You vs. the NSA

Why everybody needs high-security crypto

Peter Schwabe

Radboud University Nijmegen, The Netherlands

October 21, 2013

Florianópolis, Brazil

Cryptography – the very basics

Alice

Bob

- Alice *encrypts* a message M using a key K obtains ciphertext C
- Sends C to Bob

- Bob *decrypts* C using K and obtains M

Let me introduce Eve

- Eve does not know the key K, tries to obtain the message M

Let me introduce Eve

- Eve does not know the key K, tries to obtain the message M
- What can Eve do?
  - Listen on the transmission channel

Let me introduce Eve

- Eve does not know the key K, tries to obtain the message M
- What can Eve do?
  - Listen on the transmission channel
  - Modify messages going over the channel
  - Send messages herself

Let me introduce Eve

- Eve does not know the key K, tries to obtain the message M
- What can Eve do?
  - Listen on the transmission channel
  - Modify messages going over the channel
  - Send messages herself
  - Obtain message-ciphertext pairs encrypted under K

Let me introduce Eve

- Eve does not know the key K, tries to obtain the message M
- What can Eve do?
  - Listen on the transmission channel
  - Modify messages going over the channel
  - Send messages herself
  - Obtain message-ciphertext pairs encrypted under K
  - Massive computations (for example to compute K)

Let me introduce Eve

- Eve does not know the key K, tries to obtain the message M
- What can Eve do?
  - Listen on the transmission channel
  - Modify messages going over the channel
  - Send messages herself
  - Obtain message-ciphertext pairs encrypted under K
  - Massive computations (for example to compute K)
  - More later ...

More cryptography

- How does Bob know that the message comes form Alice?
  Answer: authentication (of users) using a key K_a

More cryptography

- How does Bob know that the message comes form Alice?
  Answer: authentication (of users) using a key K_a
- How does Bob know that the message hasn't been modified?
  Answer: authentication (of the message) using a key K_a

More cryptography

- How does Bob know that the message comes form Alice?
  Answer: authentication (of users) using a key K_a
- How does Bob know that the message hasn't been modified?
  Answer: authentication (of the message) using a key K_a
- How do Alice and Bob get K in the first place?
  Answer: Key-exchange protocols

More cryptography

- How does Bob know that the message comes form Alice?
  Answer: authentication (of users) using a key K_a
- How does Bob know that the message hasn't been modified?
  Answer: authentication (of the message) using a key K_a
- How do Alice and Bob get K in the first place?
  Answer: Key-exchange protocols
- How can Alice send a message such that *everybody* can be sure that *she* sent *that* message?
  Answer: Cryptographic signatures

More cryptography

- How does Bob know that the message comes form Alice?
  Answer: authentication (of users) using a key K_a
- How does Bob know that the message hasn't been modified?
  Answer: authentication (of the message) using a key K_a
- How do Alice and Bob get K in the first place?
  Answer: Key-exchange protocols
- How can Alice send a message such that *everybody* can be sure that *she* sent *that* message?
  Answer: Cryptographic signatures

Eve's goals

- In short: Everything forbidden

More cryptography

- How does Bob know that the message comes form Alice?
  Answer: authentication (of users) using a key K_a
- How does Bob know that the message hasn't been modified?
  Answer: authentication (of the message) using a key K_a
- How do Alice and Bob get K in the first place?
  Answer: Key-exchange protocols
- How can Alice send a message such that *everybody* can be sure that *she* sent *that* message?
  Answer: Cryptographic signatures

Eve's goals

- In short: Everything forbidden
- Impersonate Alice or Bob, forge messages, obtain keys (most powerful attack!)

You (Alice and Bob)

- Average computer user
- Your computing and communication equipment:
  - Laptop (2–3 GHz)
  - Smartphone (1–2 GHz)
- No expert knowledge about cryptography
- Use readily available software

The NSA (Eve)

National Security Agency

- US American secret service
- Largest employer for mathematicians in the world
  - Estimated $40000 - 75000$ employees
  - "Black budget" of US$$52.6$ billion / year
  - Power-bill for Utah data center (estimated): US$$40$ million / year

How secure is cryptography?

Kerckhoffs' principle

An encryption algorithm takes as input a message and a key. The security of the system must rely only on the secrecy of the key, not on the secrecy of the algorithm.

