RSAConference2016

San Francisco | February 29 – March 4 | Moscone Center

Connect to Protect

SESSION ID: PDAC-R02

How Broken Is Our Crypto Really?

Martijn Grooten

Editor, Security Researcher
Virus Bulletin
@martijn_grooten

#RSAC

How Broken Is Our Crypto Really?

UK high-street banks accused of "shockingly bad" online security (SC Magazine)

RC4 NOMORE crypto exploit used to decrypt user cookies in mere hours (ZDNet)

Why Diffie–Hellman Encryption May Be Past Its Prime (IBM Security Intelligence)

virus BULLETIN

How Broken is Our Crypto Really?

Or: what do we mean when we say as cryptographic protocol is “broken”?

What is this talk about?

Broken algorithms and protocols and the likeliness of these being exploited.

- Diffie Hellman / Logjam
- SSL3 / POODLE
- RC4 / NOMORE
- SHA-1

What is this talk NOT about?

- An excuse not to use the latest and greatest cryptographic protocols
- Implementation issues
- Short keys and downgrade attacks
- Dual_EC_DRBG

Dual_EC_DRBG

- This algorithm (likely) contains a backdoor
- This backdoor can be (and likely has been) exploited
- This backdoor was (likely) inserted deliberately
- **This is an exception**

Introducing Alice and Bob

Alice

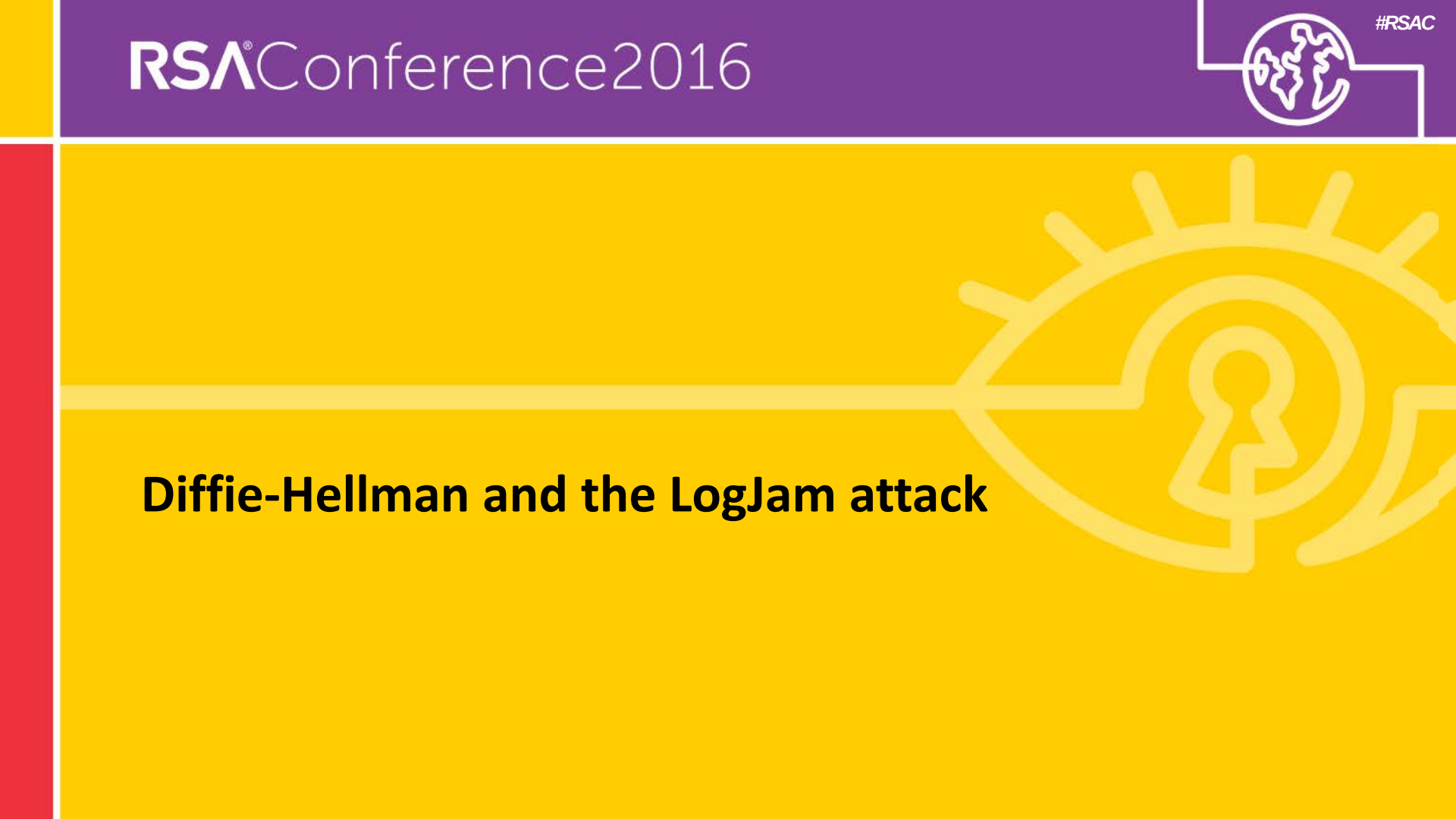
Eve

Bob

- Authentication (SHA-1)
- Key agreement (Diffie-Hellman)
- Encrypted data-exchange (SSL3, RC4)

HTTP and HTTPS: ideal for crypto-attacks

- Alice visits **https**://target.website/ and also **http**://example.com/
- With a MitM-position, Eve can freely insert JavaScript into http://example.com/ to force Alice to make certain requests
- This is ideal for many crypto-attacks, but doesn't work in other protocols (SMTP, SSH etc.)



