Case Studies

Learn from the mistakes of others

Michael L Perry
qedcode.com
@michaellperry

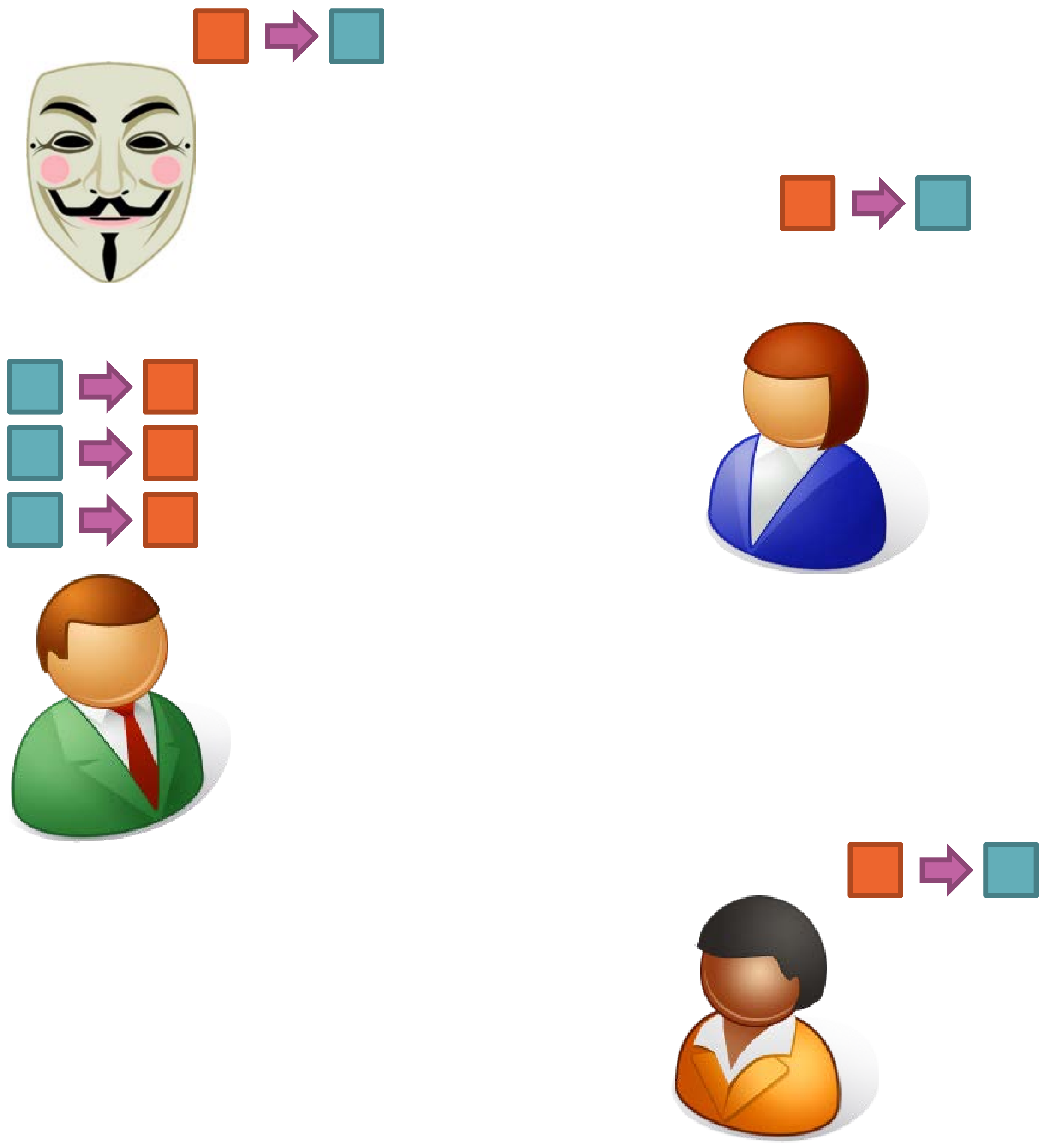
pluralsight
hardcore dev and IT training

