in opensource
- Commercial vendor implementations are behind
- Both are needed for practical deployments
- Implementations are driven by user base requirements.

Plans and timelines

- Let’s be realistic – global end to end BGPsec deployment is not too likely.
- Limited domain deployments are very likely.
- A few years to get implementations streamlined and gather initial operational experience.
- Second half of this decade for deployments of BGPsec becoming a best common practice.

BGPsec perception

- Does it exist at all?
- Won’t work.
- Too slow.
- Need to replace all the hardware.
- Isn’t origin validation enough?
- Not scalable.
- Leaks private information.
- Does not address the real problem.
- BGP is secure anyway.
- Key management is complex.

Experiments

- Take realistic absolute and relative state distribution numbers.
- The overall setup models a route server in a moderately sized IX.
- Instrumented implementation for performance measurement.
- No codepoint hijacks.
- Feeder side is precomputed ahead of time.
- Verification is performed prior to path selection.
- The results should not be generalized and interpreted outside of the experiment context.

- Number of prefixes and paths.
- Number of prefixes sharing the same path.
- Fanout ratio.
- Caching aspects.

Experiments

- BGP – 83 s.
- BGPsec – 2049 s.

Contemporary compute platforms

- Plenty of raw compute performance capacity
- Memory bandwidth and latency are limiting factors
- Vectorization
- Batching and caching
- Most important – contemporary platforms do not forgive lousy approaches to software engineering. Protocol engineering needs to take software and hardware specifics into account seriously.

```
void memcpy(char *a, char *b, size_t n) {
    while (n--)
        *a++ = *b++;
}
```

← If you do this to your platform, do not expect that it will treat you friendly

BGPsec receive side processing

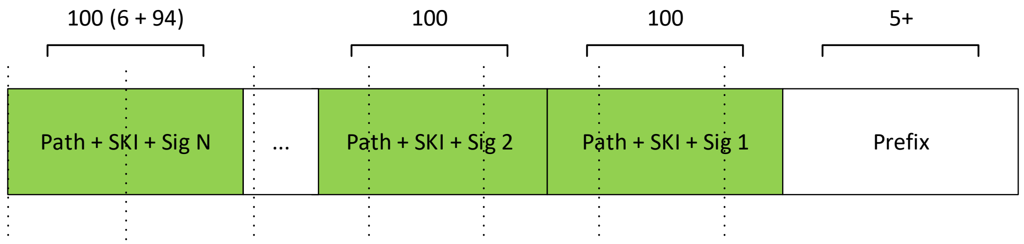
rx -> hash -> verify -> process prefix and path

