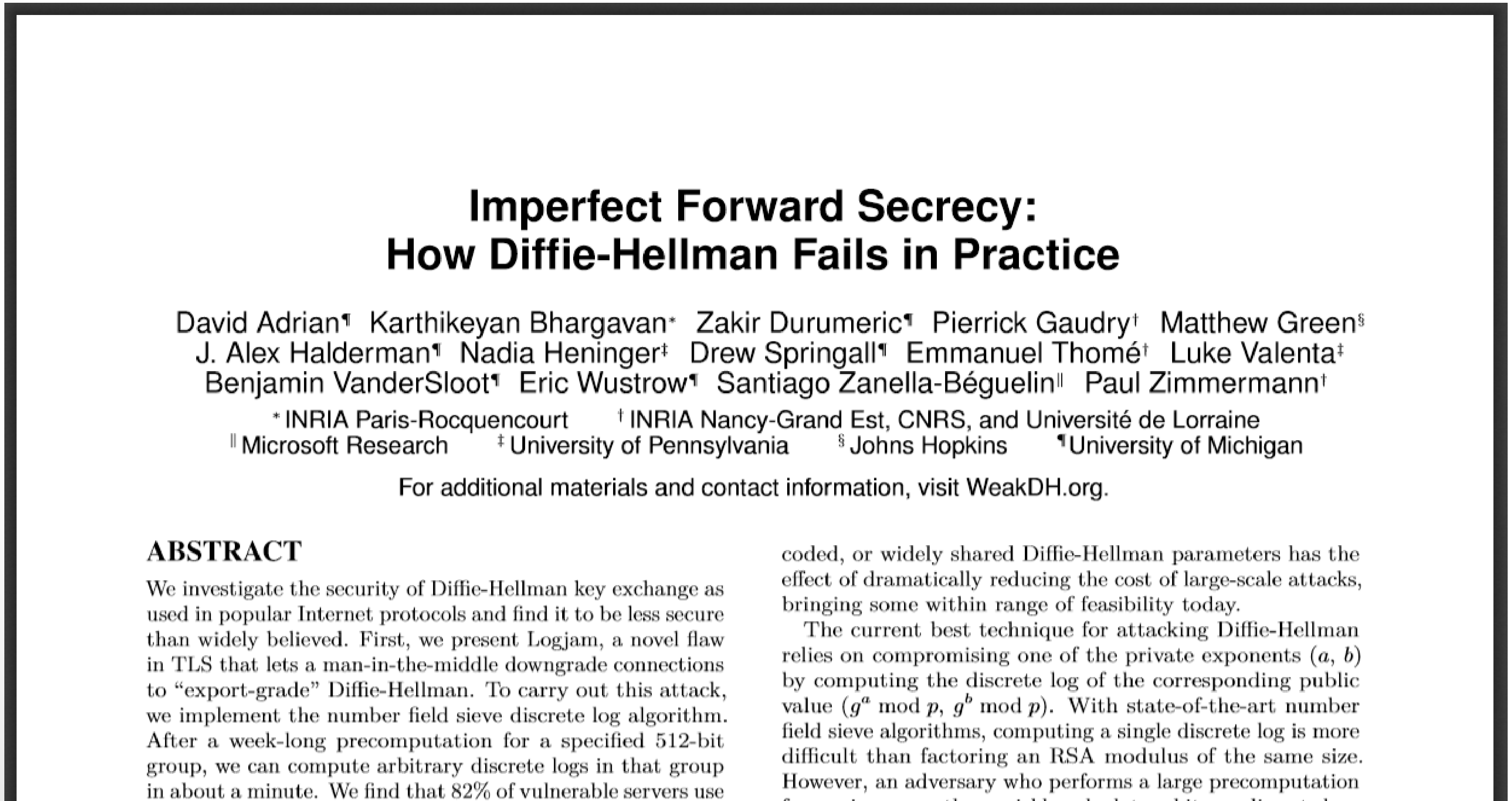
The “WeakDH” Result and its Significance for Security and Privacy

Jason Perry
Lewis University CaMS
December 1, 2015

The work being presented in this talk

ACM CCS 2015 – Best Paper Award



Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice

David Adrian[¶] Karthikeyan Bhargavan[*] Zakir Durumeric[¶] Pierrick Gaudry[†] Matthew Green[§]
J. Alex Halderman[¶] Nadia Heninger[‡] Drew Springall[¶] Emmanuel Thomé[†] Luke Valenta[‡]
Benjamin VanderSloot[¶] Eric Wustrow[¶] Santiago Zanella-Béguelin[‖] Paul Zimmermann[†]

[*] INRIA Paris-Rocquencourt [†] INRIA Nancy-Grand Est, CNRS, and Université de Lorraine
[‖] Microsoft Research [‡] University of Pennsylvania [§] Johns Hopkins [¶] University of Michigan

For additional materials and contact information, visit WeakDH.org.