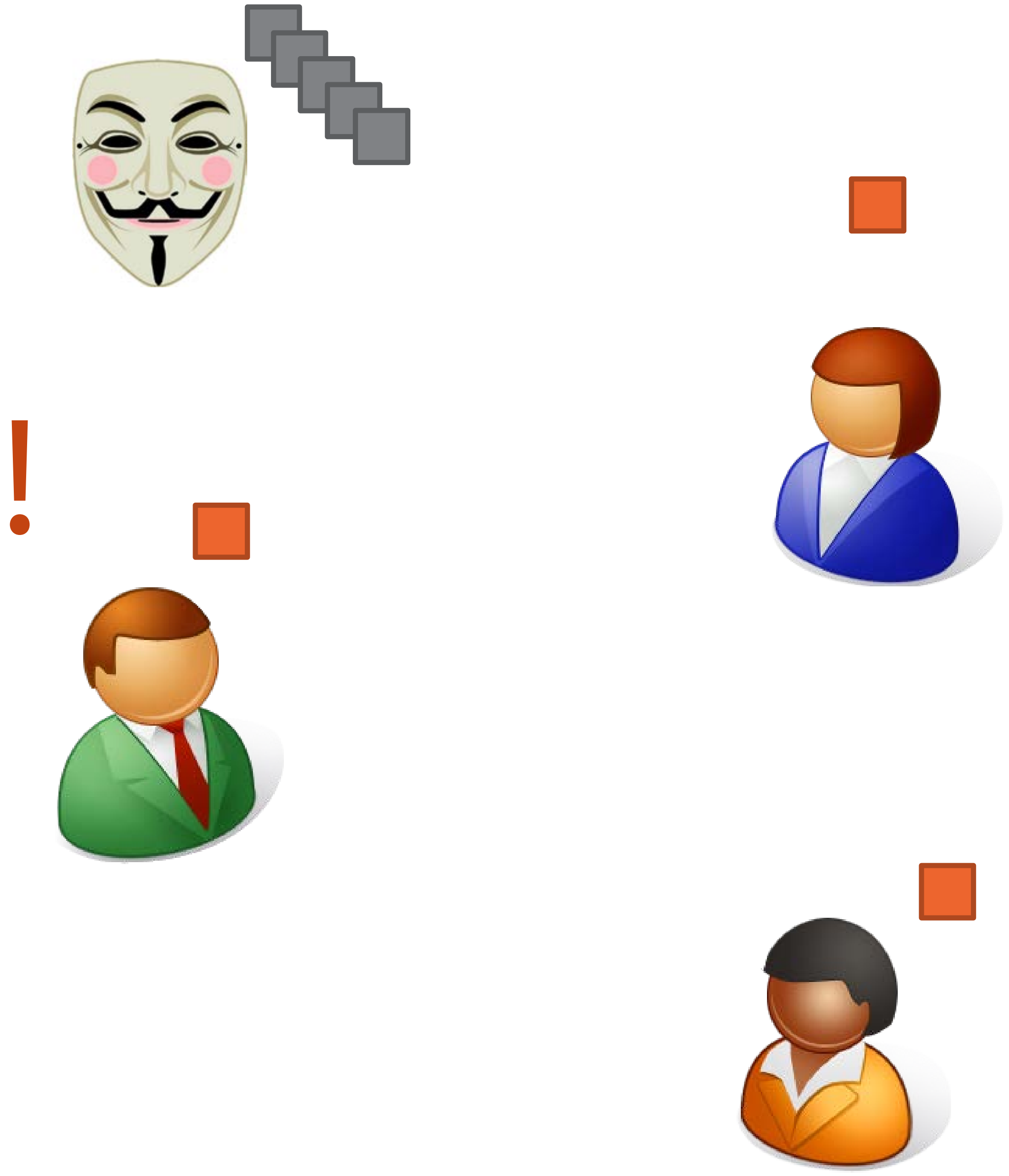
Snapchat

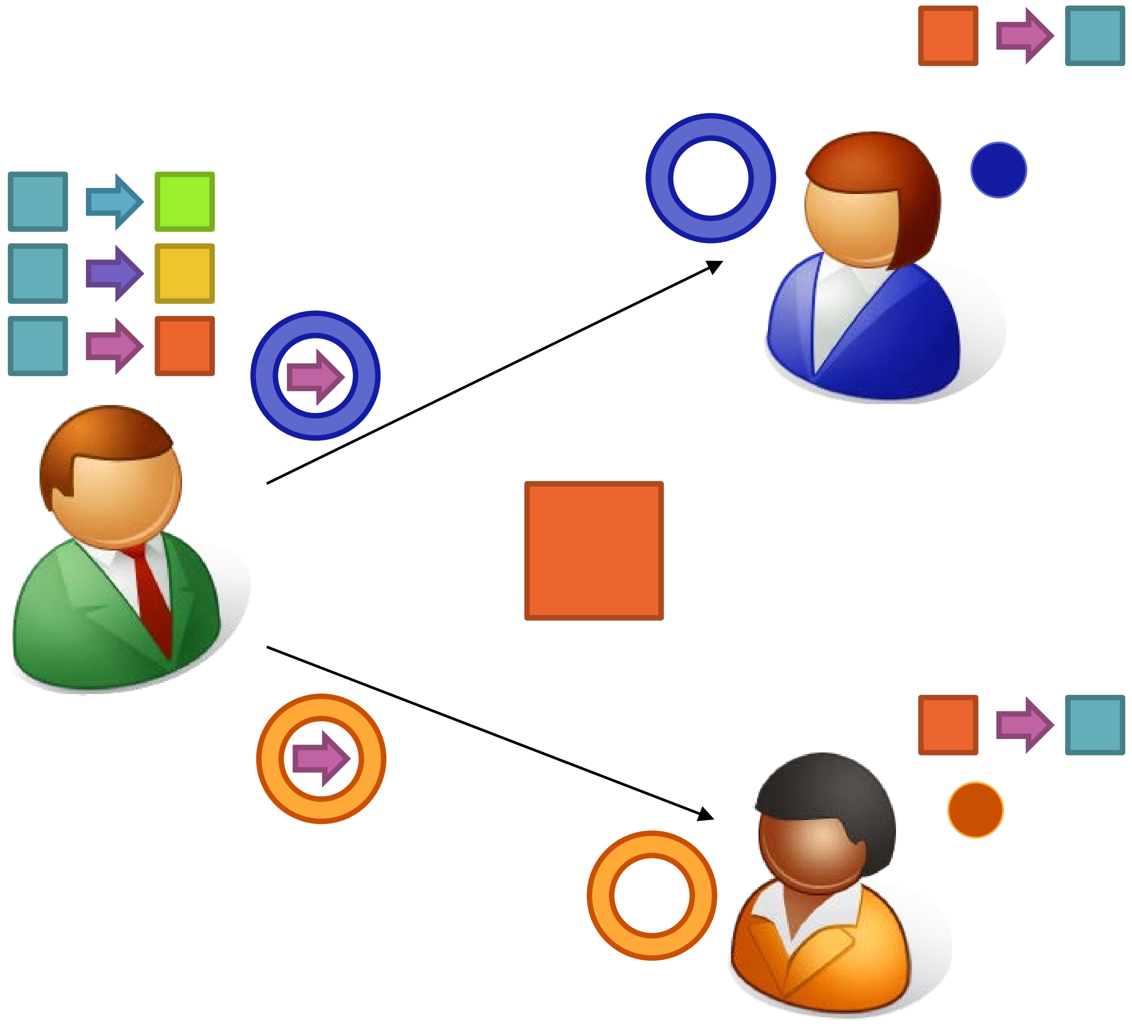
- **Social Network**
- **Reviewed by Gibson Research**
  - http://gibsonsec.org/snapchat
  - Steve Gibson
- **Published API**
  - …and security flaws

