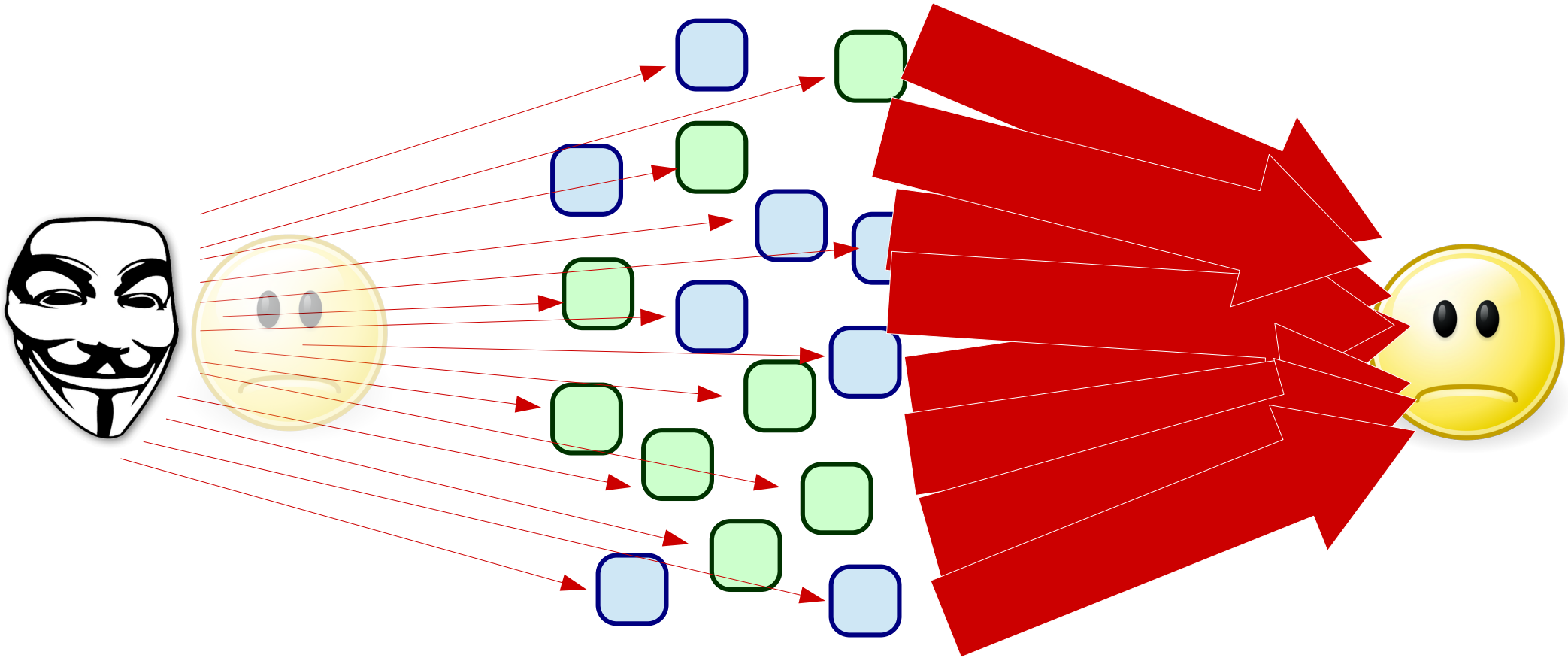
Interoperable DNS Server Cookies

Willem Toorop
Ondřej Surý
Donald E. Eastlake 3rd
Mark Andrews

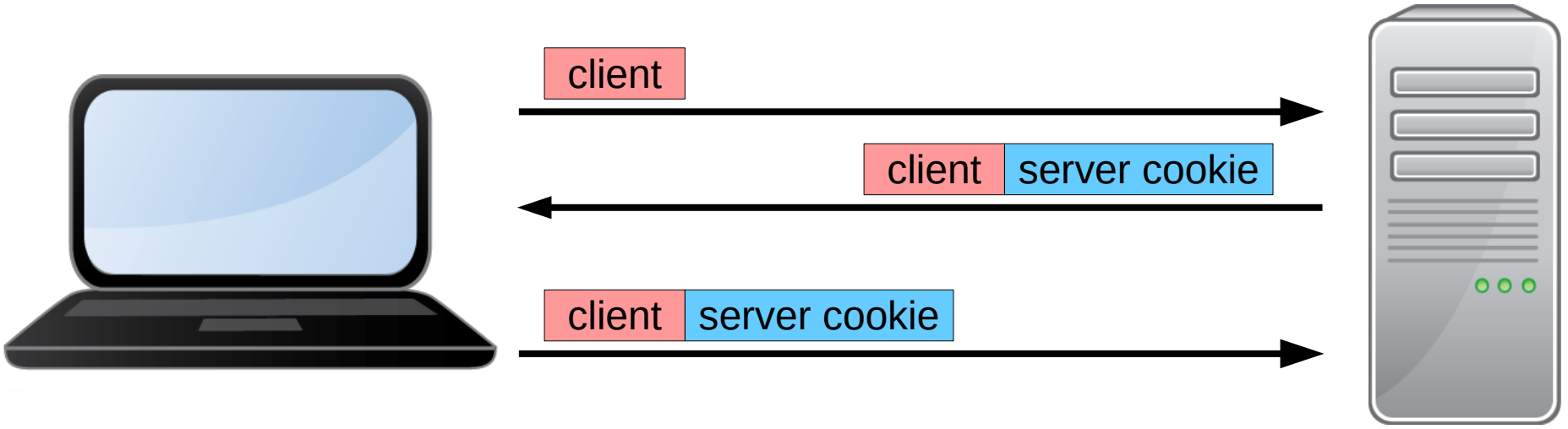
draft-ietf-dnsop-server-cookies-02

CC0 Public Domain from publicdomainpictures.net

Why DNS Cookies?