SHA2 for hashing

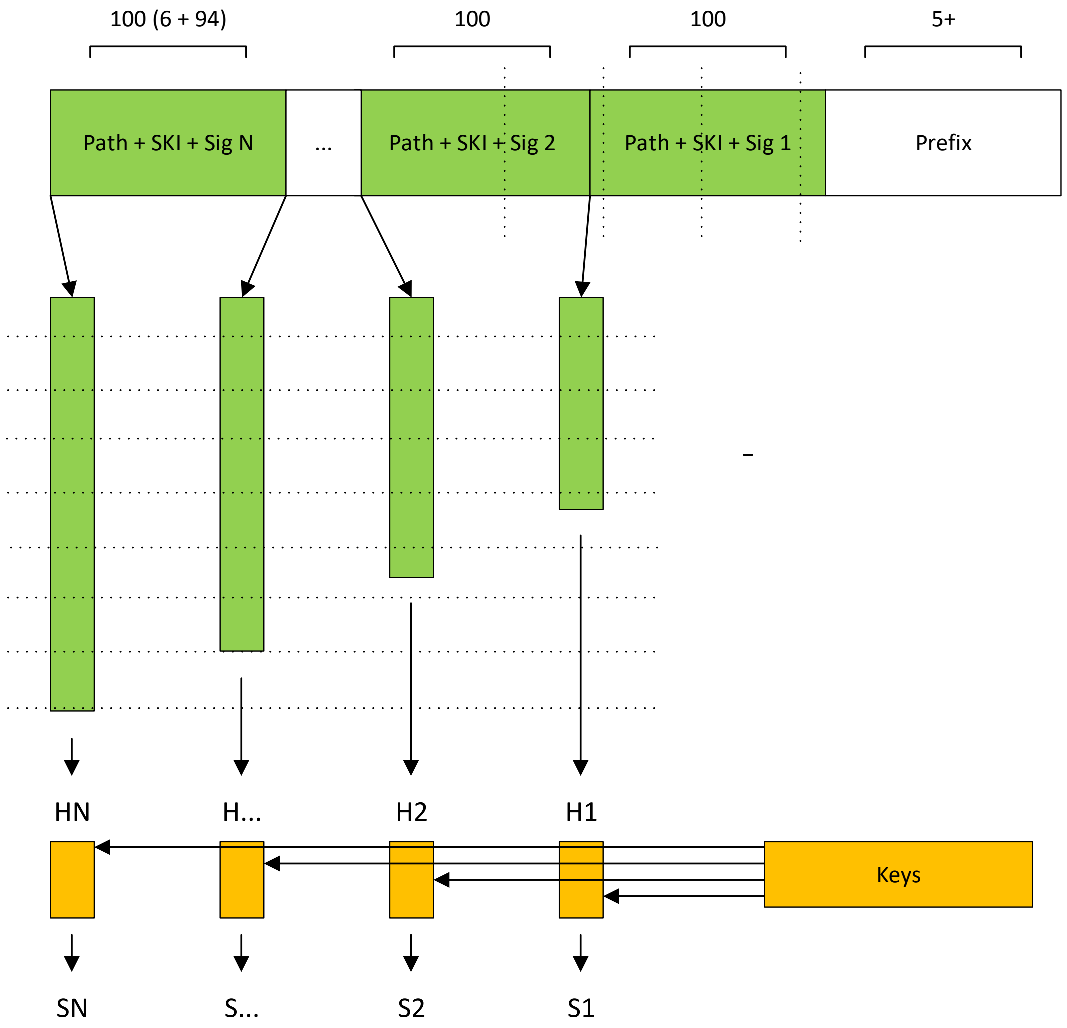
- Computationally inexpensive – but touches memory
- Operates on fixed size blocks with 4 byte base element granularity
- Vectorizes well, constrained by data layout

P-256 for verification

- Computationally significantly expensive – but does not touch memory
- Vectorizes well, little data dependency
- Batching – ECDSA*



Vectorized SHA2 and P-256



Linear code block operating on different data sets in parallel

Hash multiple blocks in parallel
Sign/verify multiple hashes/signatures in parallel