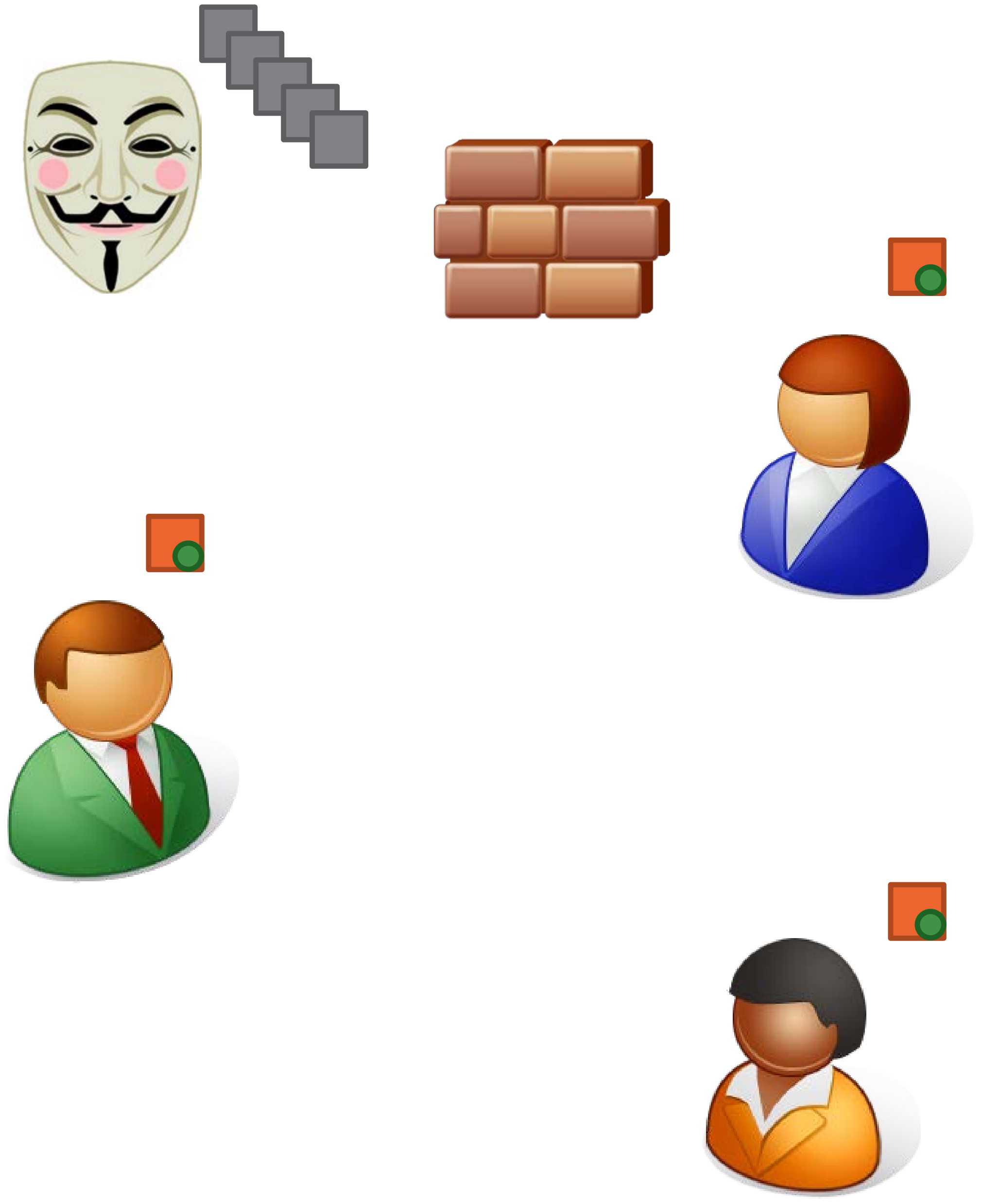
Encryption of Snaps

- **AES with Electronic Code Book**
  - Weak block cypher mode
- **Changed to Cypher Block Chaining**



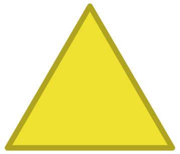






Find Friends

555-1212	→	chrisecclelston9
555-1213	→	0
555-1214	→	0
555-1215	→	mattsmith11
555-1216	→	0
555-1217	→	0
555-1218	→	tombaker4
555-1219	→	davidtennant10
555-1220	→	0
555-1221	→	0
555-1222	→	0



Don't

- Use hard-coded symmetric keys
- Embed keys in mobile apps

Do

- Generate a new key for each message
- Use asymmetric algorithms to securely exchange keys
- Digitally sign messages to identify the sender

Safari

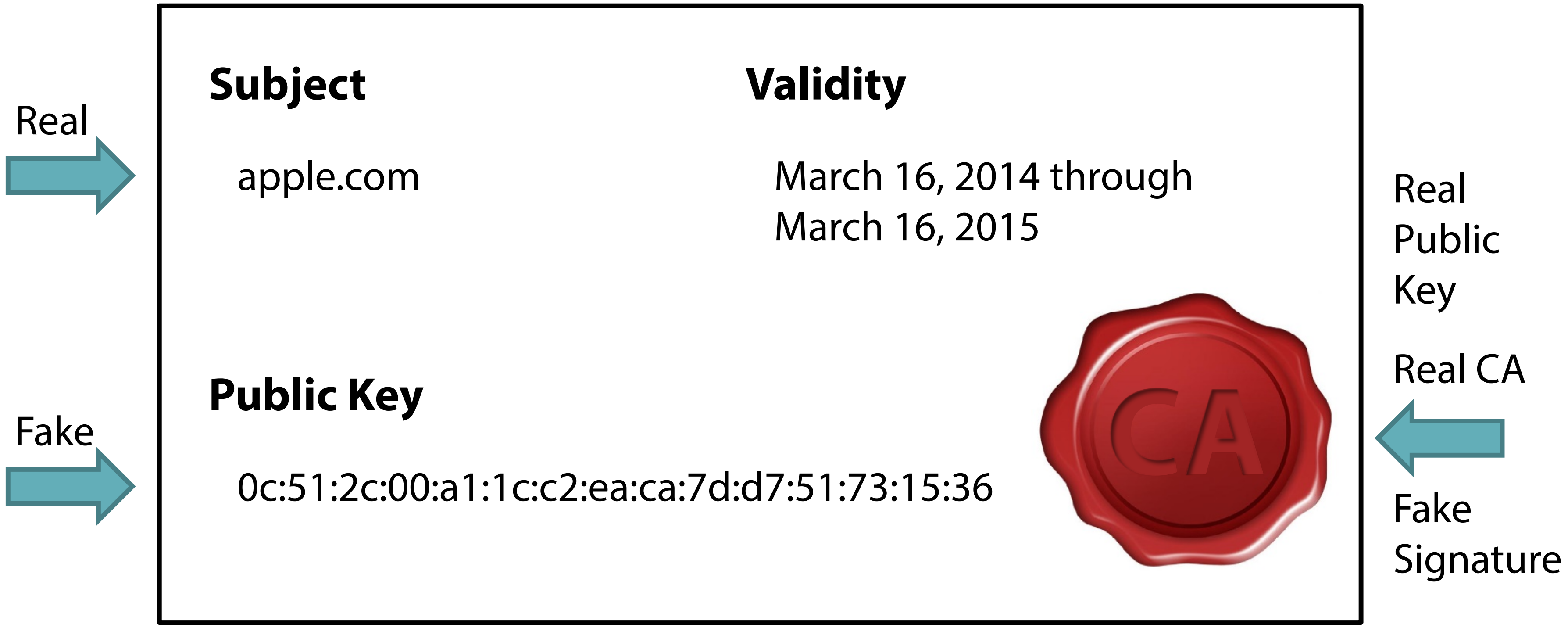
- **Reviewed by Adam Langly**
    - https://www.imperialviolet.org/2014/02/22/applebug.html

```
static OSStatus
SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRsa, SSLBuffer signedParams,
    uint8_t *signature, UInt16 signatureLen)
{
    OSStatus err;
    ...

    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
        goto fail;
        goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
        goto fail;
    ...

fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    return err;
}
```

