RSA CONFERENCE 2014
FEBRUARY 24 – 28 | MOSCONE CENTER | SAN FRANCISCO

Share.
Learn.
Secure.
Capitalizing on
Collective Intelligence

Running Secure Server Software on Insecure Hardware without a Parachute

SESSION ID: STU-M06B

Nicholas Sullivan

Systems Engineer
CloudFlare
@grittygrease

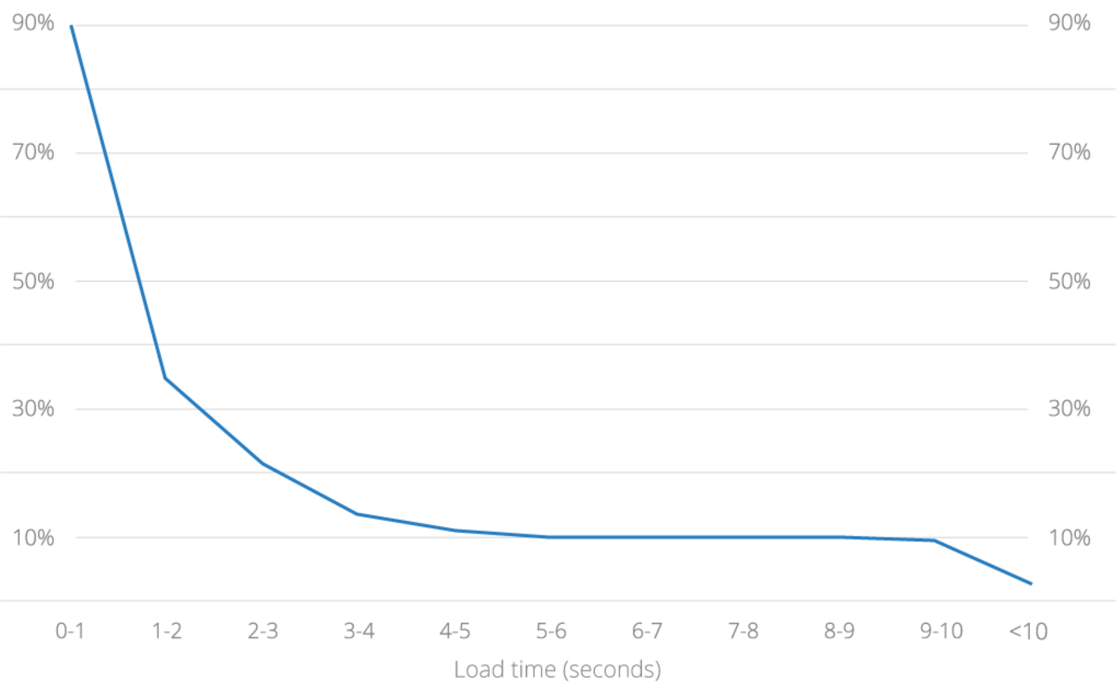
What this talk is about

- The web is changing — consolidation at the edge
- Fundamental assumptions about server security are wrong
- How do we design server software with the worst case in mind?
  - Distinguish between long and short term secrets
  - Devise approaches for protecting each