ABSTRACT

We investigate the security of Diffie-Hellman key exchange as used in popular Internet protocols and find it to be less secure than widely believed. First, we present Logjam, a novel flaw in TLS that lets a man-in-the-middle downgrade connections to "export-grade" Diffie-Hellman. To carry out this attack, we implement the number field sieve discrete log algorithm. After a week-long precomputation for a specified 512-bit group, we can compute arbitrary discrete logs in that group in about a minute. We find that 82% of vulnerable servers use

coded, or widely shared Diffie-Hellman parameters has the effect of dramatically reducing the cost of large-scale attacks, bringing some within range of feasibility today.

The current best technique for attacking Diffie-Hellman relies on compromising one of the private exponents (a, b) by computing the discrete log of the corresponding public value ($g^a \bmod p$, $g^b \bmod p$). With state-of-the-art number field sieve algorithms, computing a single discrete log is more difficult than factoring an RSA modulus of the same size. However, an adversary who performs a large precomputation

Outline

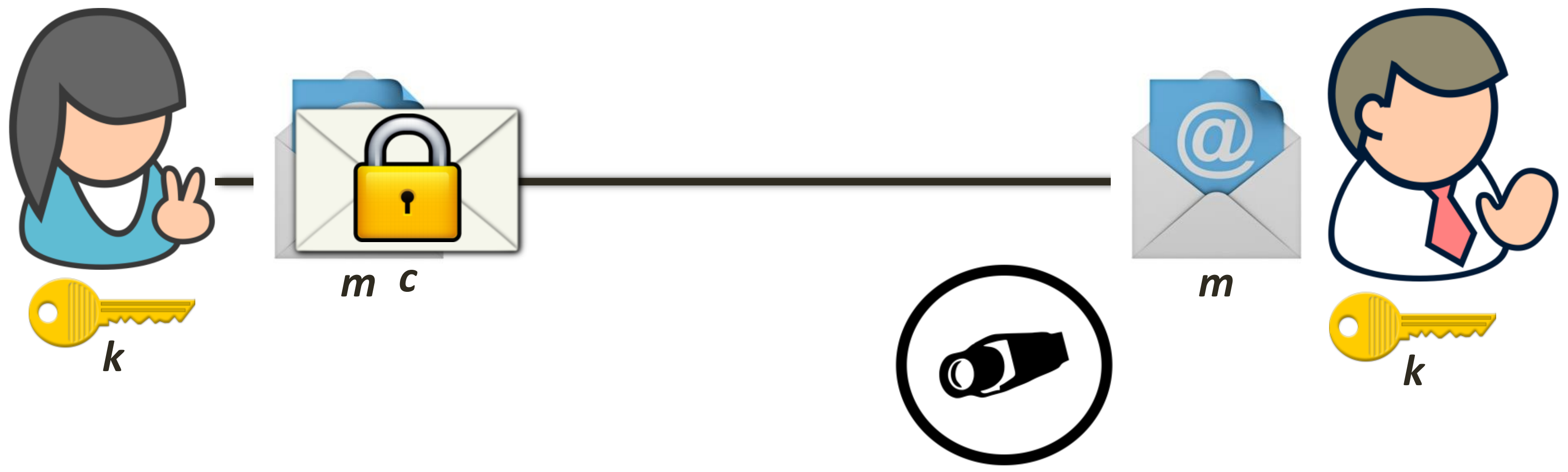
1. Intro: Encryption on the Internet
2. The Diffie-Hellman Key Exchange Protocol
3. Breaking DH: The discrete logarithm problem
4. Result #1: the LOGJAM security degradation attack
5. Result #2: Possible break of 1024-bit DH with state-level resources
6. Conclusion

Encryption on the Internet

- Due to the way the Internet is architected, all traffic is subject to eavesdropping
  - “Packet sniffing”
- Without strong encryption, the world of e-commerce and social media that we enjoy on the internet would not be possible.