Exploit



Subject

apple.com

Validity

March 16, 2014 through
March 16, 2015

Public Key

0c:51:2c:00:a1:1c:c2:ea:ca:7d:d7:51:73:15:36

Generating Fraudulent Certificate

- **Can’t use OpenSSL**
  - Will generate a valid siguature
  - Requires a private key
- **Open source**
  - Modify software

Don't

- Write your own security code

Do

- Hire auditors if you write security code
- Run penetration tests
- Rely upon trusted vendors (ironically)

Heartbleed

- OpenSSL
- Reviewed by Sean Cassidy
  - http://blog.existentialize.com/diagnosis-of-the-openssl-heartbleed-bug.html

Dangers

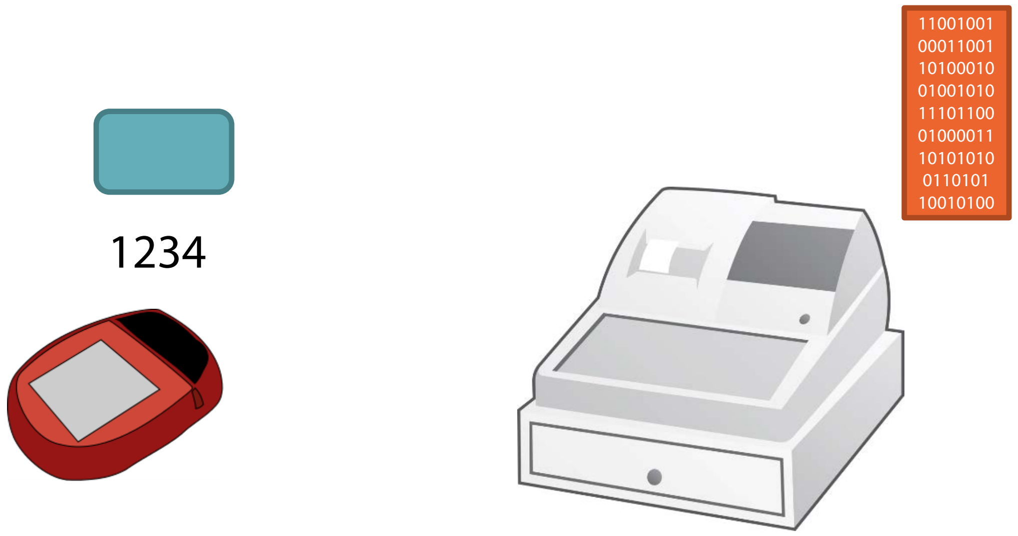
- **Private keys for X.509 certificates**
- **Demonstrated vulnerability**
- **Leaves no trace**
- **Certificates installed on OpenSSL for the past two years are suspect**
  - Revoke