Let’s Talk About Web Infrastructure

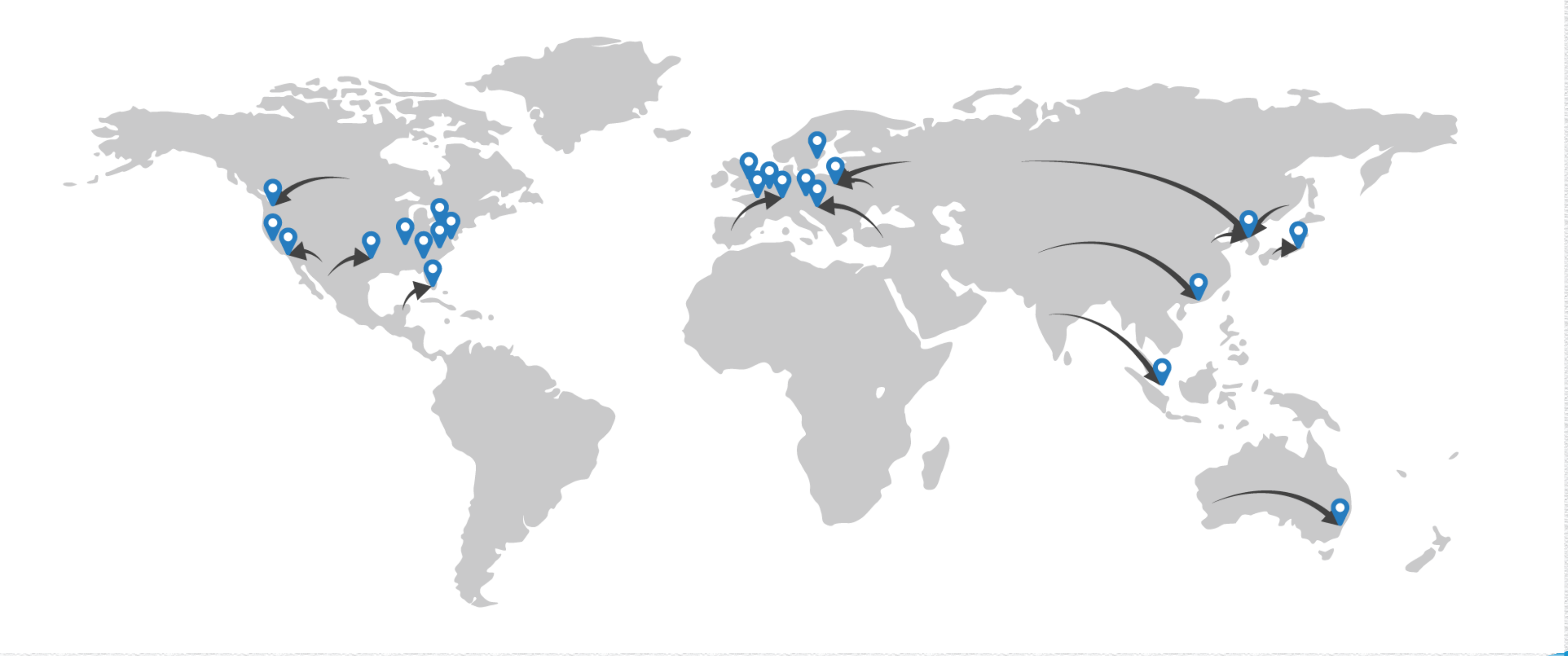
Conversion rate vs. load time

Load time in seconds against conversion rate percentage

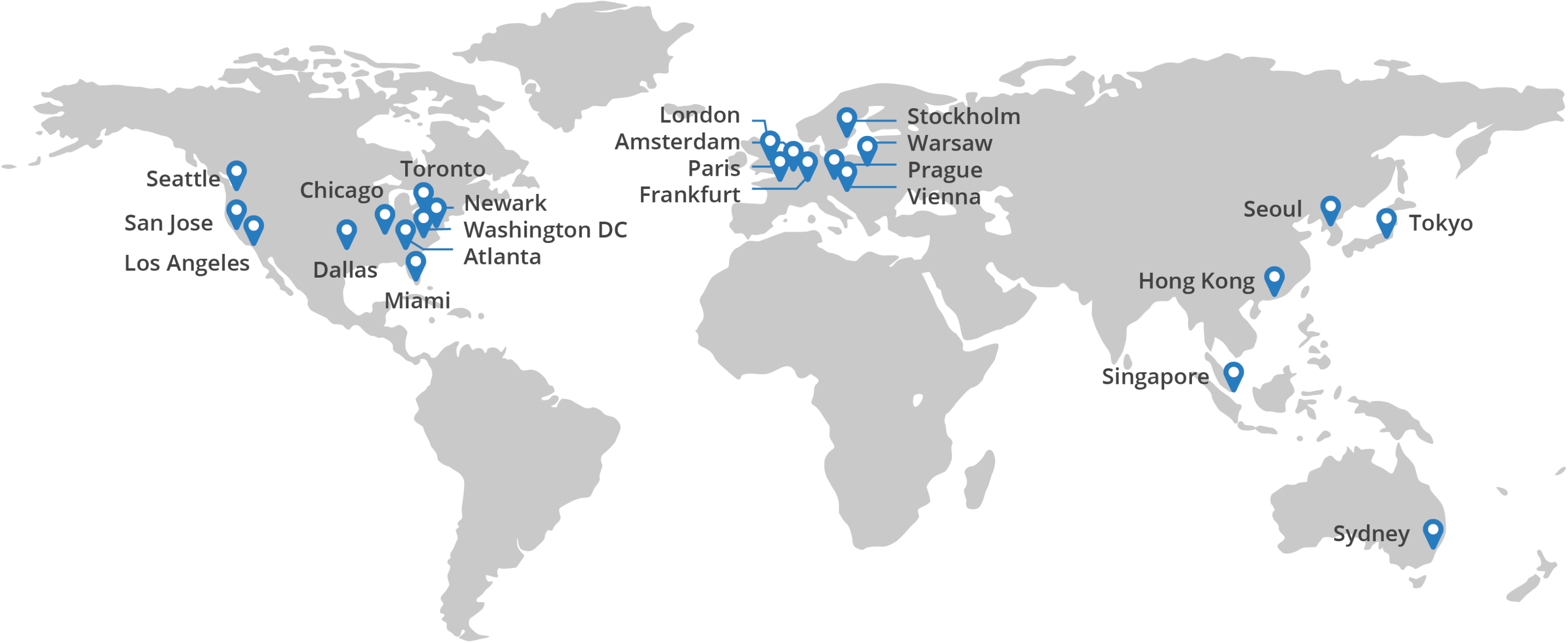


Global Website Traffic

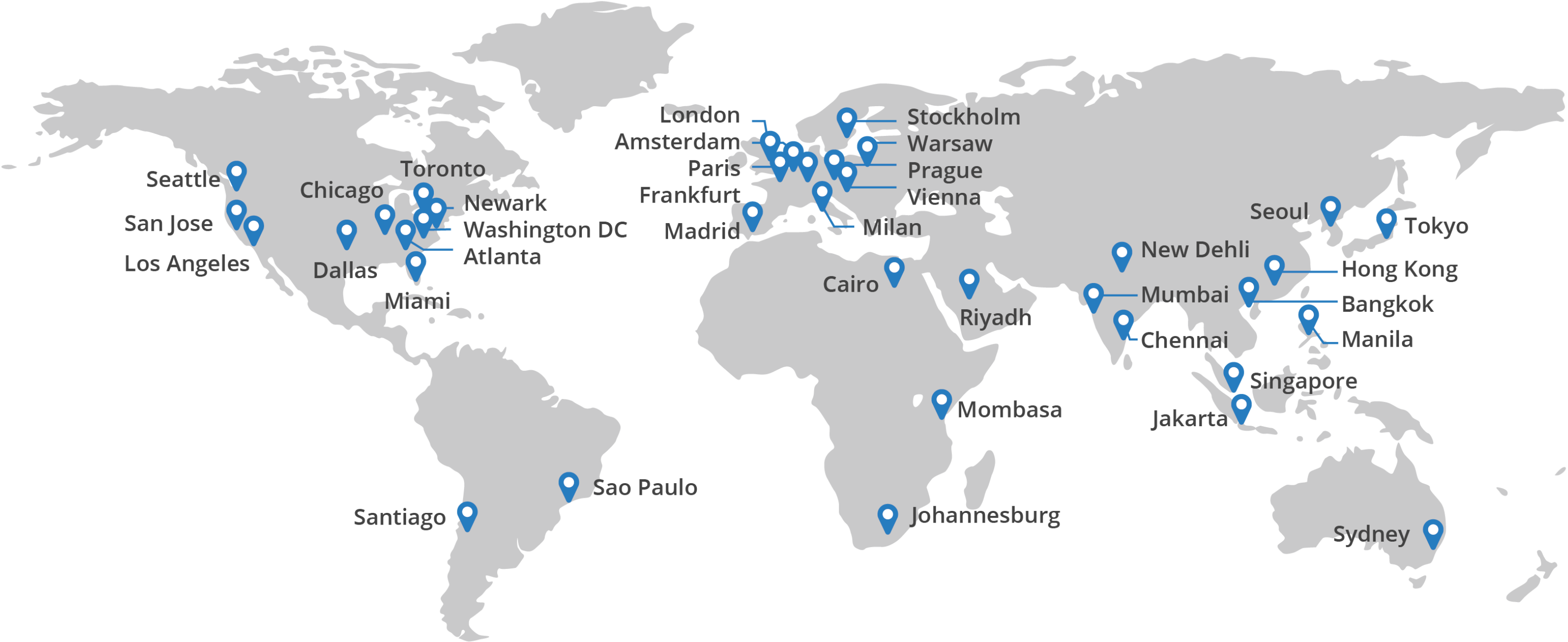
Global Website Traffic with CDN



Current Map


Seattle
San Jose
Los Angeles
Chicago
Dallas
Toronto
Newark
Washington DC
Atlanta
Miami
London
Amsterdam
Paris
Frankfurt
Stockholm
Warsaw