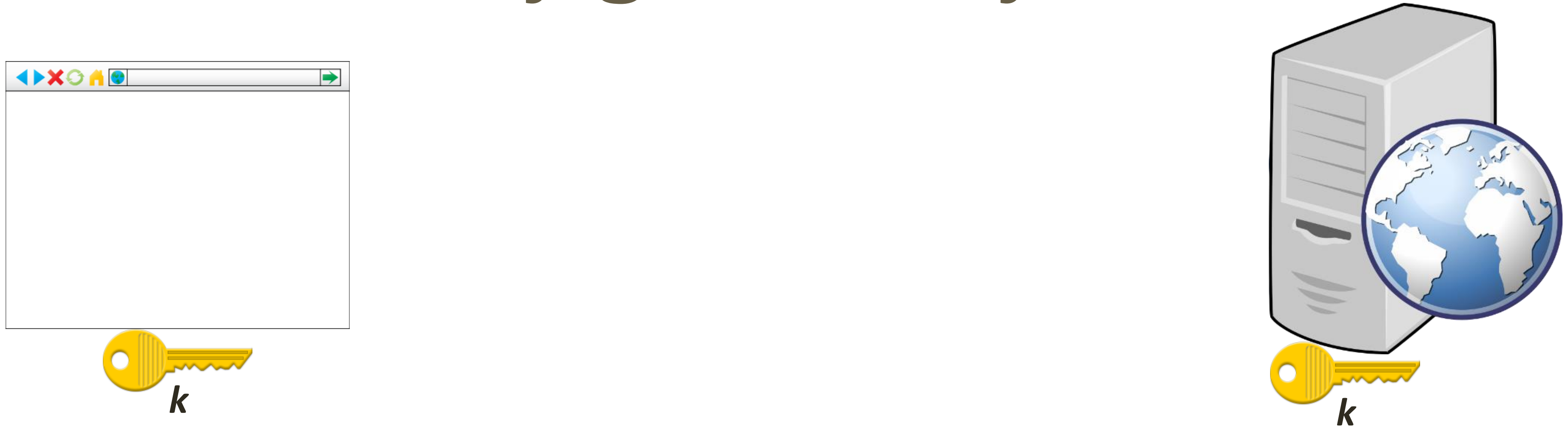
- Most prevalent uses of encryption:
  - A URL beginning with HTTPS indicates that the web browser and website are carrying out encrypted communications.
  - Equally important to many organizations are **VPNs** (virtual private networks), whose traffic is most commonly encrypted using the **IPsec** protocol.
  - Allows secure access to company networks from outside

Overview of Symmetric-key Encryption



- Can be thought of as an envelope with a lock (encryption algorithm) that can only be locked and opened by a key (string of random bits)
- When a pair of parties shares a key *k*, they can send messages to each other that only they can read.
- If the key is recovered by an adversary, all bets are off.

But how do they get the key?



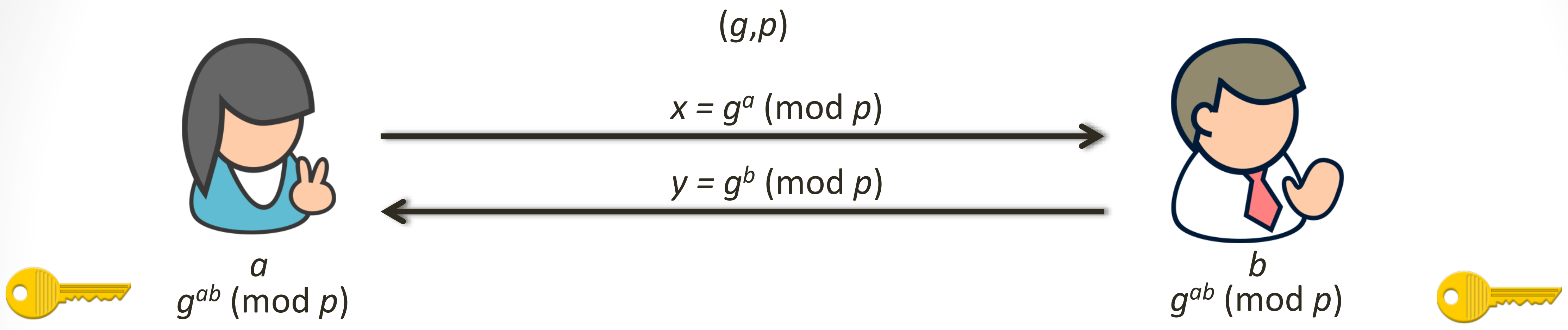
- The two parties need some way of sharing an identical key known to no one else.
- Meeting in person, or establishing a different channel for communicating, is not practical in many internet applications.
- This problem remained unsolved until the advent of *public-key cryptography*.