Diffie-Hellman and the LogJam attack

Agreeing on protocol standards

Alice and Bob publicly agree on a “group” *g* to be used.

g?

OK!

Generating private keys

Alice chooses a (large) random number *a.*

Bob chooses a (large) random number *b*.

Both are kept secret.

a

b

Computing public keys

Using a ‘mechanism’, Alice computes a number g^a.

Bob computes a number g^b.

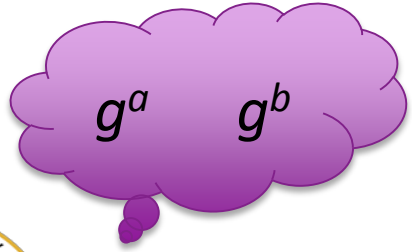
Both numbers are shared publicly.

g^a

g^b

Who knows what?

- Alice knows a, g^a, g^b.
- Bob knows b, g^a, g^b.
- Eve knows g^a, g^b.



Generating the public key

- Alice combines a and g^b to compute $(g^b)^a$.
- Bob combines b and g^a to compute $(g^a)^b$.
- Mathematics is kind to us:

$$(g^b)^a = (g^a)^b$$

- This is the secret key they will use.
- The key is broken if Eve can compute a from g^a (or b from g^b).

Nothing is impossible

- Like most cryptographic algorithms, Diffie-Hellman can be broken. It is just very expensive to do so.
- Breaking 1024-bit Diffie-Hellman costs about $100m.
- **No one will ever pay that to crack a single key!**

So how would such an attack work?

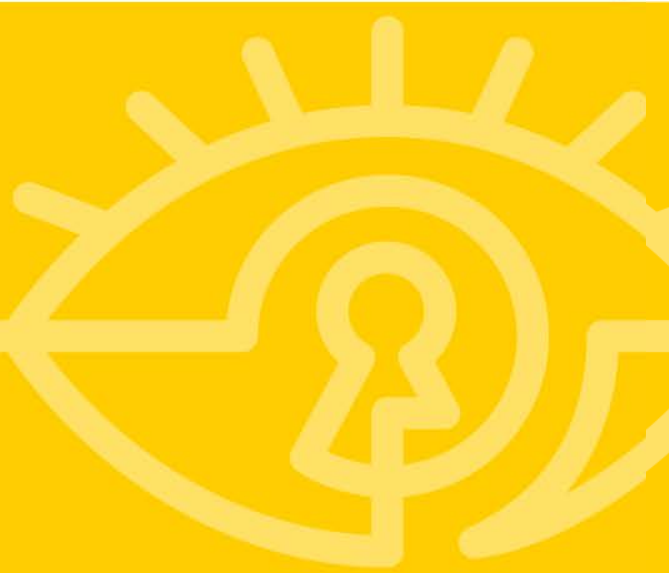
- Most of the attack involves computations specific only to the group *g*.
- Only a small (and cheap) part of the attack targets the specific numbers g^a and g^b.
- So what if most Alices and Bobs use the same group *g*?

Guess what?

- Many Alices and Bobs do indeed use same group *g*: e.g. some HTTPS, VPN implementations.
- For $100m worth of “preparations” you can crack all these implementations at once, in (near) real-time.
- This is the “LogJam” attack.
- It is believed that this is how the NSA has broken many VPN connections.