How secure is cryptography?

Kerckhoffs' principle

An encryption algorithm takes as input a message and a key. The security of the system must rely only on the secrecy of the key, not on the secrecy of the algorithm.

- Strongest attack: find the key

How secure is cryptography?

Kerckhoffs' principle

An encryption algorithm takes as input a message and a key. The security of the system must rely only on the secrecy of the key, not on the secrecy of the algorithm.

- Strongest attack: find the key
- Security of the system (simplified): Hardness to find the key
- If the best known algorithm takes 2^n "operations" to find the key, we say that a system is assumed to have n bits of security

How secure is cryptography?

Kerckhoffs' principle

An encryption algorithm takes as input a message and a key. The security of the system must rely only on the secrecy of the key, not on the secrecy of the algorithm.

- Strongest attack: find the key
- Security of the system (simplified): Hardness to find the key
- If the best known algorithm takes 2^n "operations" to find the key, we say that a system is assumed to have n bits of security
- Generic attack against n-bit key: try all possibilities. Cost: 2^n

How secure is cryptography?

Kerckhoffs' principle

An encryption algorithm takes as input a message and a key. The security of the system must rely only on the secrecy of the key, not on the secrecy of the algorithm.

- Strongest attack: find the key
- Security of the system (simplified): Hardness to find the key
- If the best known algorithm takes 2^n "operations" to find the key, we say that a system is assumed to have n bits of security
- Generic attack against n-bit key: try all possibilities. Cost: 2^n
- If a system is believed to have n bits of security, an attacker can break it
  - if he can carry out 2^n operations, or
  - if he knows a better algorithm

How many bits of security has X?

keylength.com

- Various institutions give recommendations based on best known attacks
- NIST (every year)
- ECRYPT (until 2012)
- BSI, ANSSI

How many bits of security has X?

keylength.com

- Various institutions give recommendations based on best known attacks
- NIST (every year)
- ECRYPT (until 2012)
- BSI, ANSSI

Some examples of popular schemes (NIST, 2012)

- **AES-128:** 128 bits
- **RSA-1024:** 80 bits
- **RSA-2048:** 112 bits
- **256-bit elliptic curve:** 128 bits

Can NSA break 128-bit-secure schemes?

- Analysis by Bernstein (slightly modified):
  - How much energy does it take to break AES-128?
  - How much energy do we get?

Can NSA break 128-bit-secure schemes?

- Analysis by Bernstein (slightly modified):
  - How much energy does it take to break AES-128?
  - How much energy do we get?
- Second question first:
  - Sun is radiating $\approx 2^{58}$ watts onto the earth
  - Geothermal energy: $\approx 2^{46}$ watts
  - Gravitation of moon and sun: $\approx 2^{43}$ watts

Can NSA break 128-bit-secure schemes?

- Analysis by Bernstein (slightly modified):
  - How much energy does it take to break AES-128?
  - How much energy do we get?
- Second question first:
  - Sun is radiating $\approx 2^{58}$ watts onto the earth
  - Geothermal energy: $\approx 2^{46}$ watts
  - Gravitation of moon and sun: $\approx 2^{43}$ watts
- First question:
  - Best mass-market chips: $\approx 2^{68}$ bit ops / watt / year
  - Perfect power usage: 2^{126} bit ops / year
  - AES key guess takes 2^{13} bit ops
  - Break key with probability 1: > 30000 years

How about better attacks?

- Many crypto algorithms survived years of intensive study by academic community
- No guarantee that NSA does not know better attacks

How about better attacks?

- Many crypto algorithms survived years of intensive study by academic community
- No guarantee that NSA does not know better attacks
- My guess (for the most important algorithms):
  - NSA *does* know better attacks
  - NSA *does not* know *much* better attacks

How about better attacks?

- Many crypto algorithms survived years of intensive study by academic community
- No guarantee that NSA does not know better attacks
- My guess (for the most important algorithms):
  - NSA *does* know better attacks
  - NSA *does not* know *much* better attacks
- Conservative design: Use large "security gap"

How about better attacks?

- Many crypto algorithms survived years of intensive study by academic community
- No guarantee that NSA does not know better attacks
- My guess (for the most important algorithms):
  - NSA *does* know better attacks
  - NSA *does not* know *much* better attacks
- Conservative design: Use large "security gap"
- Usually also early "warning signs" (experts expressing concerns)
- Example: MD5 was fully broken in 2004, Dobbertin, Bosselaers, and Preneel warned in 1996

How about better attacks?