Prague
Vienna
Seoul
Tokyo
Hong Kong
Singapore
Sydney

Future Map

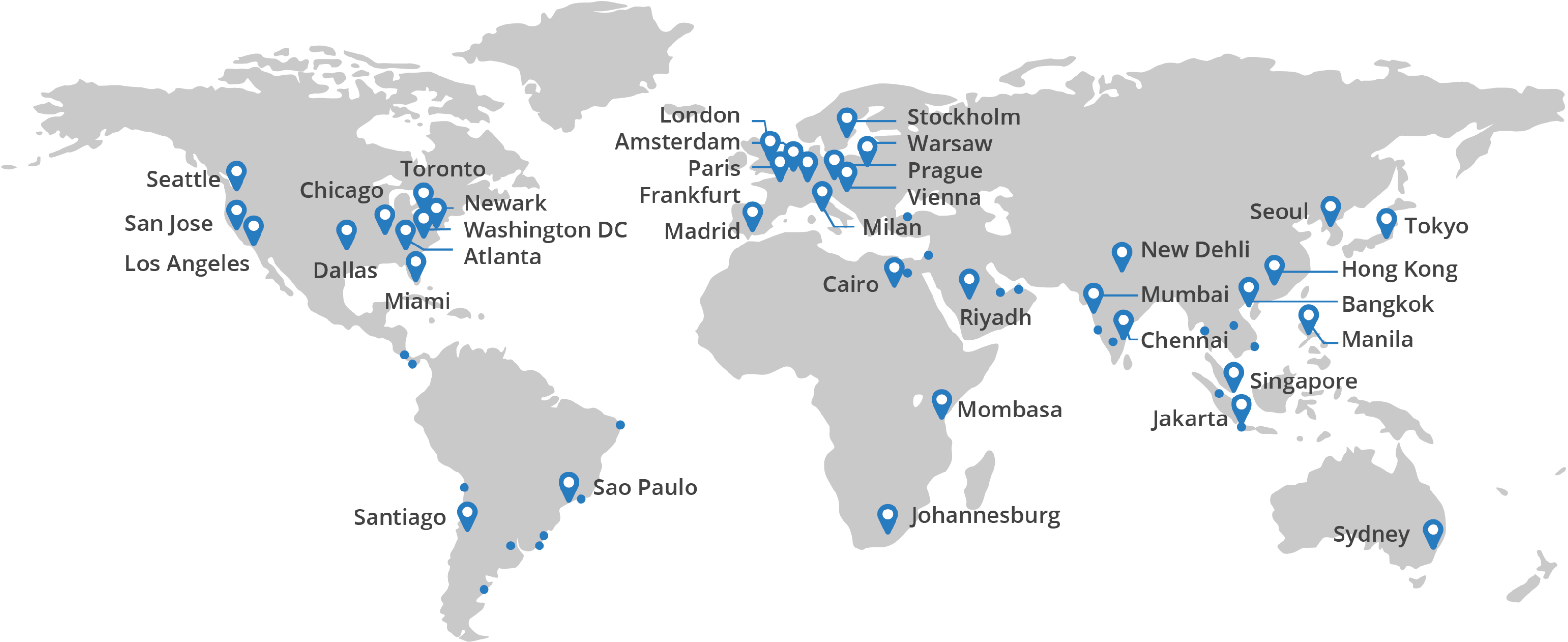


CLOUDFLARE

#RSAC

RSA CONFERENCE 2014

Future Map


Seattle
San Jose
Los Angeles
Chicago
Dallas
Toronto
Newark
Washington DC
Atlanta
Miami
London
Amsterdam
Paris
Frankfurt
Madrid
Stockholm
Warsaw
Prague
Vienna
Milan
Cairo
Riyadh
Mombasa
Johannesburg
Sao Paulo
Santiago
New Dehli
Mumbai
Chennai
Seoul
Tokyo
Hong Kong
Bangkok
Manila
Singapore
Jakarta
Sydney