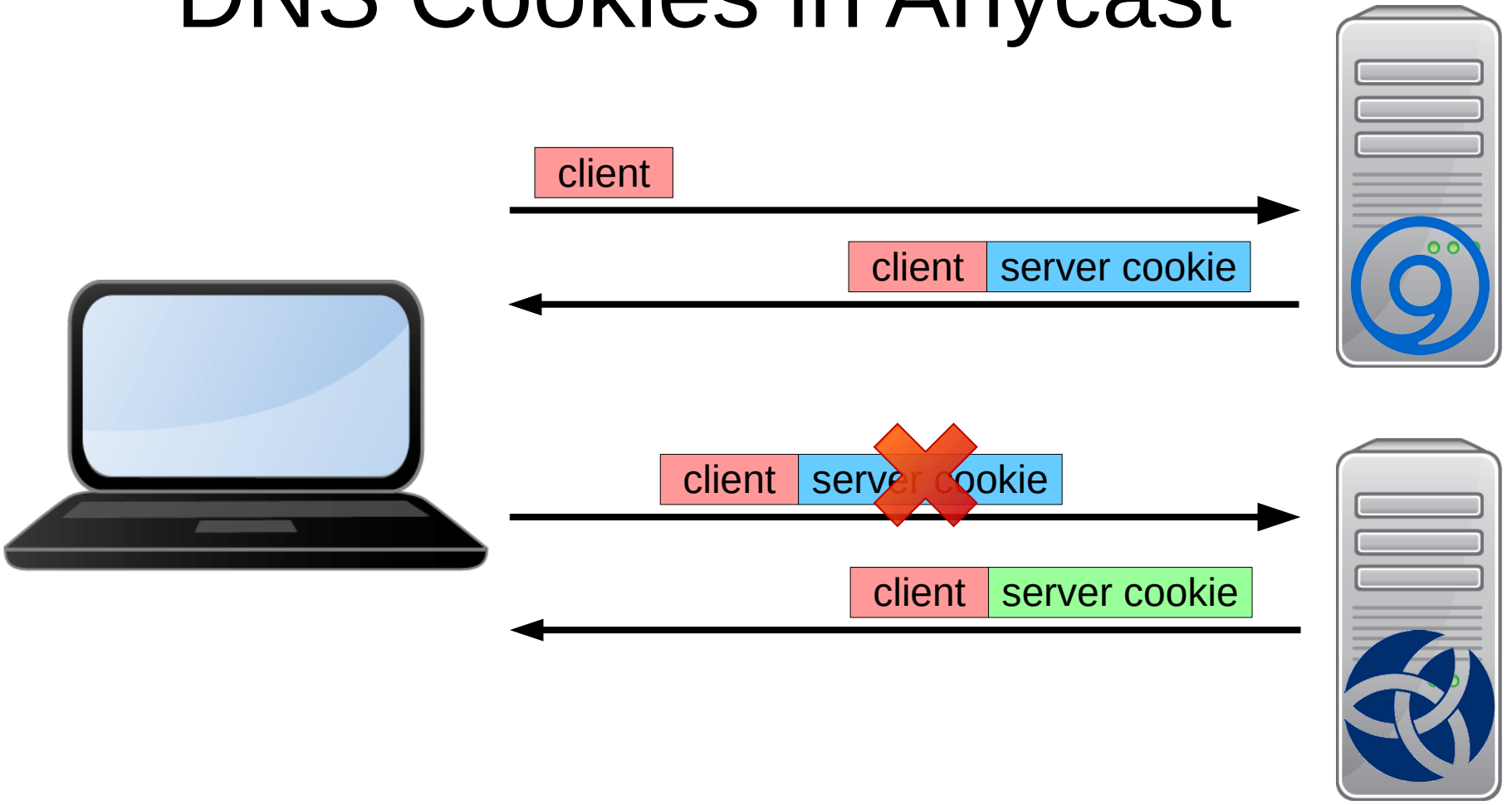


DNS Cookies Operation



- Valid Server Cookie? Large answers
- Valid Server Cookie? RRL Disabled!

DNS Cookies in Anycast



Hackthon @IETF104 Results

- Witold Krecicki, Ondřej Surý, Pieter Lexis, Willem Toorop
- Focus on the Server Cookie
- Interoperable Server Cookies for:

Hackthon @IETF105 Results

- Witold Krecicki, Ondřej Surý, Pieter Lexis, Willem Toorop
- Focus on the Client Cookie

```
Client-Cookie = MAC_Algorithm(
    Client IP Address | Server IP Address, Client Secret )
```

- **`Server IP Address`** for minimal authentication
- **`Client IP Address`** for privacy

Hackthon @IETF105 Results

Client-Cookie = MAC_Algorithm(
Client IP Address | Server IP Address, Client Secret)

```
void send_query(int sock, struct pkt *query , struct sockaddr *dst, socklen_t dst_len)
{
    struct sockaddr_storage my_ip;
    socklen_t               my_ip_len;
    uint8_t                 new_client_cookie[8];

    getsockname(sock, (struct sockaddr*)&my_ip, &my_ip_len);

    mk_client_cookie(new_client_cookie, my_ip, dst, secret);

    if (memcmp(query->client_cookie, new_client_cookie, 8)) {
            memcpy(query->client_cookie, new_client_cookie, 8);
            query->server_cookie = NULL;
    }
    sendto(sock, query->buf, query->len, 0, dst, dst_len);
}
```

Hackthon @IETF105 Results

Client-Cookie = MAC_Algorithm(
Client IP Address | Server IP Address, Client Secret)

```
void send_query(int sock, struct pkt *query , struct sockaddr *dst, socklen_t dst_len)
{
    struct sockaddr_storage my_ip;
    socklen_t               my_ip_len;
    uint8_t                 new_client_cookie[8];

    getsockname(sock, (struct sockaddr*)&my_ip, &my_ip_len);

    mk_client_cookie(new_client_cookie, my_ip, dst, secret);