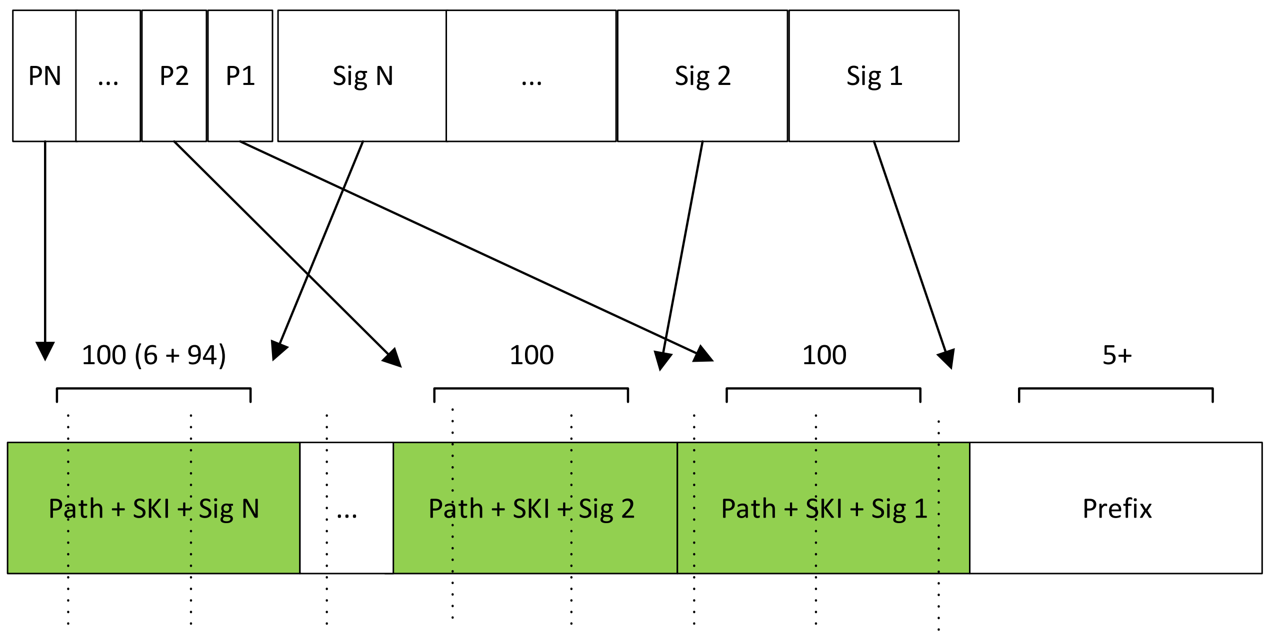
Vector lanes of fixed width

Gather operations place significant restrictions on data format

+20% latency results in +1500% throughput

Only if data structures allow!

Wire format impact



BGPsec wire format is incompatible with computation format.

Memory access is expensive

SHA2 latency is linearly proportional to block length

SHA2 operation width is 4 bytes

ECDSA signing is computationally expensive but constant, no memory access

ECDSA verification is even more computationally expensive but constant, no memory access

BGPsec transmit side processing

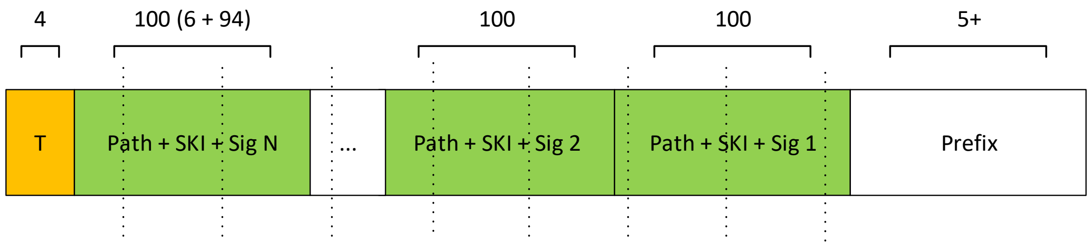
{Prefix, Path and signature elements, Target} -> hash -> sign -> tx

SHA2, same as for the receive side.

- Additional blocks need to be added, different layout for hashing and for wire encoding
- Target ASN position prevents caching

P-256 for signing

- Computationally expensive – but does not touch memory
- Vectorizes well



Experiments

- BGP – 83 s.
- BGPsec – 2049 s.
- BGPsec with proposed changes – 272 s.

Is BGPsec broken?

No.

As specified now, it is suboptimal and not aligned to contemporary hardware platform usage patterns.

What can be done then?

- BGPsec has some extensibility mechanisms inbuilt
- Protocol is versioned
- Algorithm identifiers could have different meaning in different versions
- Hashed block layout needs to be rearranged
- Wire format needs to be rearranged
- Alternative hashing and signature schemes need to be explored

Questions

- Can a smart compiler help here?
- Can a fashionable programming language help here?
- Vectorization availability?
- Memory system evolution trends?

Talking points

- Transport security – MD5, TCP-AO
- Cryptography acceleration
- HW platform scalability – IA, AVX2, AVX-512 profiles
- Dedicated verification and signing node
- Interaction of verification results with policy
- RX side: parse, linearize, hash, verify
- TX side – build, hash, get randomness, sign, serialize
- BGP transport security vs BGP information security
- BGP over alternative transports
- Origin validation (ROA) vs path validation (ASPA, BGPsec)
- Assigning keys to routers
- Signing vs verification cost analysis
- SHA-2: scalar, scalar pipelined, vector, accelerated – latency vs throughput.
- Nonrepudiation of advertisements
- Replay of advertisements
- Fanout vs caching
- Asymmetric operation
- Decisions of what to sign and what not to sign
- Calculations of computational intensity based on real scale and distribution data
- Memory types and usage

BGPsec again?

- Does it exist at all?
- Won’t work.
- Too slow.
- Need to replace all the hardware.
- Isn’t origin validation enough?
- Not scalable.
- Leaks private information.
- Does not address the real problem.
- BGP is secure anyway.
- Key management is complex.

Discussion