RSACONFERENCE2014

FEBRUARY 24 – 28 | MOSCONE CENTER | SAN FRANCISCO

Edge Computing Threat Model

Traditional server threat model

- Assume server is secure
- Add layers of protection to keep attackers out
  - Network layer protection
  - Operating System Level: principle of least privilege
  - Protection against maliciously installed code
  - More advanced barriers

Globally distributed servers

- Less jurisdictional control = less physical security
- Physical access trumps static defense layers

- Traditional defenses helpful, but not ideal
  - Cannot rely on security of keys
  - Single break-in results in immediate compromise

A More Effective Approach

Approach system security the ‘DRM way’

- Assume attacker has bypassed all static defenses
- Goal is to refresh secrets they are compromised
- Split system into long-term secrets and short-term secrets
- Focus on renewability of secrets

Secrets must be split into two tiers

- Long-term Secrets
  - Useful for attacker for long period of time
  - Do **not** store at the edge

- Short-term Secrets
  - Expire after a short period of time
  - Cannot be re-used

Example: Traditional TLS termination

- TLS handshake with nginx and Apache
  - SSL keys on disk
  - Read from disk, use in memory

- Cryptographic elements at risk if server is compromised
  - Private key
  - Session key

TLS revisited for untrusted hardware

- Long term secrets
  - Private key
- Short term secrets
  - Session key
  - Session IDs and Session ticket keys
  - Credentials to access private keys

How to Protect Short-term Secrets

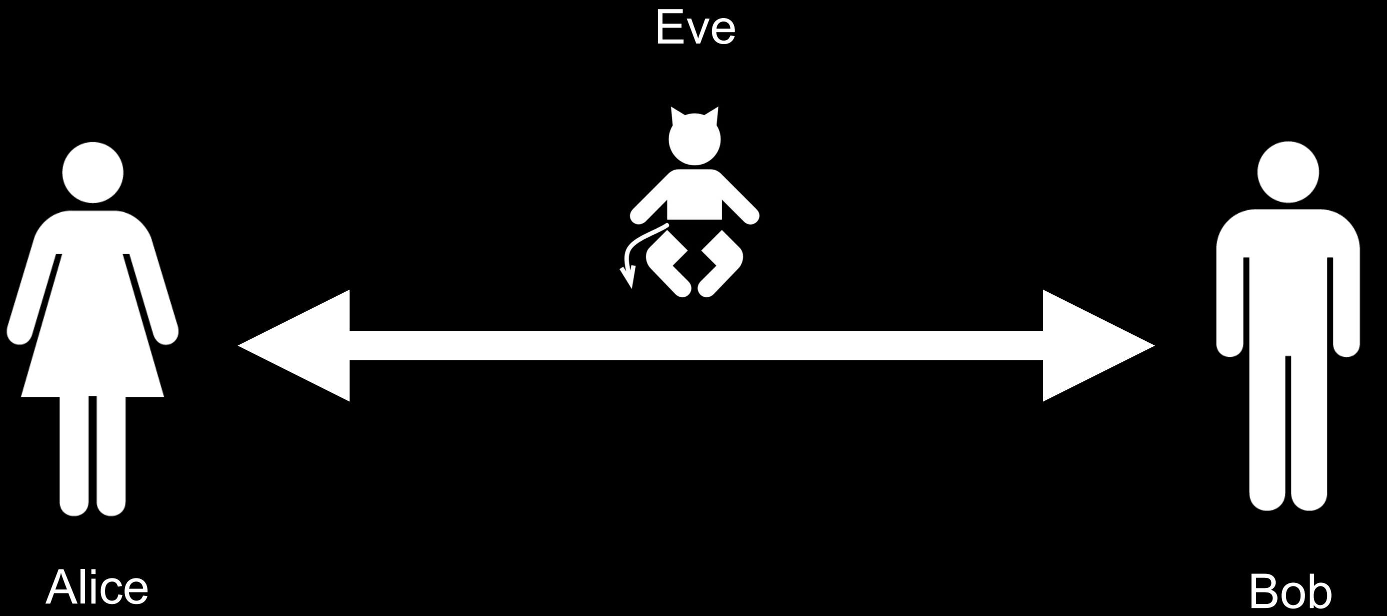
Short-term secrets — threat model

- Must live on machines in unsafe locations
  - Memory
  - Control Flow
- By the time a secret is broken, it should be expired
  - Don’t keep secrets in a useable state
  - Impose computational cost to retrieve the original secret
  - Expire secrets quickly