    if (memcmp(query->client_cookie, new_client_cookie, 8)) {
            memcpy(query->client_cookie, new_client_cookie, 8);
            query->server_cookie = NULL;
    }
    sendto(sock, query->buf, query->len, 0, dst, dst_len);
}
```

Hackthon @IETF105 Results

Client-Cookie = MAC_Algorithm(
Client IP Address | Server IP Address, Client Secret)

```
void send_query(int sock, st
{
    struct sockaddr_storage
    socklen_t
    uint8_t

    getsockname(sock, (struct sockaddr*)&my_ip, &my_ip_len);

    mk_client_cookie(new_client_cookie, my_ip, dst, secret);

    if (memcmp(query->client_cookie, new_client_cookie, 8)) {
            memcpy(query->client_cookie, new_client_cookie, 8);
            query->server_cookie = NULL;
    }
    sendto(sock, query->buf, query->len, 0, dst, dst_len);
}
```

```
int trial_sock = socket(dst->sa_family, SOCK_DGRAM, 0);

connect(trial_sock, dst, dst_len);
getsockname(sock, (struct sockaddr*)&my_ip, &my_ip_len);
close(trial_sock);
```

Hackthon @IETF106 Results

~~`Client-Cookie = MAC_Algorithm(`~~
~~`    Client IP Address | Server IP Address, Client Secret )`~~

`Client-Cookie = 64 bits of entropy`

- **`Server IP Address`** for minimal authentication
  - Create new random Client Cookie for each new Server
  - If Server returns Server Cookie:
    - Register Client IP alongside Client Cookie & Server Cookie
- **`Client IP Address`** for privacy
  - Bind UDP socket to Client IP …. Reset Cookies on failure

draft-ietf-dnsop-server-cookies-02

- Rewritten *Constructing a Client Cookie* Section
- New *Security and Privacy Considerations* Section
- Took a few iterations … *Thank you Philip Homburg!*

- **Next step**, Implementation experience with


CC BY 2.0 by Scott McLeod