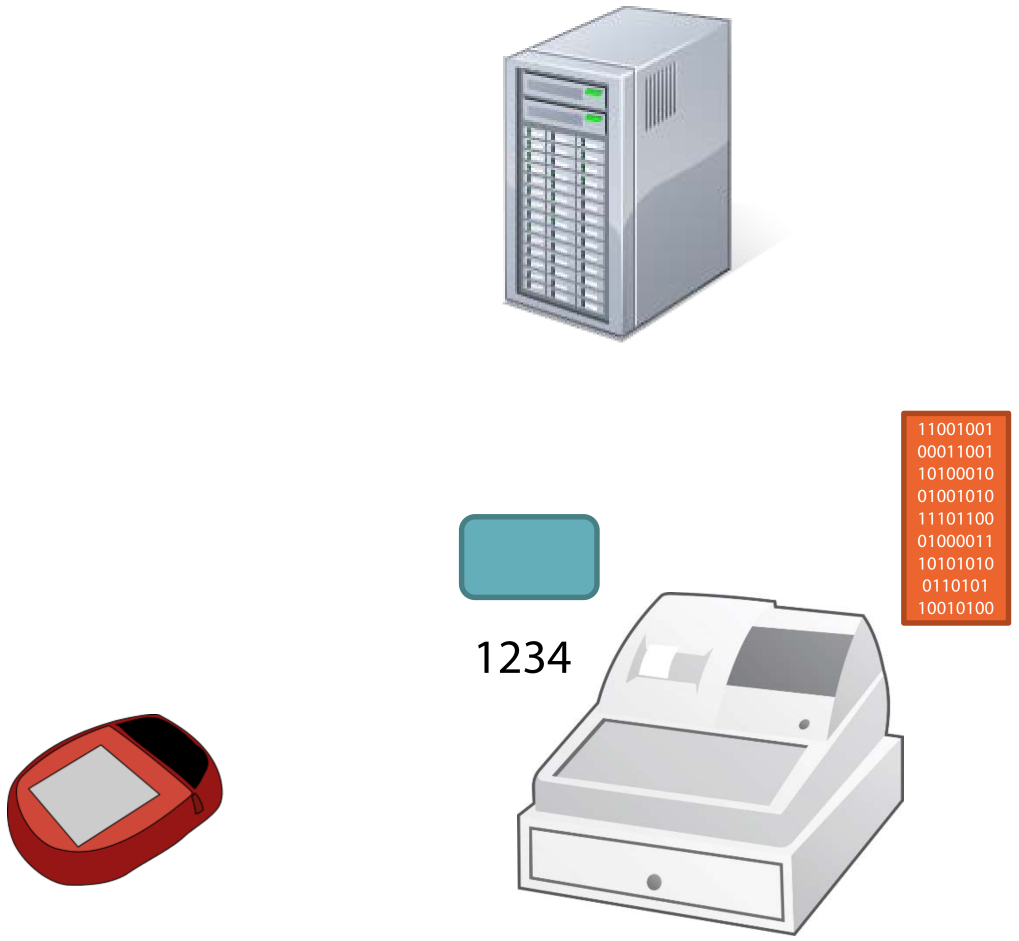
Takeaways

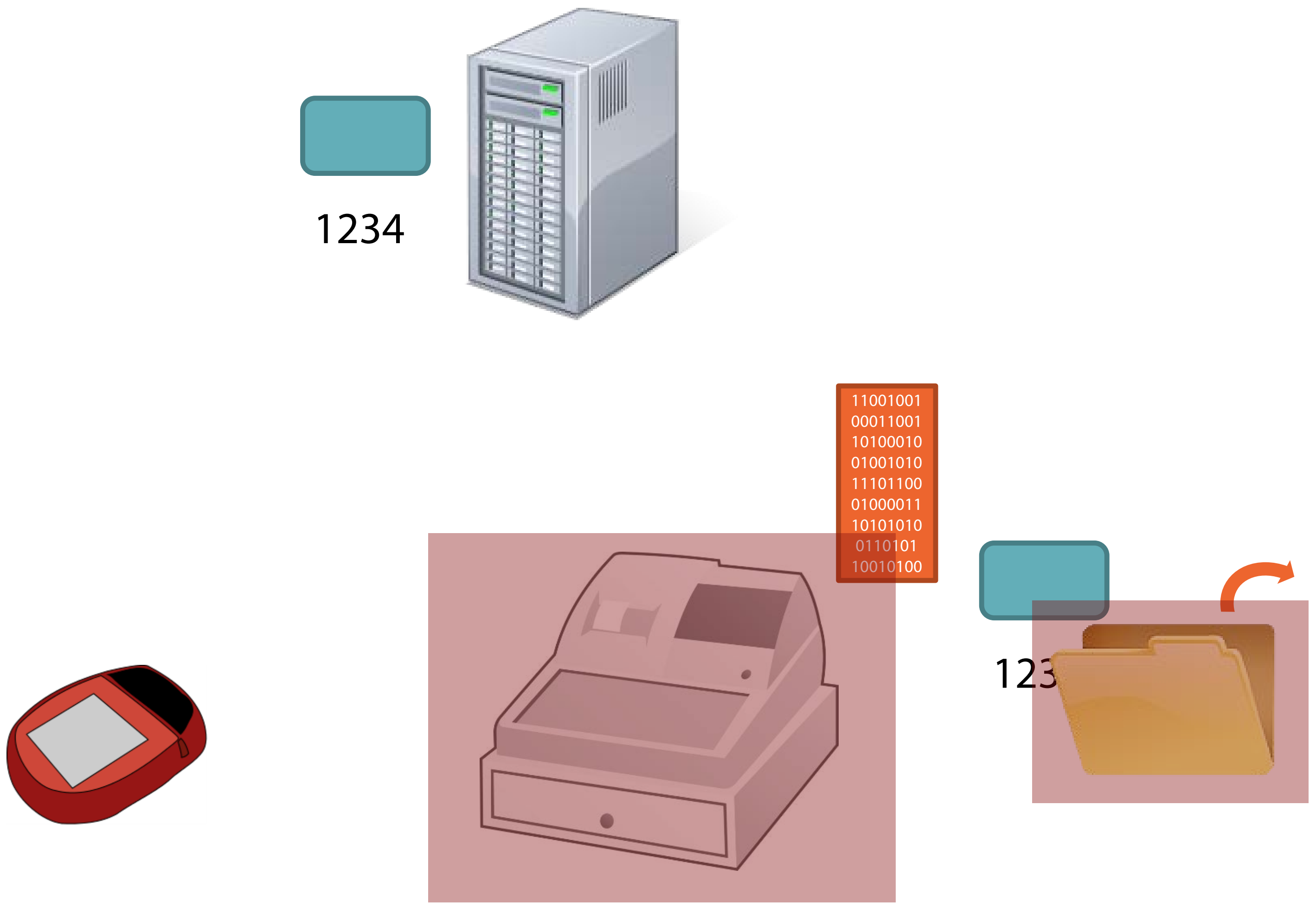
- **Do open source projects undergo public scrutiny?**
- **Can we trust private vendors?**
- **Do not write security code yourself**

Target

- **Reviewed by Brian Krebs**
  - http://krebsonsecurity.com/2014/01/new-clues-in-the-target-breach/

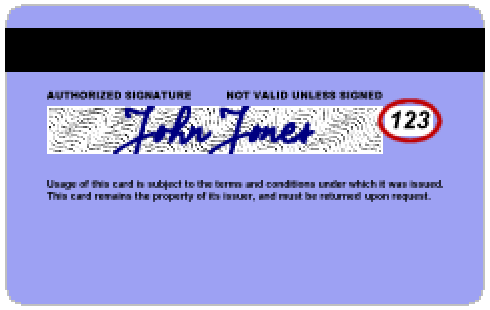

1234
11001001
00011001
10100010
01001010
11101100
01000011
10101010
0110101
10010100


11001001
00011001
10100010
01001010
11101100
01000011
10101010
0110101
10010100
1234

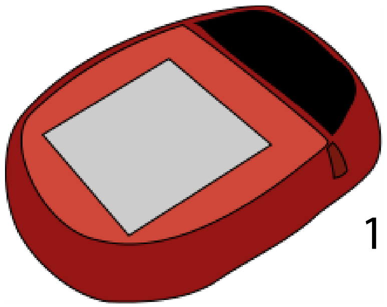

1234
11001001
00011001
10100010
01001010
11101100
01000011
10101010
0110101
10010100
123

Stolen Data

- **Track Data**
  - Not encrypted!
  - Card number
  - Name of person
  - CVV1 (Card Verification Value)
- **PIN Numbers**

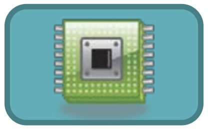

AUTHORIZED SIGNATURE
NOT VALID UNLESS SIGNED
123
Usage of this card is subject to the terms and conditions under which it was issued.
This card remains the property of its issuer, and must be returned upon request.