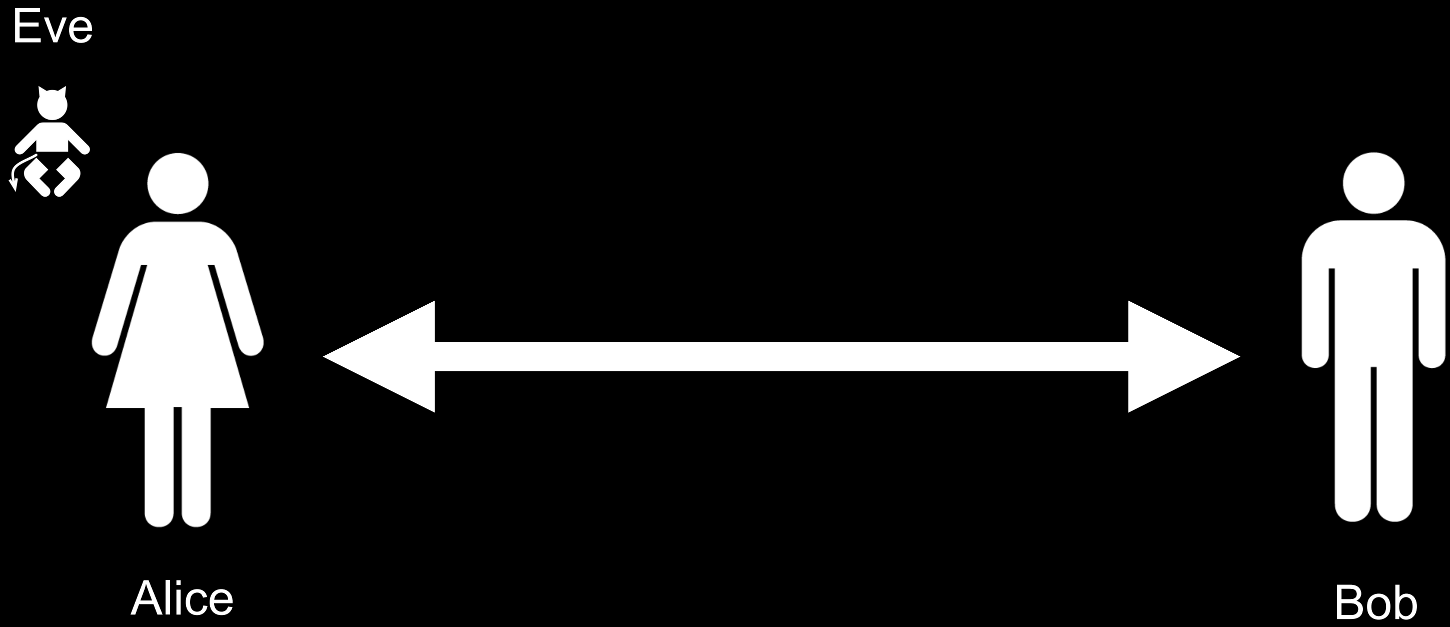
Techniques from DRM are applicable

- White-box cryptography
- Code obfuscation

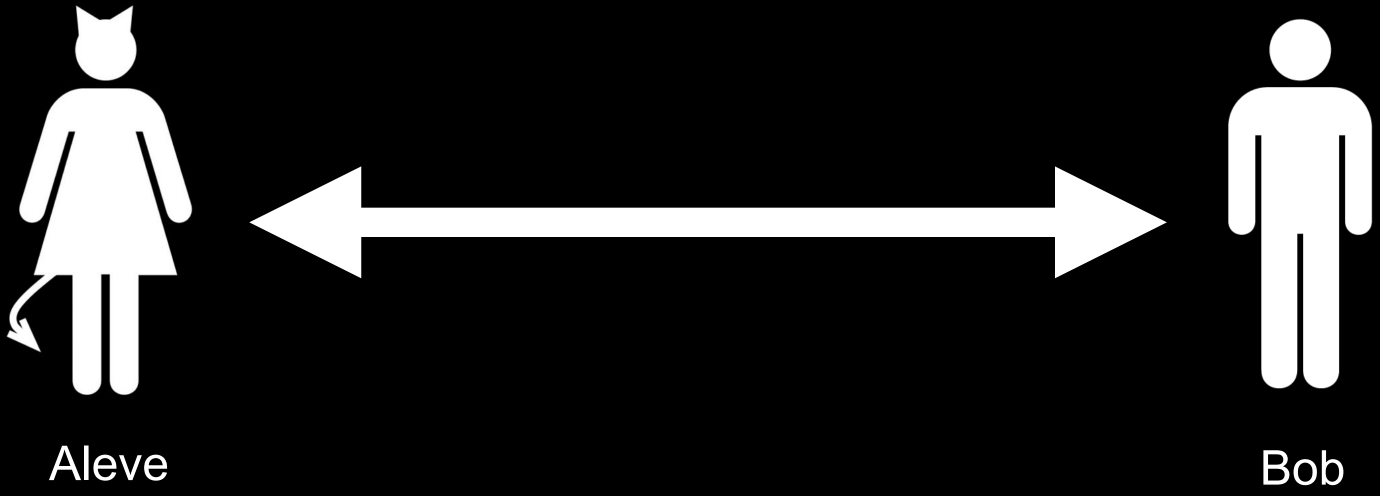
Standard Cryptography Threat Model



White-box Cryptography Threat Model



White-box Cryptography Threat Model



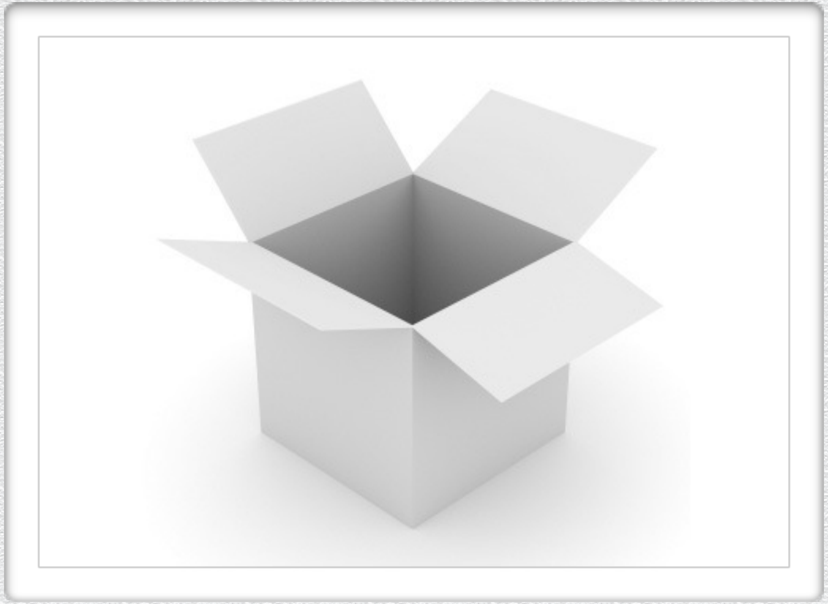
White-box cryptography

- Hide the cryptographic key from everyone
- Protect against **key extraction** in the strongest threat model