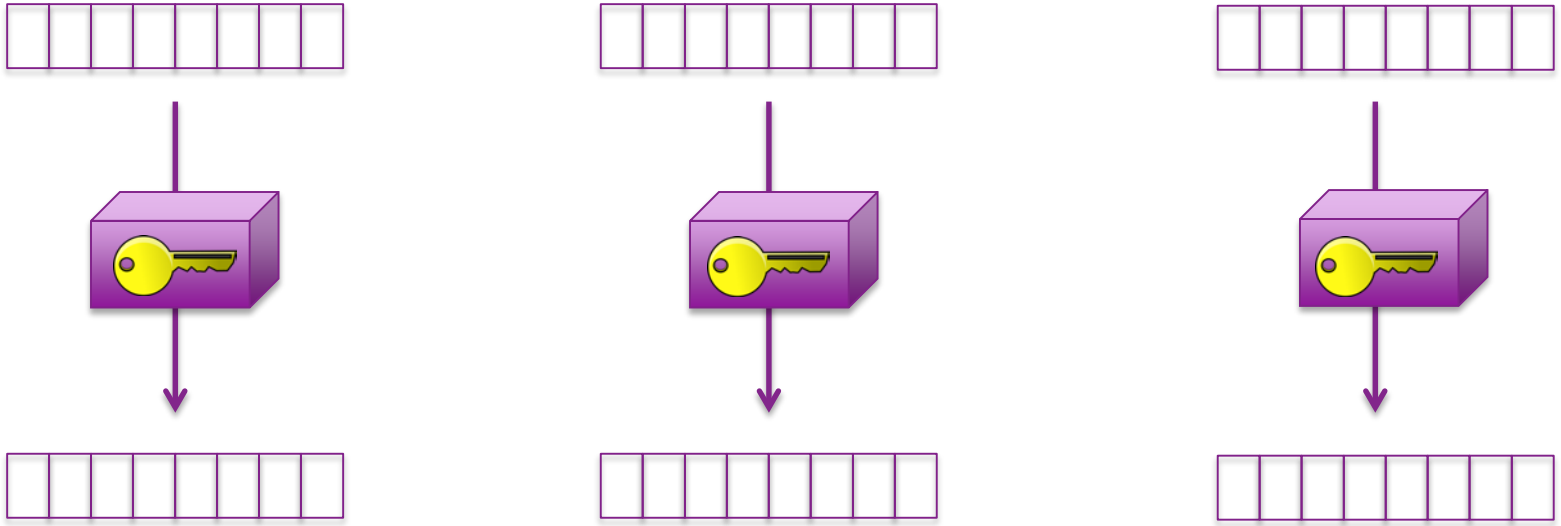
Is Diffie-Hellman broken?

- Diffie-Hellman has **not** been broken!
- 2048-bit Diffie-Hellman is very secure.
- 1024-bit Diffie-Hellman with a unique *g* is secure in practice. (But it's best avoided.)



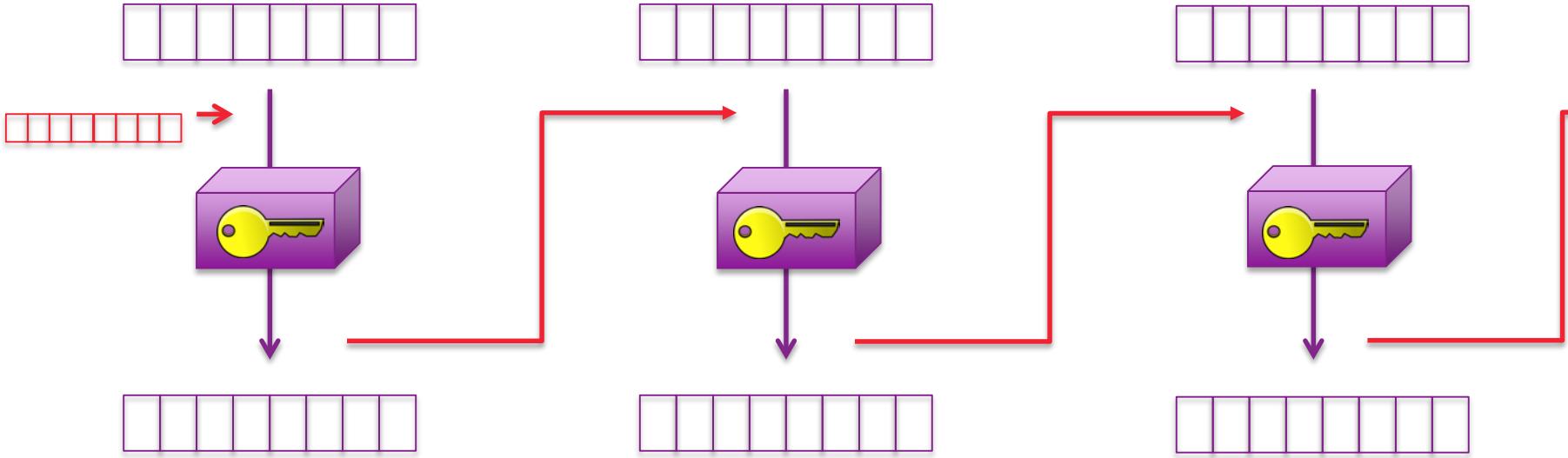
SSL3 and POODLE

Block cipher encryption in SSLv3



Electronic Codebook (ECB)

Block cipher encryption in SSLv3

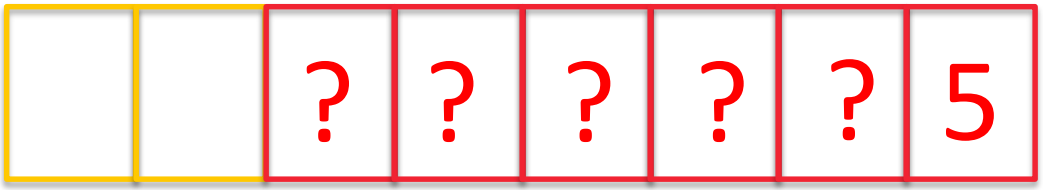


Cipher Block Chaining (CBC)