- Many crypto algorithms survived years of intensive study by academic community
- No guarantee that NSA does not know better attacks
- My guess (for the most important algorithms):
  - NSA *does* know better attacks
  - NSA *does not* know *much* better attacks
- Conservative design: Use large "security gap"
- Usually also early "warning signs" (experts expressing concerns)
- Example: MD5 was fully broken in 2004, Dobbertin, Bosselaers, and Preneel warned in 1996
- Problem: Warnings are often ignored

Broken algorithms I: Dual_EC_DRBG

- Random-number generator (RNG) based on elliptic curves
- Standardized by NIST in NIST SP 800-90A

Broken algorithms I: Dual_EC_DRBG

- Random-number generator (RNG) based on elliptic curves
- Standardized by NIST in NIST SP 800-90A
- Snowden leak: Dual_EC_DRBG contains an NSA backdoor

Broken algorithms I: Dual_EC_DRBG

- Random-number generator (RNG) based on elliptic curves
- Standardized by NIST in NIST SP 800-90A
- Snowden leak: Dual_EC_DRBG contains an NSA backdoor
- Brown 2006: potential for a backdoor
- NIST 2013: Don't use Dual_EC_DRBG
- Not a very popular RNG, because it's slow

Broken algorithms I: Dual_EC_DRBG

- Random-number generator (RNG) based on elliptic curves
- Standardized by NIST in NIST SP 800-90A
- Snowden leak: Dual_EC_DRBG contains an NSA backdoor
- Brown 2006: potential for a backdoor
- NIST 2013: Don't use Dual_EC_DRBG
- Not a very popular RNG, because it's slow
- Used in RSA Security products until 2013

Broken algorithms II: SHA-1

- "Cryptographic hash function", designed for 80 bits of security against "collisions"
- Standard component in secure Internet communication
- 80 bits is bleeding-edge security

Broken algorithms II: SHA-1

- "Cryptographic hash function", designed for 80 bits of security against "collisions"
- Standard component in secure Internet communication
- 80 bits is bleeding-edge security
- Wang, Yin, and Yu in 2005: only 69 bits of security

Broken algorithms II: SHA-1

- "Cryptographic hash function", designed for 80 bits of security against "collisions"
- Standard component in secure Internet communication
- 80 bits is bleeding-edge security
- Wang, Yin, and Yu in 2005: only 69 bits of security
- Wang, Yao, and Yao later in 2005: 63 bits of security

Broken algorithms II: SHA-1

- "Cryptographic hash function", designed for 80 bits of security against "collisions"
- Standard component in secure Internet communication
- 80 bits is bleeding-edge security
- Wang, Yin, and Yu in 2005: only 69 bits of security
- Wang, Yao, and Yao later in 2005: 63 bits of security
- Mendel, Rechberger, Rijmen in 2007: ≈ 61 bits of security

Broken algorithms II: SHA-1

- "Cryptographic hash function", designed for 80 bits of security against "collisions"
- Standard component in secure Internet communication
- 80 bits is bleeding-edge security
- Wang, Yin, and Yu in 2005: only 69 bits of security
- Wang, Yao, and Yao later in 2005: 63 bits of security
- Mendel, Rechberger, Rijmen in 2007: ≈ 61 bits of security
- Stevens 2013: Strategies to go below 61-bit barrier

Broken algorithms II: SHA-1

- "Cryptographic hash function", designed for 80 bits of security against "collisions"
- Standard component in secure Internet communication
- 80 bits is bleeding-edge security
- Wang, Yin, and Yu in 2005: only 69 bits of security
- Wang, Yao, and Yao later in 2005: 63 bits of security
- Mendel, Rechberger, Rijmen in 2007: ≈ 61 bits of security
- Stevens 2013: Strategies to go below 61-bit barrier
- I would be surprised if NSA did not have SHA-1 collisions