Chip and PIN

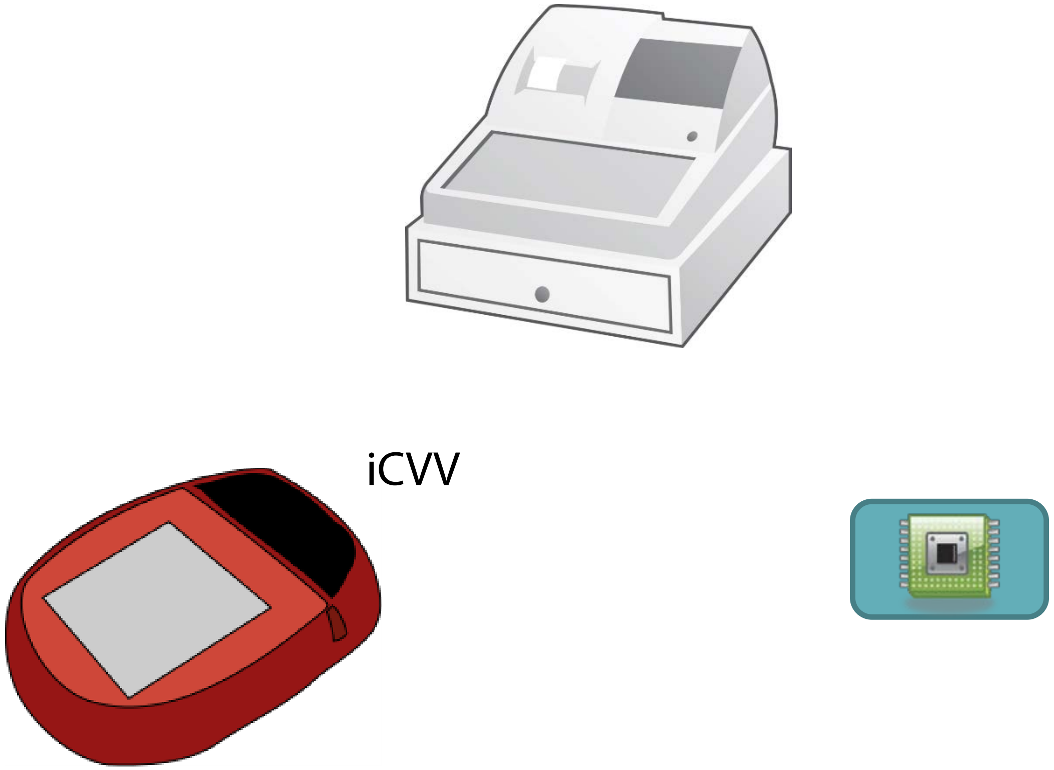


1234

iCVV

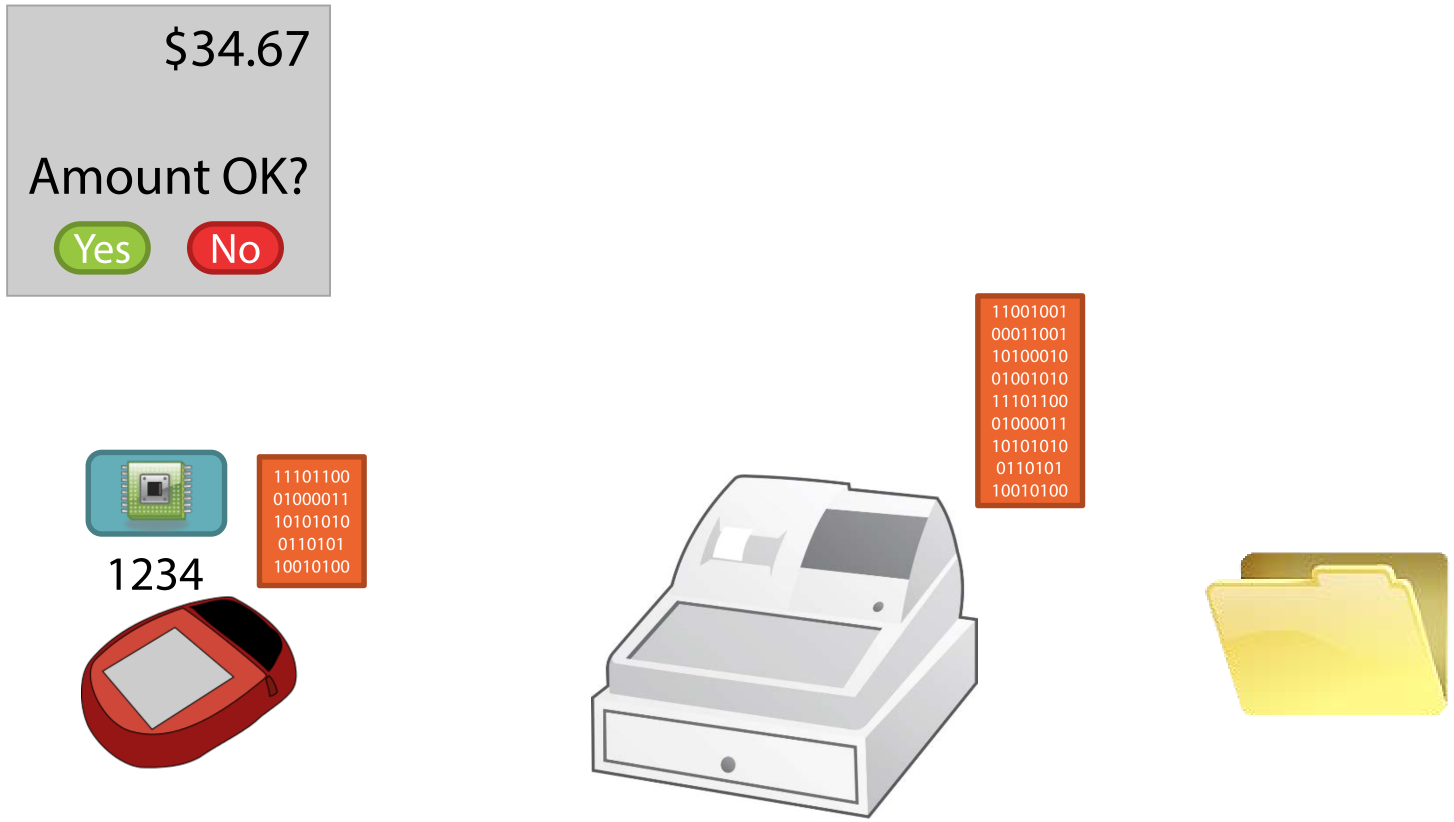


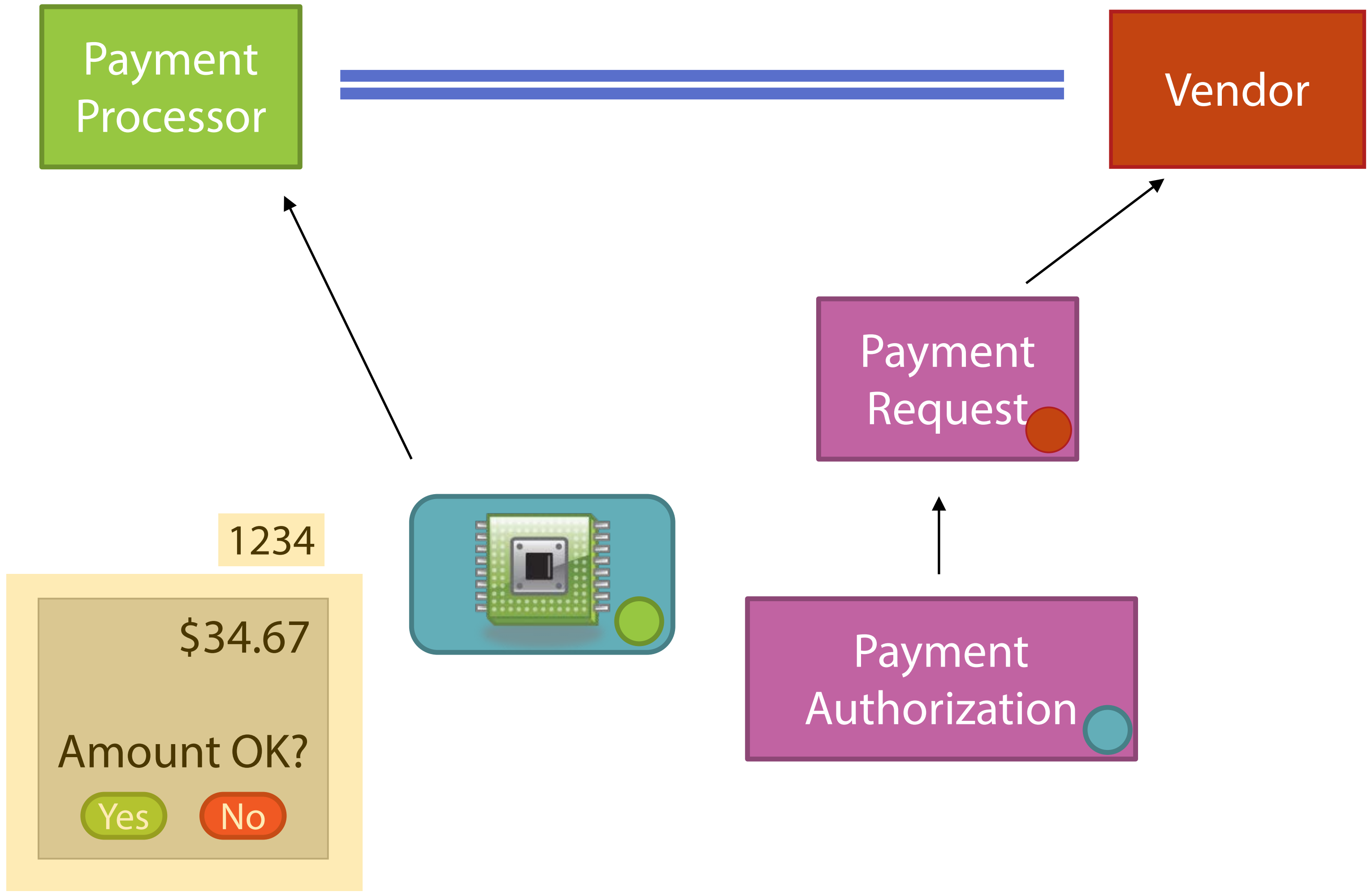
Chip and PIN

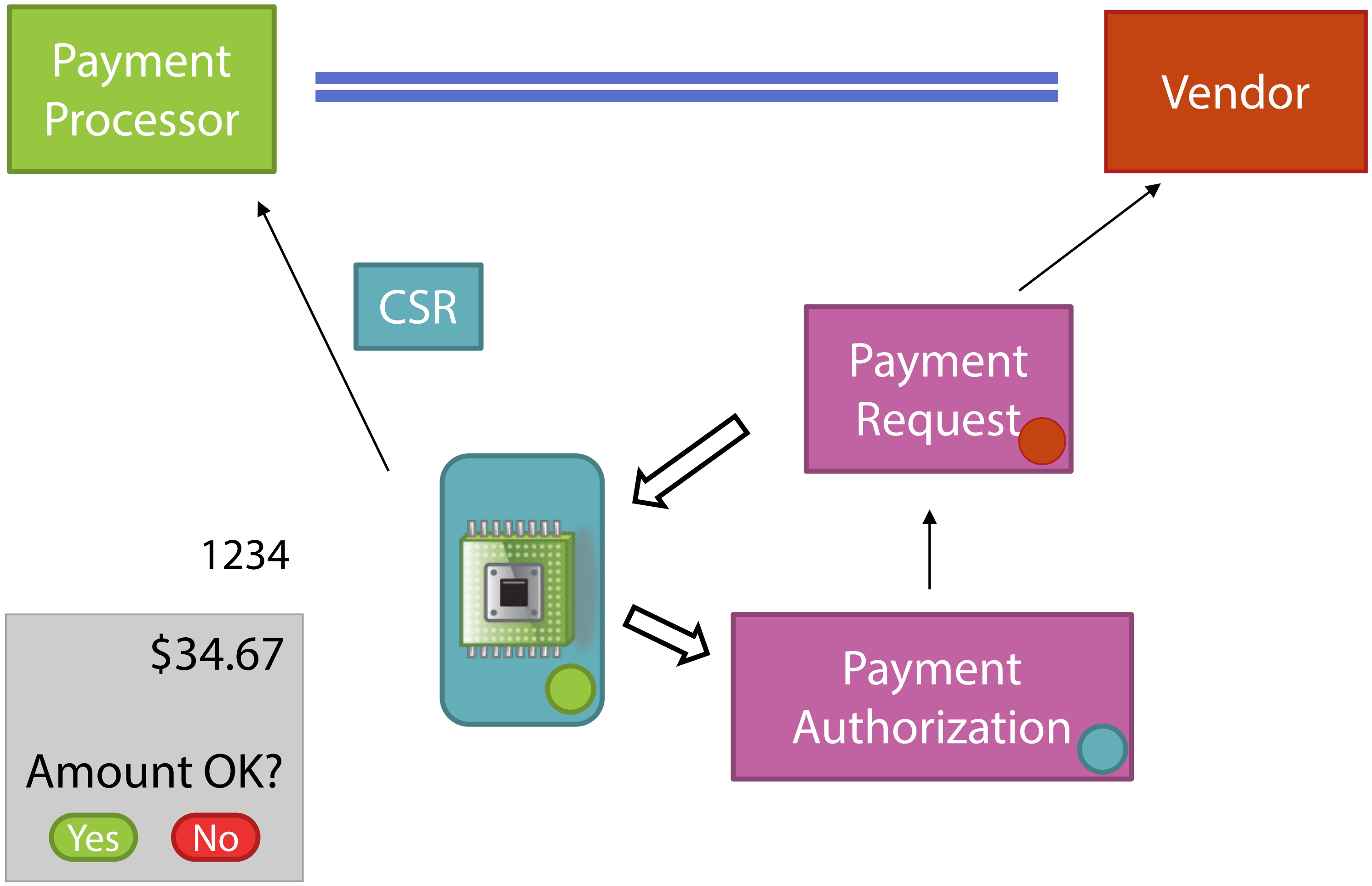


Chip and PIN Systems

- **Static iCVV**
  - Vulnerable to replay attack
- **Dynamic iCVV**
  - Generate a random number
  - Generate iCVV
  - Send both to payment processor
  - Number used once (NONCE)


$34.67
Amount OK?
Yes
No
1234
11101100
01000011
10101010
0110101
10010100
11001001
00011001
10100010
01001010
11101100
01000011
10101010
0110101
10010100


Payment Processor
Vendor
Payment Request
1234
$34.67
Amount OK?
Yes
No
Payment Authorization

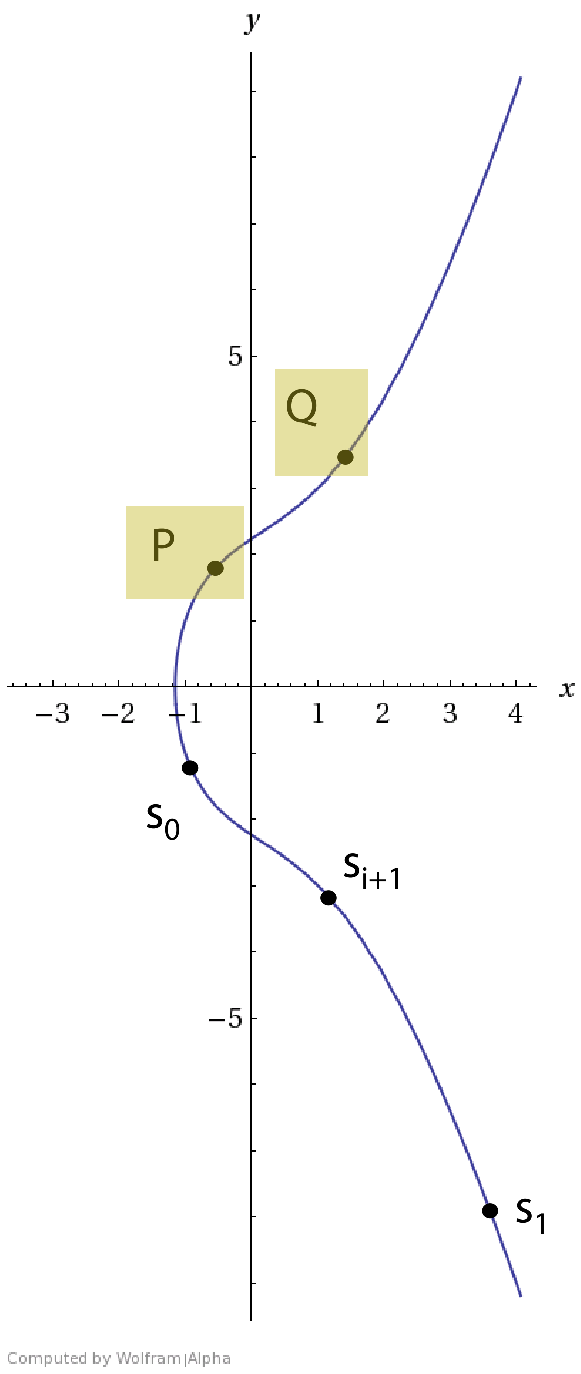

Payment Processor
Vendor