“Tidying up” the plaintext

- Add a MAC to “avoid tampering”
- Add padding to fill up final block

Padding

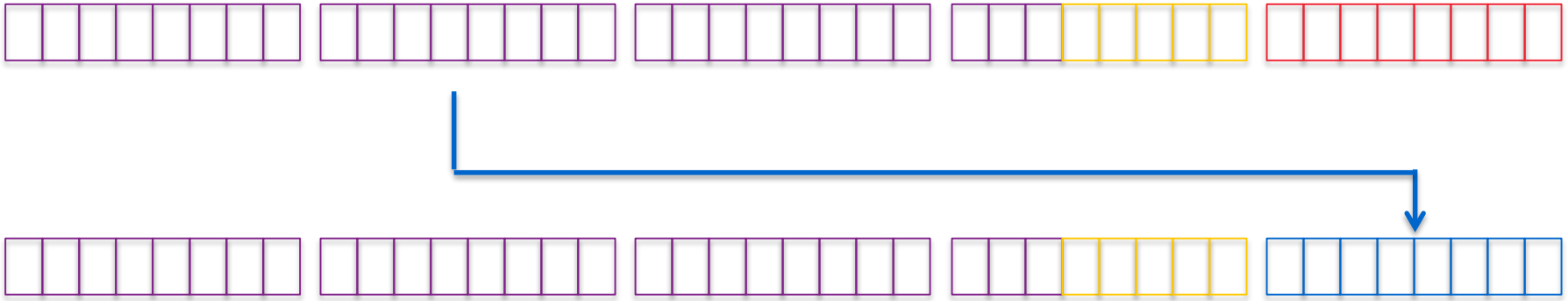


size of rest of padding

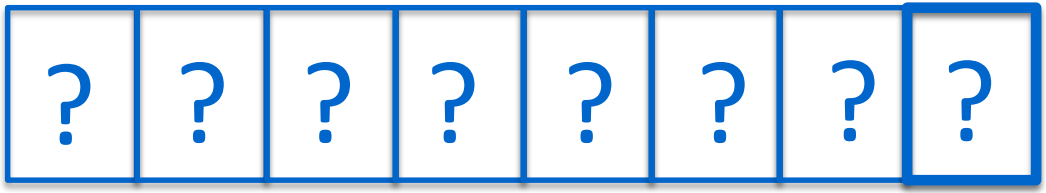
whatever

If padding fills last block

- In HTTPS, Eve can make sure Alice's request looks like this
- She can also replace the final block with a previous block

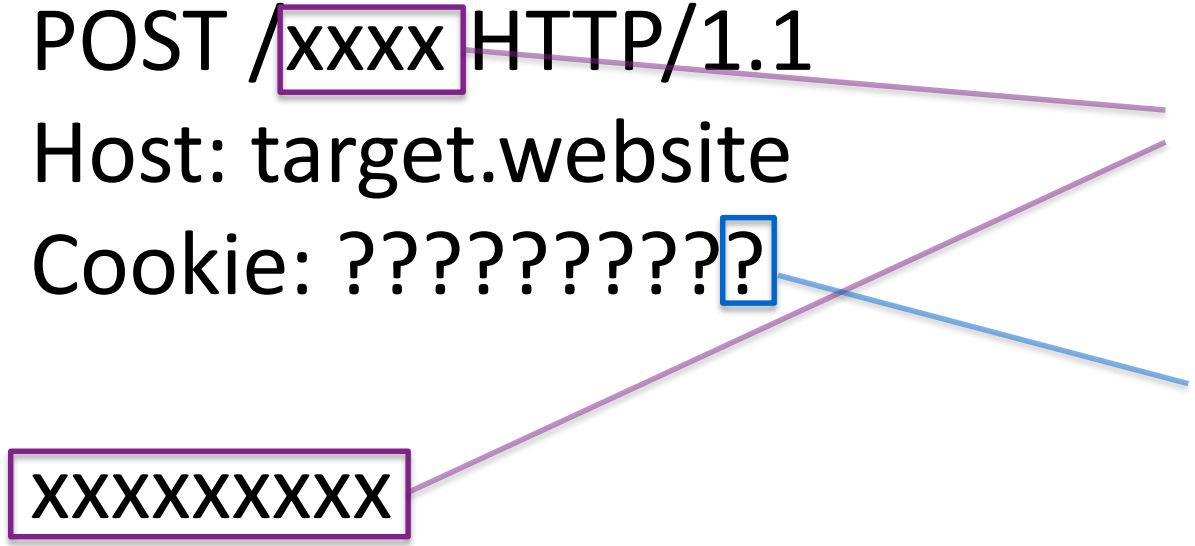


Bob decrypts the last block



- Bob accepts this only if the final byte is “correct”.
- This happens once every 256 times.
- If it fails, the connection is reset and Eve can try again.
- **If it succeeds Eve can compute one byte of the plaintext.**