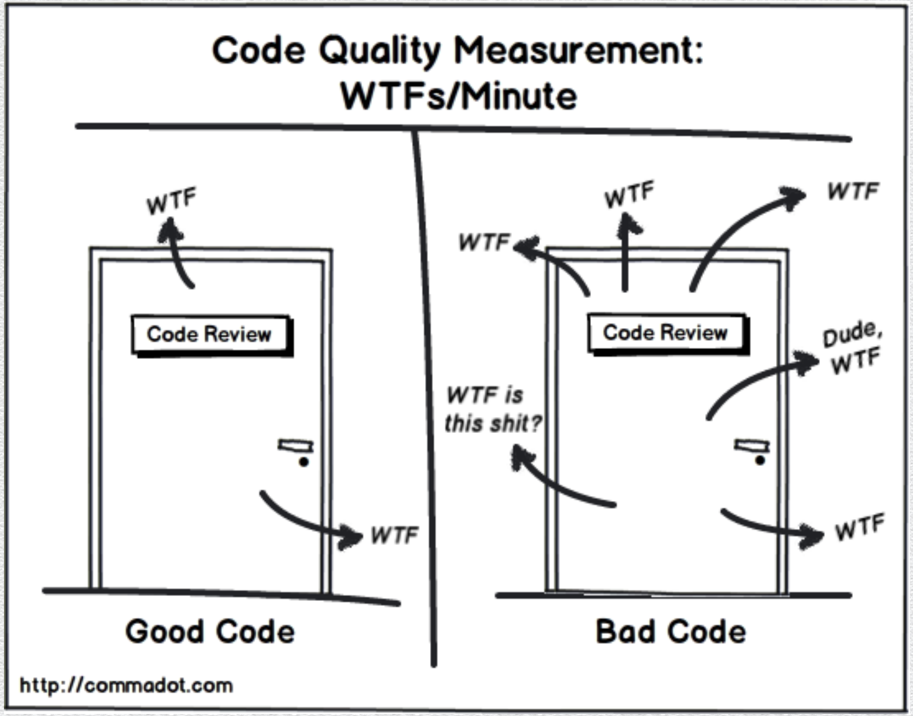
- Takes time to extract key — lots of math
- Choose difficulty based on secret lifetime

White-box cryptography implementations

- Commercial products
  - Irdeto, Arxan, SafeNet, etc.
- Open source
  - OpenWhiteBox