Broken algorithms II: SHA-1

- "Cryptographic hash function", designed for 80 bits of security against "collisions"
- Standard component in secure Internet communication
- 80 bits is bleeding-edge security
- Wang, Yin, and Yu in 2005: only 69 bits of security
- Wang, Yao, and Yao later in 2005: 63 bits of security
- Mendel, Rechberger, Rijmen in 2007: ≈ 61 bits of security
- Stevens 2013: Strategies to go below 61-bit barrier
- I would be surprised if NSA did not have SHA-1 collisions
- I would not be surprised if NSA had broken SHA-1 even more

Broken algorithms III: RC4

- "Stream cipher", can use 128-bit key for (targeted) 128-bit security
- Used for about $1/2$ of all SSL/TLS connections

Broken algorithms III: RC4

- "Stream cipher", can use 128-bit key for (targeted) 128-bit security
- Used for about $1/2$ of all SSL/TLS connections
- Fluhrer, Mantin, and Shamir in 2001: first output bytes leak information about key

Broken algorithms III: RC4

- “Stream cipher”, can use 128-bit key for (targeted) 128-bit security
- Used for about $1/2$ of all SSL/TLS connections
- Fluhrer, Mantin, and Shamir in 2001: first output bytes leak information about key
- Klein 2005: More output-key correlations
- This attack was used to crack WEP in < 1 minute

Broken algorithms III: RC4

- "Stream cipher", can use 128-bit key for (targeted) 128-bit security
- Used for about $1/2$ of all SSL/TLS connections
- Fluhrer, Mantin, and Shamir in 2001: first output bytes leak information about key
- Klein 2005: More output-key correlations
- This attack was used to crack WEP in < 1 minute
- AlFardan, Bernstein, Paterson, Poettering, Schuldt in 2013: TLS plaintext recovery in $< 2^{34}$
- Attack is based on statistical analysis of a lot of RC4 output

Broken algorithms III: RC4

- “Stream cipher”, can use 128-bit key for (targeted) 128-bit security
- Used for about $1/2$ of all SSL/TLS connections
- Fluhrer, Mantin, and Shamir in 2001: first output bytes leak information about key
- Klein 2005: More output-key correlations
- This attack was used to crack WEP in < 1 minute
- AlFardan, Bernstein, Paterson, Poettering, Schuldt in 2013: TLS plaintext recovery in $< 2^{34}$
- Attack is based on statistical analysis of a lot of RC4 output
- I would be surprised if NSA did not have good RC4 attacks

Attacking implementations

- What if the math is right, but the *implementation* of the math is not?
- Programmers make mistakes, some crypto is hard to get right

Attacking implementations

- What if the math is right, but the *implementation* of the math is not?
- Programmers make mistakes, some crypto is hard to get right
- About 40% of SSL/TLS servers uses Microsoft CryptoAPI
- Another big target: OpenSSL library

Attacking implementations

- What if the math is right, but the *implementation* of the math is not?
- Programmers make mistakes, some crypto is hard to get right
- About 40% of SSL/TLS servers uses Microsoft CryptoAPI
- Another big target: OpenSSL library
- A practical attack against one of these *implementations* breaks a lot!

Side channels

Side channels

- So far: attacker could see inputs and outputs
- Attackers can see more:
  - power consumption,
  - electromagnetic radiation
  - timing (even remotely!)

Side channels

- So far: attacker could see inputs and outputs
- Attackers can see more:
    - power consumption,
    - electromagnetic radiation
    - timing (even remotely!)
- *Side-channel attacks*: Use this data to break cryptographic protection

Side channels

- So far: attacker could see inputs and outputs
- Attackers can see more:
  - power consumption,
  - electromagnetic radiation
  - timing (even remotely!)
- *Side-channel attacks*: Use this data to break cryptographic protection
- Side-channel attacks also target specific *implementations*

Timing attacks

- Most scary for Internet communication: (remote) timing attacks
- Two main sources for timing-attack vulnerabilities
  - `if` statement with secret condition
  - load from or store to secret address

Timing attacks

- Most scary for Internet communication: (remote) timing attacks
- Two main sources for timing-attack vulnerabilities
  - `if` statement with secret condition
  - load from or store to secret address
- We can remove such vulnerabilities (“constant-time software”)

Timing attacks

- Most scary for Internet communication: (remote) timing attacks
- Two main sources for timing-attack vulnerabilities
  - `if` statement with secret condition
  - load from or store to secret address
- We can remove such vulnerabilities ("constant-time software")
- Performance penalty:
  - Can be huge (e.g., AES on 32-bit platforms)
  - Can be close to zero (e.g., Salsa20)