CSR
Payment Request
1234
$34.67
Amount OK?
Yes
No
Payment Authorization

NSA

- **DES**
  - Replaced S-box
  - Back door?
  - Weaknesses not related to S-box
  - Differential cryptographic analysis
  - NSA S-box resilient to differential cryptographic analysis

Dual Elliptic Curve

- Pseudo Random Number Generator



$r_0 = \varphi(s_0 P)$ $n_0 = lsb(r_0)$

$s_1 = \varphi(r_0 Q)$ $r_1 = \varphi(s_1 P)$ $n_1 = lsb(r_1)$

$s_i = \varphi(r_{i-1} Q)$ $r_i = \varphi(s_i P)$ $n_i = lsb(r_i)$

$s_{i+1} = \varphi(r_i Q)$

Back Door?

- **Dan Shumow and Niles Ferguson**
  - Microsoft

NSA?

Known e

$$Q^e = P$$

Given n_i, n_{i+1}, n_{i+2}

Could determine s_{i+3}

And produce n_{i+3}, n_{i+4}, $n_{i+5}\ldots$

Vulnerability

- **Generate symmetric keys**
- **Make valid requests**
  - Capture n_i, n_{i+1}, n_{i+2}
  - Reconstitute s_{i+3}
  - Produce n_{i+3}, n_{i+4}, n_{i+5}…
  - Try candidate keys
- **Attacker could spy on other users of the system**

Don't

- Use numbers generated by a third party

Do

- Understand the math
- Generate your own numbers
- Trust vendors
- And verify

Lessons Learned

- **Avoid static symmetric keys**
- **Keep private key private**
- **Use asymmetric cryptography to establish trust**
- **Don’t write crypto code yourself**
  - Or if you must, audit it regularly
- **Question crypto provided by a third party**
  - Understand the source
  - Understand the implementation

Snapchat

Safari

Heartbleed

Target

NSA