HTTP(S) requests are predictable



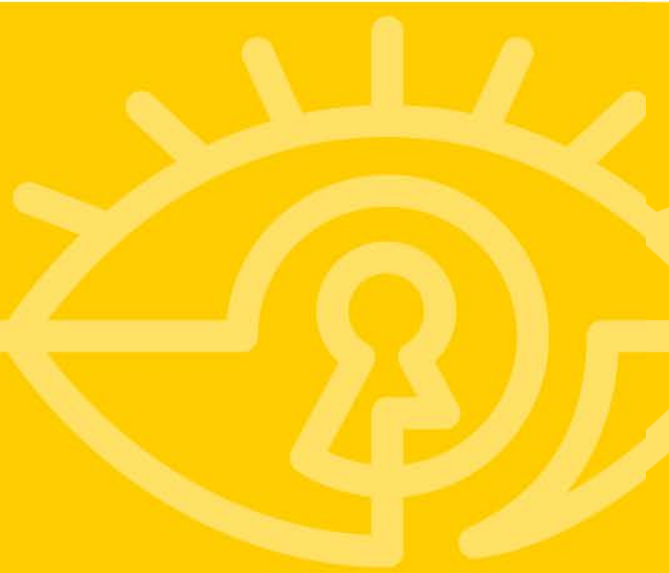
Eve can force this to vary to ensure the padding fills the last block…

…**and** for this to be the final byte of a block.

And thus “construct” the cookie byte by byte.

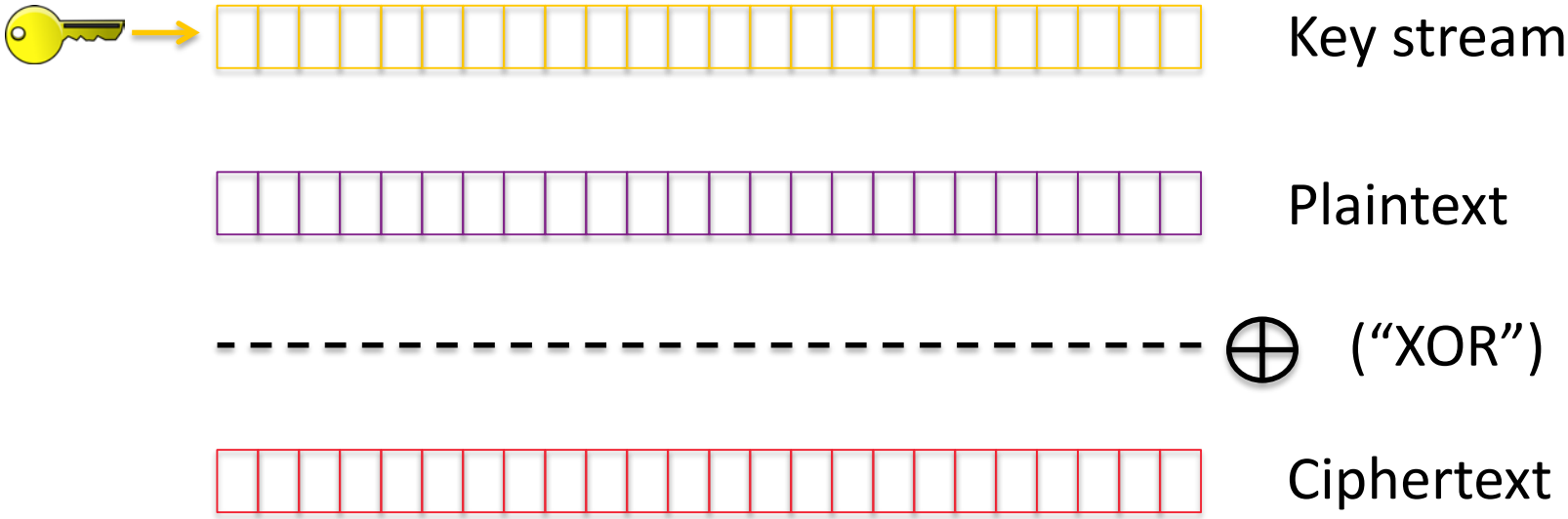
POODLE attack on SSLv3

- Makes it very easy for Eve to compute the session cookie (under reasonable circumstances)
- Does **not** allow Eve to decrypt the full request, or any part of the response.
- Does **not** work against other protocols.
- Session hijacks are very bad. But they usually don't give you the keys to the kingdom.