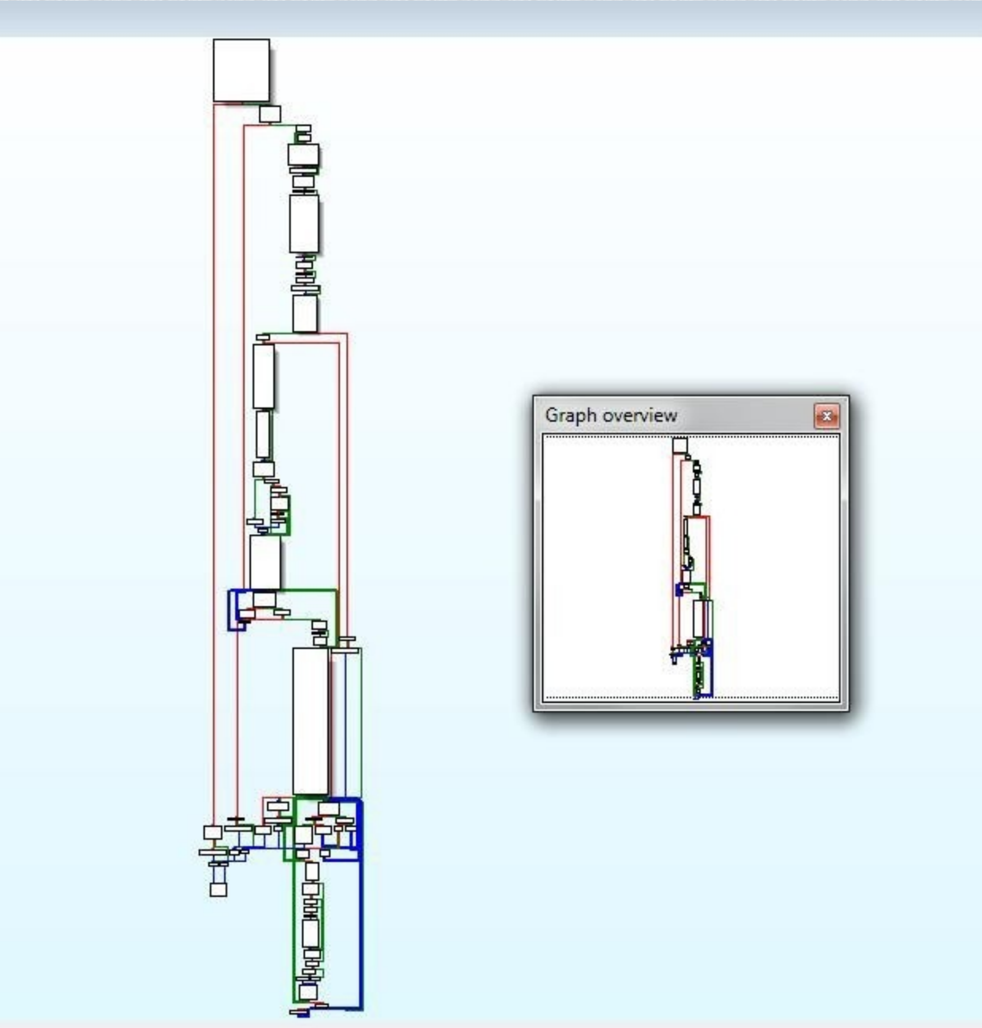
Code obfuscation



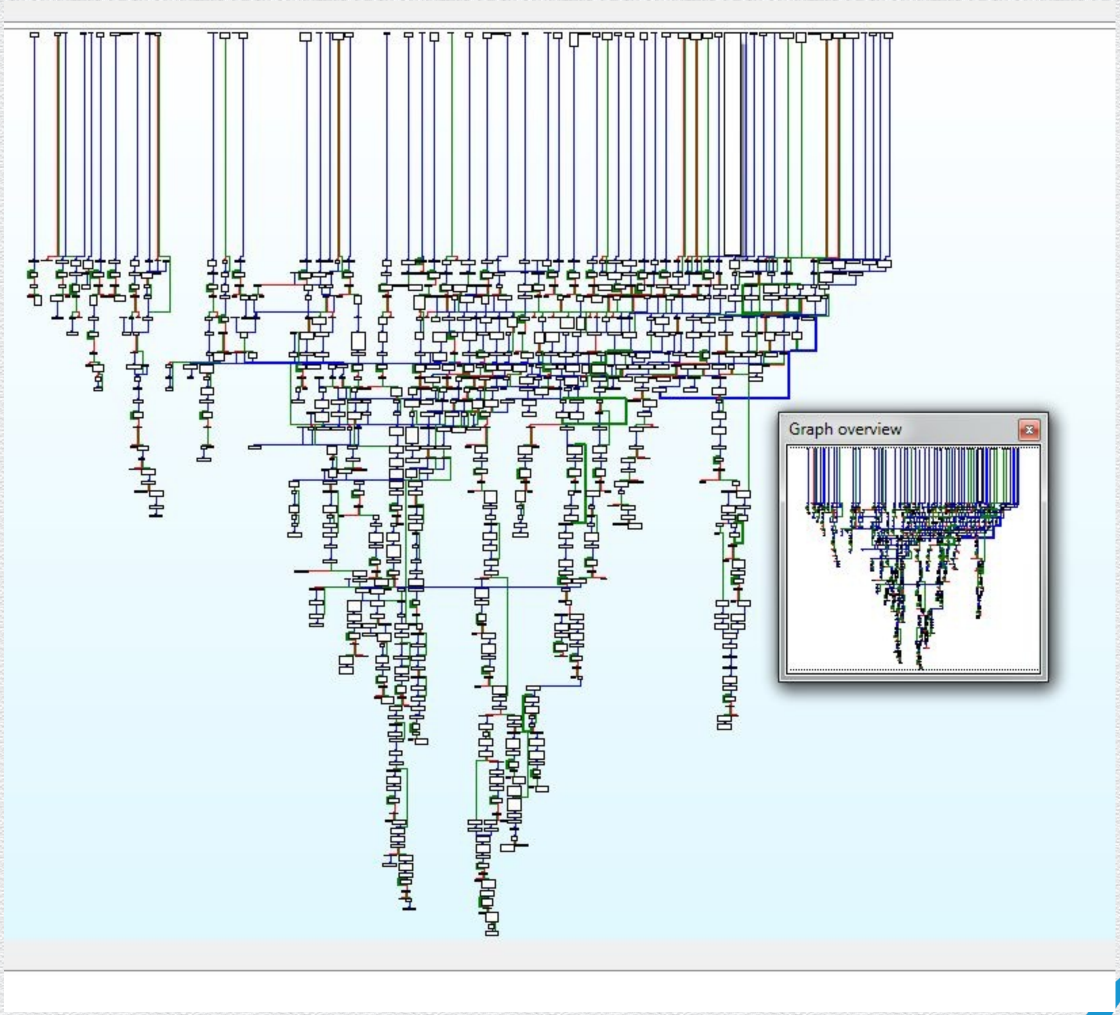
Code obfuscation

- Making reverse engineering difficult
  - Compile-time control-flow modification
  - Data transformation in memory
  - Anti-debugging