Timing attacks

- Most scary for Internet communication: (remote) timing attacks
- Two main sources for timing-attack vulnerabilities
  - `if` statement with secret condition
  - load from or store to secret address
- We can remove such vulnerabilities ("constant-time software")
- Performance penalty:
  - Can be huge (e.g., AES on 32-bit platforms)
  - Can be close to zero (e.g., Salsa20)
- For many algorithms it is *hard* to write (efficient) constant-time software
- Most cryptographic software in use today leaks secret data through timing information

Practical timing attacks

Linux hard-disk encryption

- Osvik, Shamir, and Tromer in 2006: timing attack against `dmcrypt`
- Attack took 65 ms to recover the AES-256 key
- Needs attacker process on the same machine

Practical timing attacks

Linux hard-disk encryption

- Osvik, Shamir, and Tromer in 2006: timing attack against `dmcrypt`
- Attack took 65 ms to recover the AES-256 key
- Needs attacker process on the same machine

OpenSSL ECDSA

- Brumley and Tuveri in 2011: *Remote* timing attack against OpenSSL ECDSA
- A few minutes to steal the key over the network

Practical timing attacks

Linux hard-disk encryption

- Osvik, Shamir, and Tromer in 2006: timing attack against `dmcrypt`
- Attack took 65 ms to recover the AES-256 key
- Needs attacker process on the same machine

OpenSSL ECDSA

- Brumley and Tuveri in 2011: *Remote* timing attack against OpenSSL ECDSA
- A few minutes to steal the key over the network

AES-CBC in TLS

- AlFardan and Kenneth G. Paterson in 2013:
  Plaintext recovery attack against TLS with AES-CBC

  > *"we expect all implementations – whether open or closed – to be vulnerable to our attacks to some extent."*

Practical timing attacks

Linux hard-disk encryption

- Osvik, Shamir, and Tromer in 2006: timing attack against `dmcrypt`
- Attack took 65 ms to recover the AES-256 key
- Needs attacker process on the same machine

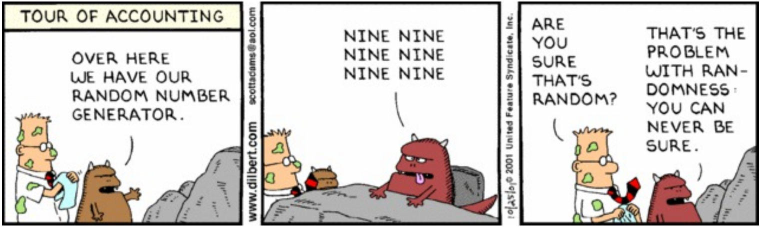
OpenSSL ECDSA

- Brumley and Tuveri in 2011: *Remote* timing attack against OpenSSL ECDSA
- A few minutes to steal the key over the network

AES-CBC in TLS

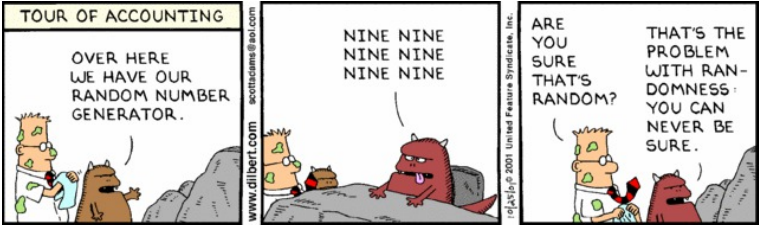
- AlFardan and Kenneth G. Paterson in 2013:
  Plaintext recovery attack against TLS with AES-CBC
- Many implementations have been fixed by now, see, e.g.
  https://www.imperialviolet.org/2013/02/04/luckythirteen.html

Randomness



- Most cryptographic algorithms need randomness
- Some algorithms only have to generate random keys
- Some algorithms need randomness for every message

Randomness



- Most cryptographic algorithms need randomness
- Some algorithms only have to generate random keys
- Some algorithms need randomness for every message
- Bad-randomness attack: guess the right value

Bad randomness I

Debian randomness disaster

- Bello in 2008: Debian/Ubuntu OpenSSL keys have only 15 bits of entropy
- Only 32768 possible keys, can be guessed in < 1 second
- Debian developer had removed on line of randomness-generating code in 2006