The Diffie-Hellman Key Exchange Protocol (DH)

W. Diffie and M. E. Hellman, "New Directions in Cryptography," *IEEE Trans. on Info. Theory*, Vol. IT-22, Nov. 1976, pp. 644-654 (Invited Paper).

- The first *published* protocol in the field of public-key cryptography
- Still one of the most widely used algorithms on the internet (HTTPS, VPNs)

DH Basic Operation



1. Alice and Bob decide on two numbers: a generator g and a prime p (both can be known to the public)
2. Alice generates a secret number a, Bob generates a secret number b
3. Alice computes $g^a \pmod{p}$ and sends it to Bob. Bob computes $g^b \pmod{p}$ and sends it to Alice
4. Alice computes $(g^b)^a = g^{ab} \pmod{p}$. Bob computes $(g^a)^b = g^{ab} \pmod{p}$. Now g^{ab} **is the shared secret key**.

What makes it secure?

- An eavesdropper can see both $x = g^a$ and $y = g^b \pmod{p}$, but with these alone it is thought to be too difficult to compute g^{ab}.

- We know nothing better than to take the logarithm of g^a or g^b to get a or b

$$\log_g g^a = a \pmod{p}$$

- In the realm of modular arithmetic, this **discrete logarithm problem (DLP)** is thought to be *computationally intractable*—no efficient algorithm is known or believed to exist.

- Therefore, the security of DH is built on the <u>assumption</u> that DLP is intractable*

*for at least some groups.

How intractable is DLP?

- The time required to compute discrete logarithms is related to the size of *p*, which is typically measured in bit length
  - A source of expressions such as “1024-bit security”, “2048-bit security”

- Best known algorithm for computing discrete logs: the **number field sieve (NFS)** [Gordon ‘92], with time complexity: $\exp\left((1.923 + o(1))(\log p)^{1/3}(\log\log p)^{2/3}\right)$

- Estimated NFS runtimes, in *core-years*, for common sizes of *p*:

Size of p, in bits	Core-years, total
512	10.2
768	36,500
1024	45,000,000

THE “LOGJAM” ATTACK

or, the chickens of 1990s crypto legislation come home to roost

Logjam attack: overview

- Allows a “man-in-the-middle” (MITM) to decrypt HTTPS traffic that uses DH.
- Main idea: Many web servers continue to support weak DH using 512-bit primes. The attack tricks the client’s browser into accepting such a small prime.