RC4 and the NOMORE attack

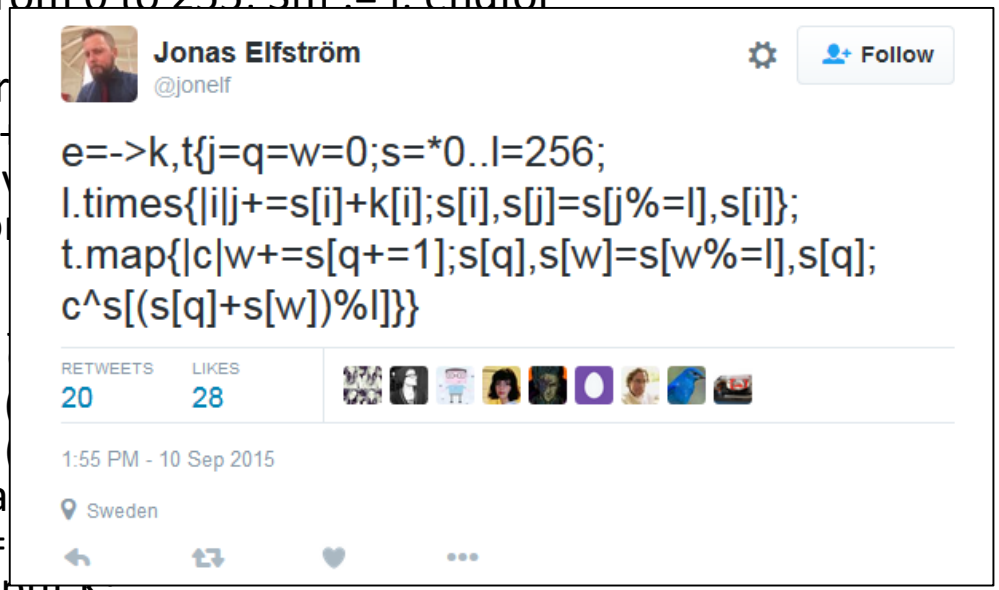
Stream ciphers: RC4



The RC4 algorithm


```
for i from 0 to 255: S[i] := i; endfor
j := 0;
for i fr
j := (j +
swap
endfo

i := 0;
while
    i :=
    j :=
    swa
    K :=
    output K;
endwhile
```



virus BULLETIN

RC4

- Easy to implement, especially in software
- Very popular in malware!
- “Too good to be true” – W. Diffie

A bias in the second byte

- There is a very obvious bias in the second byte of an RC4 keystream: its value is 0 twice as likely as one would expect.
- It takes a few thousand encryptions of the same plaintext for Eve to be certain of the second byte.
- **That is really bad.**
- It is never useful.

More biases

- There are many pairs and triplets of bytes with a small bias.
- It takes much more than a thousand encryptions to know these bytes in the plaintext.

The NOMORE attack

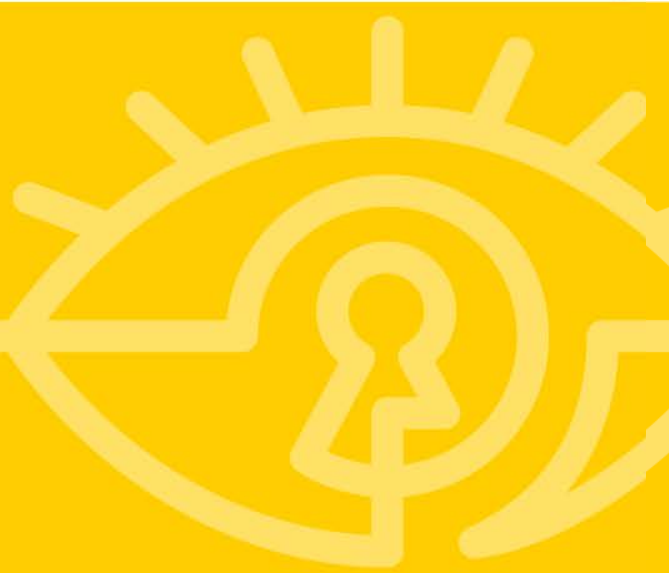
- Eve forces Alice to make the same HTTPS request a **very large** number of times.
- After ~75 hours, she still isn't sure of the plaintext.
- But she has enough candidates for the cookie that she can brute-force the cookie.

The NOMORE attack

- This requires certain conditions about the cookie to be met.
- The attack can **not** be sped up by using fast computers!
- Still only gives you a (session) cookie.
- This will never be used in practice.

Conclusion so far

- Diffie-Hellman is safe, but choose a large and/or unique group.
- Avoid SSLv3 for HTTPS because of POODLE.
- There are better alternatives to RC4, but in general, the algorithm has not been broken.