Bad randomness I

Debian randomness disaster

- Bello in 2008: Debian/Ubuntu OpenSSL keys have only 15 bits of entropy
- Only 32768 possible keys, can be guessed in < 1 second
- Debian developer had removed on line of randomness-generating code in 2006

Sony randomness disaster

- "Bushing", Cantero, Boessenkool, Peter in 2010: Sony ignored ECDSA requirement of new randomness for each signature
- Signatures leaked PlayStation 3 code-signing key

Bad randomness II

Internet host randomness

- Heninger, Durumeric, Wustrow, Halderman in 2012: Obtain millions of TLS and SSH public keys
- Compute private keys for 0.5% of TLS and 1.06% of SSH public keys
- Reason: lack of randomness during key generation

Bad randomness II

Internet host randomness

- Heninger, Durumeric, Wustrow, Halderman in 2012: Obtain millions of TLS and SSH public keys
- Compute private keys for 0.5% of TLS and 1.06% of SSH public keys
- Reason: lack of randomness during key generation

Taiwanese citizen cards

- Bernstein, Chang, Cheng, Chou, Heninger, Lange, and van Someren in 2013: Obtain public keys from Taiwanese "Citizen Digital Certificate" database
- Compute private keys of 184 Taiwanese citizens
- Reason: lack of randomness during key generation

High-security crypto

Required for secure internet communication

- At least 128 bits of security against all known attacks

High-security crypto

Required for secure internet communication

- At least 128 bits of security against all known attacks
- Full protection against timing attacks

High-security crypto

Required for secure internet communication

- At least 128 bits of security against all known attacks
- Full protection against timing attacks
- Sensible handling of randomness

High-security crypto

Required for secure internet communication

- At least 128 bits of security against all known attacks
- Full protection against timing attacks
- Sensible handling of randomness
- Fast on a broad variety of platforms

High-security crypto

Required for secure internet communication

- At least 128 bits of security against all known attacks
- Full protection against timing attacks
- Sensible handling of randomness
- Fast on a broad variety of platforms
- Open source

NaCl (advertisement)

- Networking and Cryptography library (NaCl, pronounced “salt”)
- Offers all security features from previous slide
- Focus on protecting Internet communication
- Core development team: Daniel J. Bernstein, Tanja Lange, Peter Schwabe
- Acknowledgment: Contributions by
  - Matthew Dempsky (Mochi Media)
  - Niels Duif (TU Eindhoven)
  - Emilia Käsper (KU Leuven, now Google)
  - Adam Langley (Google)
  - Bo-Yin Yang (Academia Sinica)

NaCl (advertisement)

- Networking and Cryptography library (NaCl, pronounced "salt")
- Offers all security features from previous slide
- Focus on protecting Internet communication
- Core development team: Daniel J. Bernstein, Tanja Lange, Peter Schwabe
- Acknowledgment: Contributions by
  - Matthew Dempsky (Mochi Media)
  - Niels Duif (TU Eindhoven)
  - Emilia Käsper (KU Leuven, now Google)
  - Adam Langley (Google)
  - Bo-Yin Yang (Academia Sinica)
- User's perspective: Bundle of functionalities rather than bundle of algorithms

NaCl (advertisement)

- Networking and Cryptography library (NaCl, pronounced "salt")
- Offers all security features from previous slide
- Focus on protecting Internet communication
- Core development team: Daniel J. Bernstein, Tanja Lange, Peter Schwabe
- Acknowledgment: Contributions by
  - Matthew Dempsky (Mochi Media)
  - Niels Duif (TU Eindhoven)
  - Emilia Käsper (KU Leuven, now Google)
  - Adam Langley (Google)
  - Bo-Yin Yang (Academia Sinica)
- User's perspective: Bundle of functionalities rather than bundle of algorithms
- Available (public domain) at

http://nacl.cr.yp.to

Man in the middle

An https session (highly simplified)

- Browser connects to Server
- Server sends its public key
- Browser uses this public key to transmit session key
- Secure communication happens

Man in the middle

An https session (highly simplified)

- Browser connects to Server
- Server sends its public key
- Browser uses this public key to transmit session key
- Secure communication happens

- Question: How do I know that the public key belongs to the right server?
  Answer: It is certified by a *Certificate Authority* (CA)