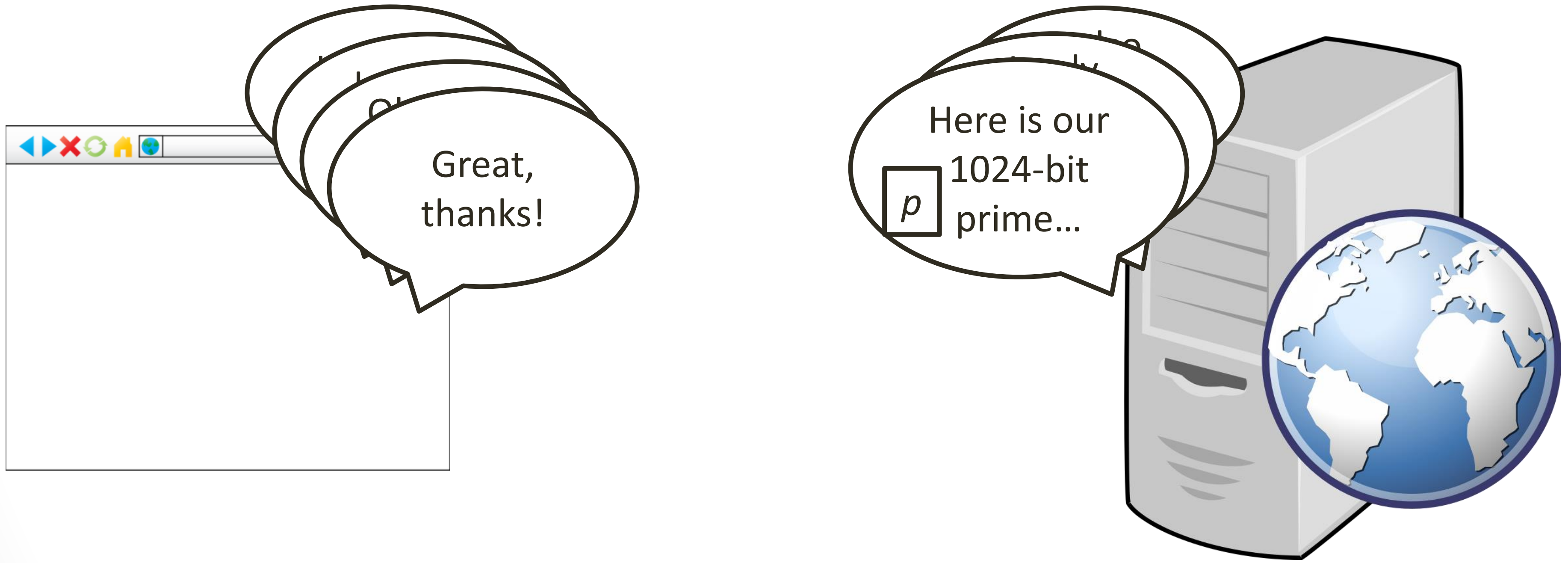
Export-grade cryptography

- In the early 1990’s, the US passed legislation prohibiting cryptography beyond certain strengths from being “exported”, i.e., used in internet communications with international servers.
  - Key-length restrictions were placed on many algorithms
  - DH was restricted to a 512-bit *p*
  - The weakened versions of the algorithms were called *export-grade cryptography.*

- The restrictions were eventually lifted, and browsers stopped supporting the weakened cryptography; However, many web servers still continued to support it for the sake of backward compatibility.
  - Authors found that 8.4% of the top 1 million domains still supported DHE_EXPORT for 512-bit primes

The TLS Handshake

- To support encrypted communications, a client (browser) and server must first negotiate the cryptographic algorithms and key lengths to use.