Before



After



Code obfuscation implementations

- Commercial products
  - Arxan, Irdeto, etc.
- Open source
  - Obfuscator-LLVM

RSA CONFERENCE 2014

FEBRUARY 24 – 28 | MOSCONE CENTER | SAN FRANCISCO

Long-term Secrets

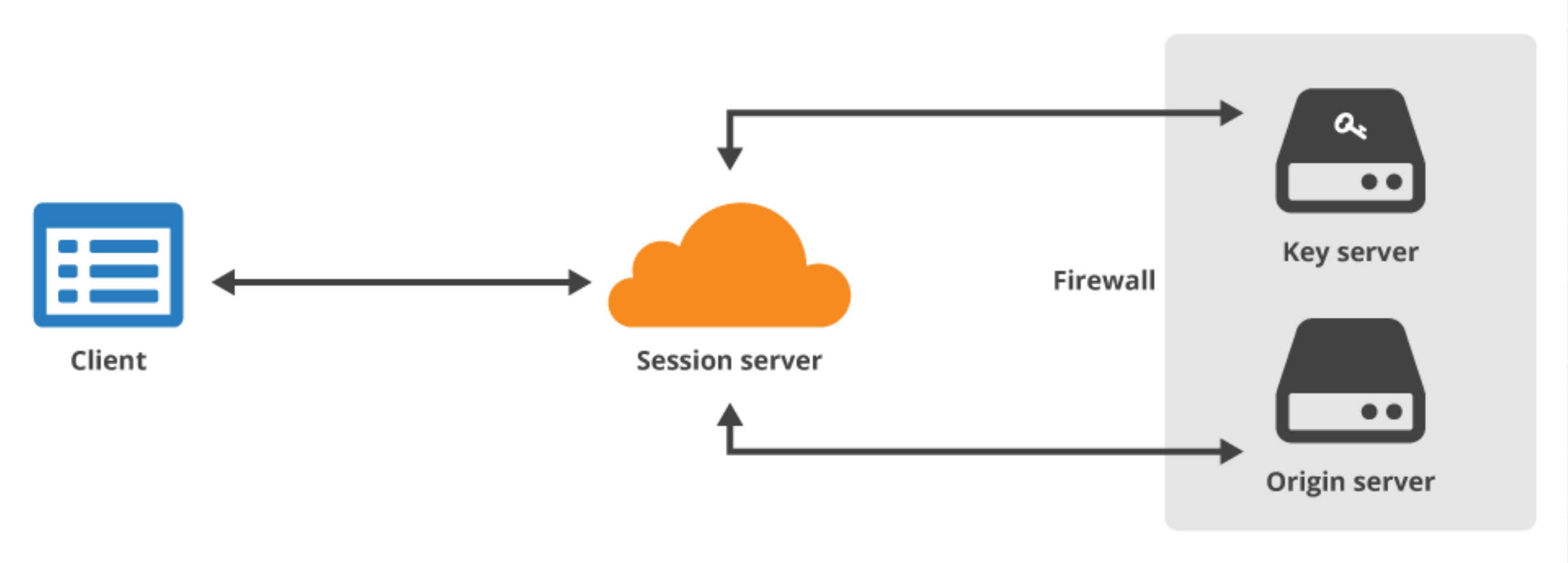
Keyless SSL

- SSL without keys? Surely you're joking.
- SSL without keys *at the edge*. That's better.

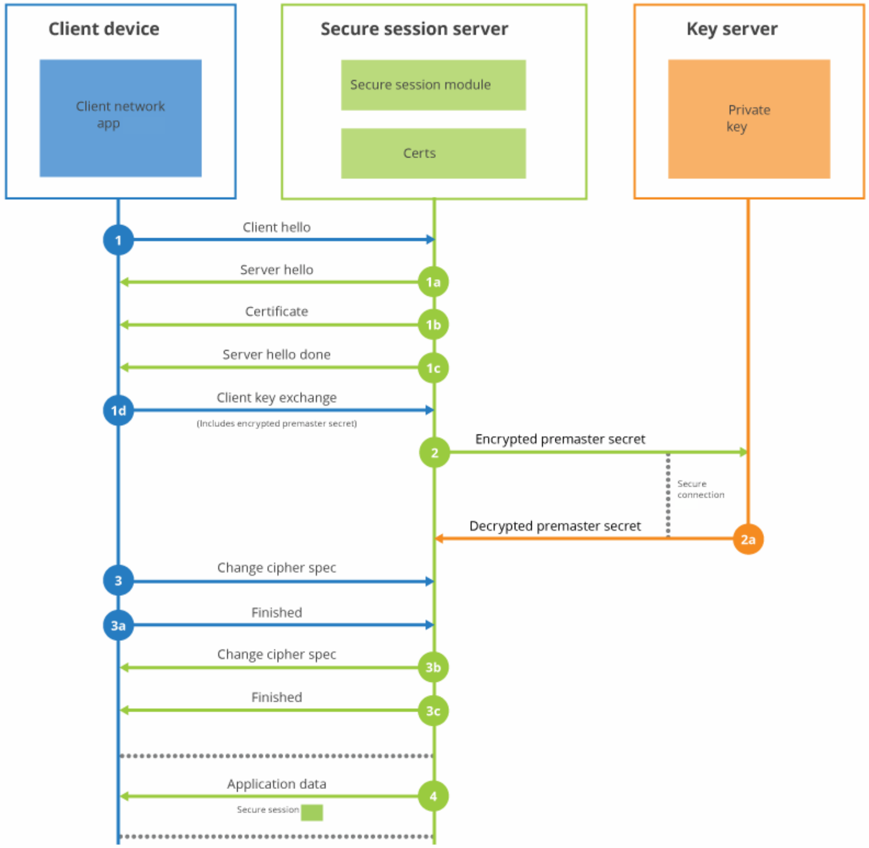
How Keyless SSL Works

- Split the TLS state machine **geographically**
  - Perform private key operation at site owner’s facility (in HSM, etc)
  - Perform rest of handshake at edge
  - Communicate with signing server over mutually authenticated TLS

Keyless SSL Diagram



Keyless SSL Handshake



Client device

Client network app

Secure session server

Secure session module

Certs

Key server

Private key

1 Client hello

1a Server hello

1b Certificate

1c Server hello done

1d Client key exchange

(Includes encrypted premaster secret)

2 Encrypted premaster secret

Secure connection

2a Decrypted premaster secret

3 Change cipher spec

3a Finished

3b Change cipher spec

3c Finished

4 Application data

Secure session 🟩

Conclusion

Conclusion

- Untrusted hardware requires a new approach
  - Split secrets into long-term and short-term
  - Design for rapid renewal — replace secrets faster than they can be broken
  - Leverage short-term secrets to access long-term secrets