Man in the middle

An https session (highly simplified)

- Browser connects to Server
- Server sends its public key
- Browser uses this public key to transmit session key
- Secure communication happens

- Question: How do I know that the public key belongs to the right server?
  Answer: It is certified by a *Certificate Authority* (CA)
- Browsers automatically verify whether the certificate is OK

Man in the middle

An https session (highly simplified)

- Browser connects to Server
- Server sends its public key
- Browser uses this public key to transmit session key
- Secure communication happens

- Question: How do I know that the public key belongs to the right server?
  Answer: It is certified by a *Certificate Authority* (CA)
- Browsers automatically verify whether the certificate is OK
- "OK" means: **You trust the CA**

Man in the middle

An https session (highly simplified)

- Browser connects to Server
- Server sends its public key
- Browser uses this public key to transmit session key
- Secure communication happens

- Question: How do I know that the public key belongs to the right server?
  Answer: It is certified by a *Certificate Authority* (CA)
- Browsers automatically verify whether the certificate is OK
- "OK" means: **You trust the CA**
- So, who exactly do you trust? Let's take a look…

Man in the middle

An https session (highly simplified)

- Browser connects to Server
- Server sends its public key
- Browser uses this public key to transmit session key
- Secure communication happens

- Question: How do I know that the public key belongs to the right server?
  Answer: It is certified by a *Certificate Authority* (CA)
- Browsers automatically verify whether the certificate is OK
- "OK" means: **You trust the CA**
- So, who exactly do you trust? Let's take a look...
- Compromise just one CA and you can do anything

Traffic data

- Why break your crypto, just record "meta data"
- Who communicated with whom, when, from where, and how long

Traffic data

- Why break your crypto, just record "meta data"
- Who communicated with whom, when, from where, and how long
- This may tell you more than the content does
- Nice example by Cindy Cohn (EFF) at Crypto 2013:

  > *"We see that you're standing on the Golden Gate Bridge calling the suicide hotline, but we don't know what you're talking."*

Traffic data

- Why break your crypto, just record "meta data"
- Who communicated with whom, when, from where, and how long
- This may tell you more than the content does
- Nice example by Cindy Cohn (EFF) at Crypto 2013:

  > *"We see that you're standing on the Golden Gate Bridge calling the suicide hotline, but we don't know what you're talking."*

- The term "meta data" makes this information sound harmless, it's not!

Traffic data

- Why break your crypto, just record "meta data"
- Who communicated with whom, when, from where, and how long
- This may tell you more than the content does
- Nice example by Cindy Cohn (EFF) at Crypto 2013:

  > *"We see that you're standing on the Golden Gate Bridge calling the suicide hotline, but we don't know what you're talking."*

- The term "meta data" makes this information sound harmless, it's not!
- You don't need the NSA for that, consider the EU

More on traffic data

EU's Data Retention Directive

"Member States shall ensure that the categories of data specified in Article 5 are retained for periods of not less than six months and not more than two years from the date of the communication."

More on traffic data

EU's Data Retention Directive

"Member States shall ensure that the categories of data specified in Article 5 are retained for periods of not less than six months and not more than two years from the date of the communication."

From Article 5:

- data necessary to trace and identify the source of a communication
- data necessary to identify the destination of a communication

More on traffic data

EU's Data Retention Directive

"Member States shall ensure that the categories of data specified in Article 5 are retained for periods of not less than six months and not more than two years from the date of the communication."

From Article 5:

- data necessary to trace and identify the source of a communication
- data necessary to identify the destination of a communication
- data necessary to identify the date, time and duration of a communication

More on traffic data

EU's Data Retention Directive

> *"Member States shall ensure that the categories of data specified in Article 5 are retained for periods of not less than six months and not more than two years from the date of the communication."*

From Article 5:

- data necessary to trace and identify the source of a communication
- data necessary to identify the destination of a communication
- data necessary to identify the date, time and duration of a communication
- data necessary to identify the type of communication

More on traffic data

EU's Data Retention Directive

"Member States shall ensure that the categories of data specified in Article 5 are retained for periods of not less than six months and not more than two years from the date of the communication."