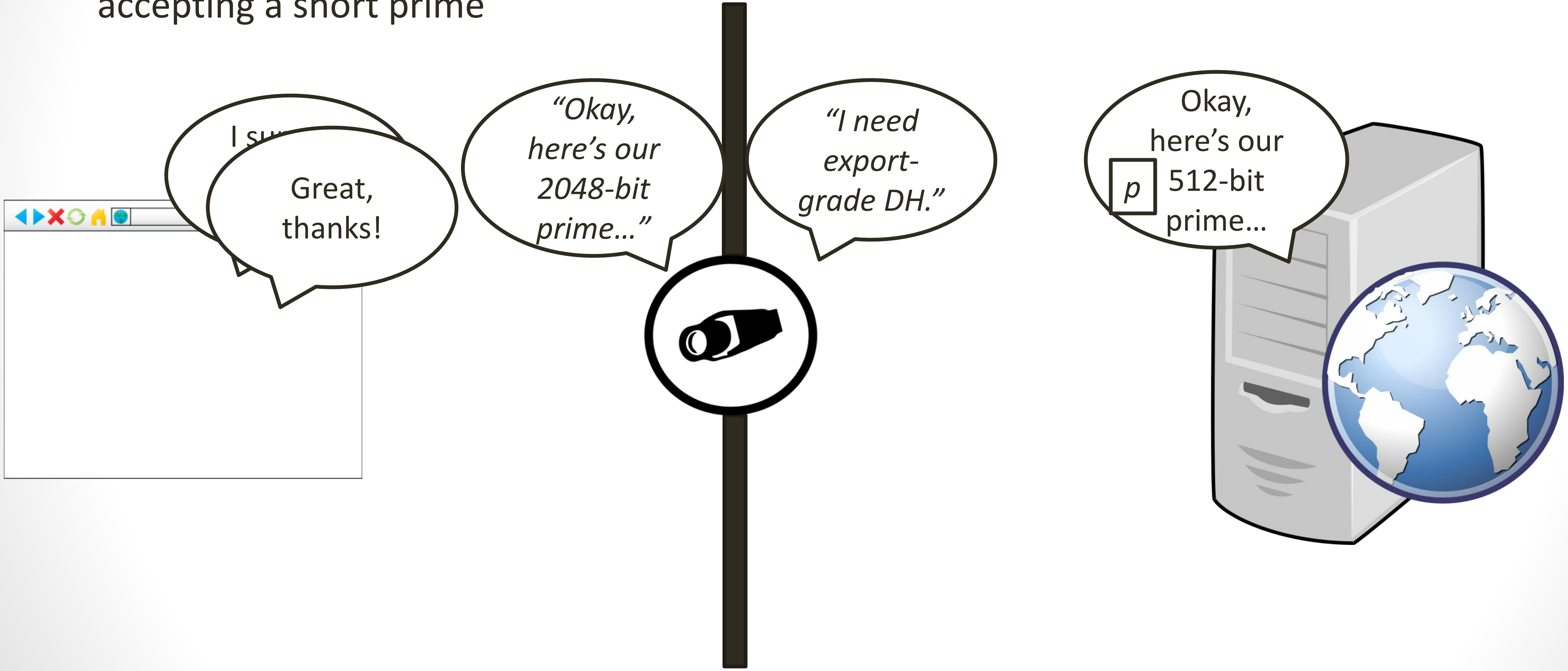
Normal Operation

- If the encryption the server supports is too weak encryption, the browser will refuse to connect.

No way,
too weak!
<click>
I only
support
512-bit DH.

Overview of the Logjam attack

- Man-in-the-middle intercepts traffic in both directions and tricks browser into accepting a short prime


I su
Great, thanks!
"Okay, here's our 2048-bit prime…"
"I need export-grade DH."
Okay, here's our 512-bit prime…
p

Making Logjam work

This works because:

- A flaw in the TLS handshake protocol: server does not sign the initial cryptographic parameters with its certificate—that's why they can be substituted by MITM.
- Browsers didn't check the length of the prime received

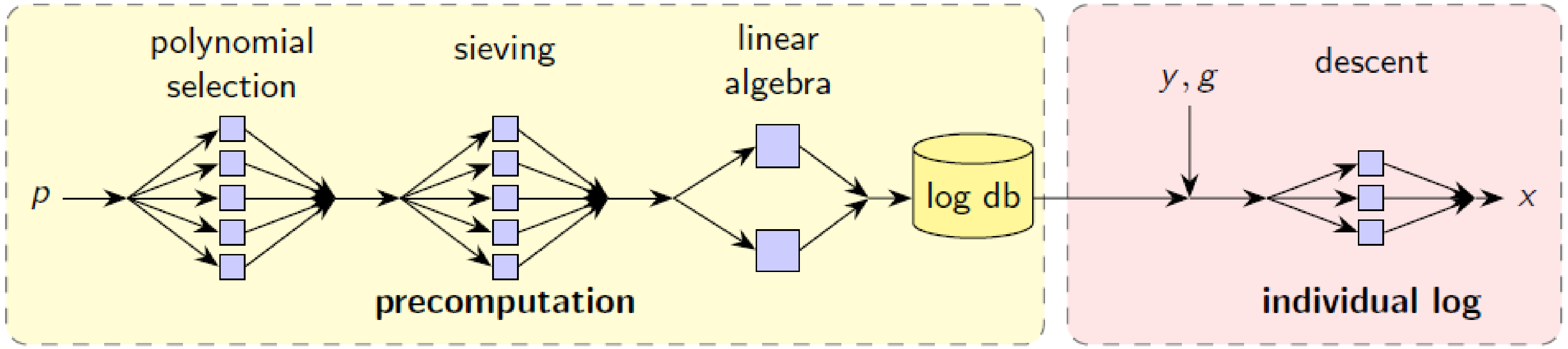
To complete the handshake and decrypt the traffic, the MITM still has to compute the discrete log of $y = g^b \pmod{p}$ very quickly

- Is this possible, even with 512-bit primes?