SHA-1

Hashing algorithms

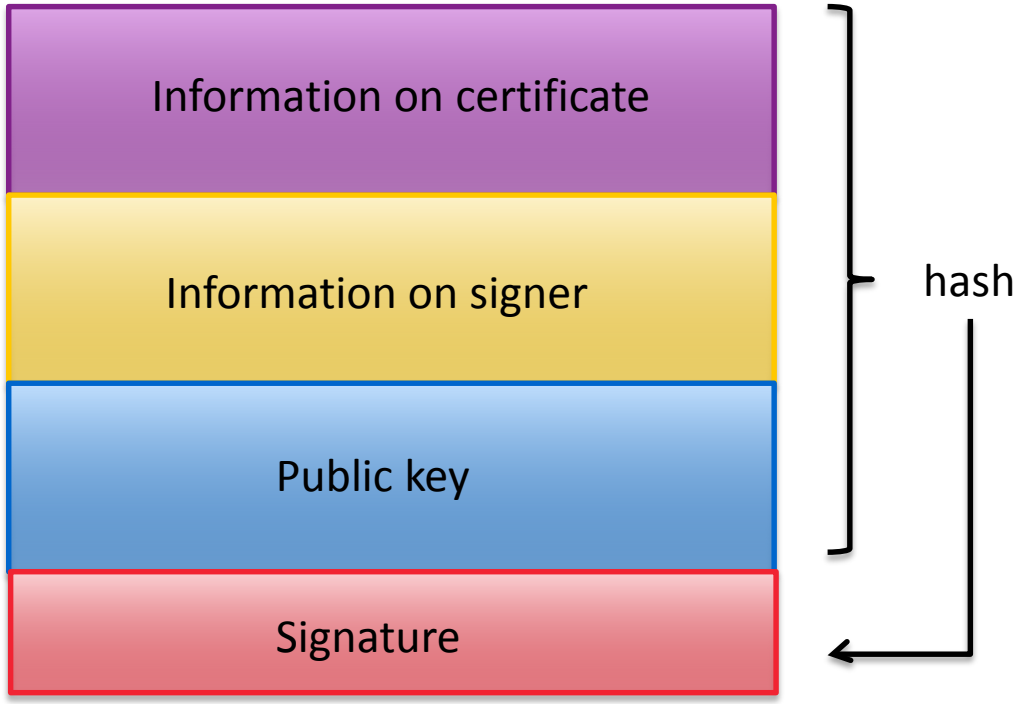
- Turns an arbitrary sequence of bytes into a fixed-length block that gives no information about the original.
- MD5, SHA-1, SHA-256

```
md5(US constitution) = 8fc81ba27fae09ff613b5edbf865fd63

md5("hello world") = 5eb63bbbe01eeed093cb22bb8f5acdc3

md5("Hello world") = 3e25960a79dbc69b674cd4ec67a72c62
```

X.509 certificates



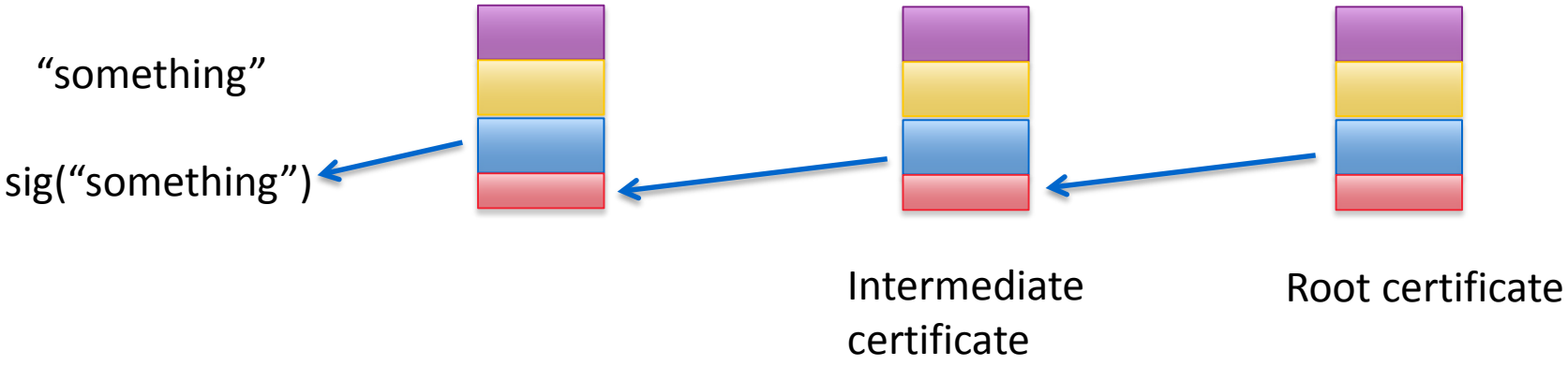
Bob sends to Alice

- His certificate
- “something” unique
- A signature of “something”, signed with the private key corresponding to the public key in his certificate.

Now Alice knows this is the certificate belonging to the person she’s talking to.