Size of p, in bits	Core-years, total
512	10.2
768	36,500
1024	45,000,000

Precomputing Discrete Logs

- Recall that the goal is to compute *b* from $y = g^b \bmod p$
- The number field sieve algorithm has three major stages, the first two of which only depend on *p*, not *g* or *y* :



- If a server keeps reusing a prime on which the adversary has performed the precomputation, individual sessions can be broken very quickly by performing only the descent phase.

Precomputing Discrete Logs

Length of *p* in bits	Sieving, core-years	Linear Algebra, core-years	Descent, core-time
512	2.5	7.7	10 mins
768	8,000	28,500	2 days
1024	10,000,000	35,000,000	30 days

- Authors found that 92.5% of all domains that supported DHE_EXPORT used one of *two* 512-bit primes
- Results of 512-bit discrete log experiments:
  - <u>Polynomial selection + sieving</u> on 2000-3000 cores in parallel: 18 hours
  - <u>Linear algebra</u> on 36-node Xeon cluster: 120 hours
    - Total precomputation time for 512-bit *p*: about one week
  - <u>Descent phase</u> on single machine with 2 18-core Xeons: median time of **70 seconds**

Potential Impact of Logjam

- Authors implemented proof-of-concept attack
- Many ways an attacker can work around the delay
- Could be used to decrypt traffic to 8% of top 1M websites

Mitigation of the Logjam vulnerability

- After the attack was published, browsers were updated to reject the short primes.
  - IE, Chrome, Firefox require minimum 1024 bits for DH; Safari: 768.

- Test your browser: weakdh.org

POSSIBLE BREAK OF 1024-BIT DH BY NATION-STATES

Prime reuse for 1024-bit DH

1024-bit DH is still a standard choice for many browsers and web servers and VPNs.

- A single 1024-bit prime was found to be used for **26%** of SSH servers and **66%** of VPNs.
- A second 1024-bit prime was used for **18%** of the top 1M HTTPS websites (Apache 2.2)

Why are servers reusing primes?

- It’s easier and potentially less error-prone to stick with a known good prime.

- *As long as it remains infeasible to break even one of them*, there’s no intrinsic problem with reusing the same *p*, as long as the *a* and *b* are different every time.

….but that may no longer be the case.

Conjecture

- Is it possible that a nation-state-level entity could have invested the time and money needed to do the full NFS precomputation stage for *even one* 1024-bit *p*?

- If so, they could to *passively* decrypt a significant fraction of all web and VPN traffic—no MITM necessary

Authors' estimations

- For sieving, 80x speedup using dedicated chips (ASICs)
  - $8M could buy enough chips to do the sieving phase for one 1024-bit *p* in one year

- Linear algebra phase a bigger unknown
  - Estimated cost of $11B in supercomputers to do this step in 1 year
  - If ASICs could achieve a similar speedup for this phase, then we're in the range of hundreds of millions of dollars—well within the range of a nation-state

Exhibit 1: NSA’s “Black budget”

- US intelligence budget for secret projects FY 2013: $52.6 billion
  - NSA: $10.8 billion, Cryptanalysis and exploitation services: $1.0 billion
  - Summary tables published in Washington Post

Exhibit 2: Statements by insiders

James Bamford, Wired Magazine, 2012:

According to another top official also involved with the program, the NSA made an enormous breakthrough several years ago in its ability to cryptanalyze, or break, unfathomably complex encryption systems employed by not only governments around the world but also many average computer users in the US. The upshot, according to this official: “Everybody’s a target; everybody with communication is a target.”

[..]

The breakthrough was enormous, says the former official, and soon afterward the agency pulled the shade down tight on the project, even within the intelligence community and Congress. “Only the chairman and vice chairman and the two staff directors of each intelligence committee were told about it,” he says. The reason? “They were thinking that this computing breakthrough was going to give them the ability to crack current public encryption.”

Is the US government decrypting 66% of all VPN traffic and 18% of all HTTPS traffic?

Mitigations

- Move to ECC (Elliptic Curve Cryptography)
  - Still DH, but with a different representation of groups, not vulnerable to NFS
- If ECC isn't possible, use at least 2048-bit primes
- If 2048-bit isn't possible, generate a fresh 1024-bit prime

Conclusion

- Diffie-Hellman Key Exchange is not broken; but it is being used with sufficiently weak parameters that powerful parties may already be able to break it and decrypt internet traffic.

- Moral for developers: The most secure cryptographic algorithms don’t do any good if deployed carelessly or incorrectly!

THANK YOU!