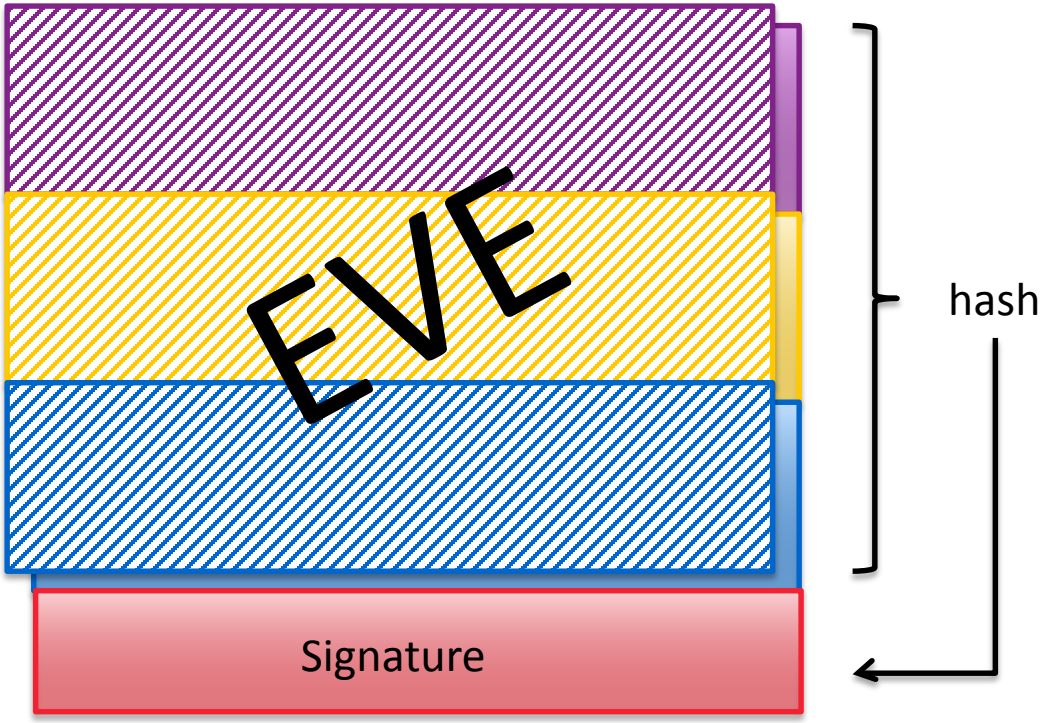
How does she know this is Bob?

Bob authenticates to Alice



The Root certificate is stored on Alice’s computer (or browser).

What could go wrong?



If Eve could forge a certificate with the very same hash as that of Bob's, she could add the signature of Bob's certificate and authenticate as Bob.

From difficult to easy

- Find a certificate with the same SHA-1 hash as a given certificate.

This is what Eve wants

- Find an arbitrary block of bytes with SHA-1 hash value equal to that of another block.
- Find two blocks with the same SHA-1 hash (“collision”).
- Find a hash collision for a weaker form of SHA-1 (“free start collision”).

This is where we are now (Marc Stevens)

Why the panic?

A brief history of MD5:

"The presented attack does not yet threaten practical applications of MD5, but it comes rather close".

Dobbertin, 1996.

"I know, I know."

Flame malware, 2012

Phasing out SHA-1 in certificates

- Certificates can last for many years; we can't just stop accepting them from one day to the next.
- **Upgrading your certificates won't prevent people from impersonating you!**
- The phasing out will last several years.
- Phasing out has already hurt people.
- Flame-like attacks have been mitigated.

virus BULLETIN

Conclusions

Why bother?

- Attacks only get better, they never get worse.
- I may be wrong!

- It shows that you care.

So what is the problem?

- A false sense of insecurity.
- A false sense of security.

A cryptographic protocol is unlikely going to be the weakest link in your organisation!

Is there ever an excuse for weak crypto?

No!

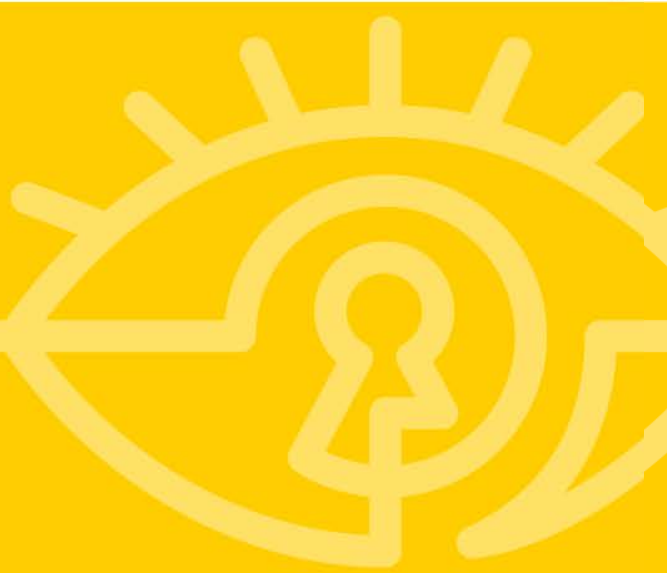
Well, OK but only if

- The alternative is worse
- You really understand the attacks
- You can respond to new threats quickly

Example: *Google* supporting SSL3 for encrypted SMTP.

What to do?

- Do inform yourself about new attacks.
- **Do upgrade to the latest and greatest crypto.**
- Don’t panic.
- Do remember that you undoubtedly have bigger problems than the crypto protocols you are using.



Thank you!

martijn.grooten@virusbtn.com
@martijn_grooten

https://www.lapsedordinary.net/rsa2016/