From Article 5:

- data necessary to trace and identify the source of a communication
- data necessary to identify the destination of a communication
- data necessary to identify the date, time and duration of a communication
- data necessary to identify the type of communication
- data necessary to identify users' communication equipment or what purports to be their equipment

More on traffic data

EU's Data Retention Directive

> *"Member States shall ensure that the categories of data specified in Article 5 are retained for periods of not less than six months and not more than two years from the date of the communication."*

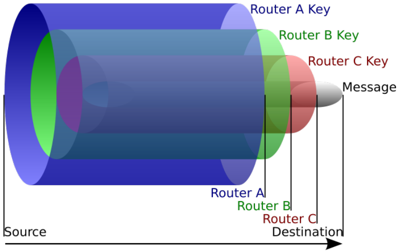
From Article 5:

- data necessary to trace and identify the source of a communication
- data necessary to identify the destination of a communication
- data necessary to identify the date, time and duration of a communication
- data necessary to identify the type of communication
- data necessary to identify users' communication equipment or what purports to be their equipment
- data necessary to identify the location of mobile communication equipment

Anonymization with TOR

How can we hide traffic data?

- Most popular: TOR ("The Onion Router")
- Route data through (at least) three TOR nodes
- Use multiple layers of encryption:



- Open-source software available at http://torproject.org

"TOR stinks"

- Snowden leaked NSA slides about TOR ("TOR stinks")
- Statement by NSA:

  > " *We will never be able to de-anonymize all Tor users all the time.* "

“TOR stinks”

- Snowden leaked NSA slides about TOR (“TOR stinks”)
- Statement by NSA:

  > “ *We will never be able to de-anonymize all Tor users all the time.* ”

- Sounds good, but slides are from 2012, based on 2007 data
- How about today?

Attacks against TOR part I

Looking at the exit node

- Data that comes out of the exit node unencrypted

Attacks against TOR part I

Looking at the exit node

- Data that comes out of the exit node unencrypted
- What if you access a website through TOR and type there

  "Hi, I'm Peter Schwabe, I'm sitting in Flórianopolis, Brazil.
  My IP address is 187.65.227.71."

Attacks against TOR part I

Looking at the exit node

- Data that comes out of the exit node unencrypted
- What if you access a website through TOR and type there

  "Hi, I'm Peter Schwabe, I'm sitting in Flórianopolis, Brazil. My IP address is 187.65.227.71."
- It can be more subtle: look for TOR users when they are not using TOR
- NSA on such attacks: *"Dumb Users (EPICFAIL)"*

Attacks against TOR part II

Controlling TOR nodes

- Attacker tries to control all nodes of a route
- NSA is known to run TOR nodes, unclear how many
- If NSA controls just 1% of the nodes, each route has a $1/1000000$ chance of being NSA controlled

Attacks against TOR part II

Controlling TOR nodes

- Attacker tries to control all nodes of a route
- NSA is known to run TOR nodes, unclear how many
- If NSA controls just 1% of the nodes, each route has a $1/1000000$ chance of being NSA controlled
- TOR changes routes every 15 minutes
- A matter of time until you're de-anonymized

Attacks against TOR part II

Controlling TOR nodes

- Attacker tries to control all nodes of a route
- NSA is known to run TOR nodes, unclear how many
- If NSA controls just 1% of the nodes, each route has a $1/1000000$ chance of being NSA controlled
- TOR changes routes every 15 minutes
- A matter of time until you're de-anonymized
- TOR has some ways to address these attacks

Attacks against TOR part III

"Tor ist tot. Tor basiert auf der Annahme, dass der Gegner nicht in der Lage ist, das gesamte Internet zu überwachen."
– Felix "Fefe" von Leitner (Aug 5, 2013)

Attacks against TOR part III

"Tor ist tot. Tor basiert auf der Annahme, dass der Gegner nicht in der Lage ist, das gesamte Internet zu überwachen."
– Felix "Fefe" von Leitner (Aug 5, 2013)

Timing analysis of traffic

- Observe large amounts of Internet communication
- In particular: Traffic entering TOR network and exiting TOR network
- Use timing correlation to de-anonymize users
- In 2007 apparently infeasible for NSA

Attacks against TOR part IV

Breaking the crypto

- Main crypto components in TOR:
  - AES-128
  - RSA-1024

Attacks against TOR part IV

Breaking the crypto

- Main crypto components in TOR:
  - AES-128
  - RSA-1024
- Estimates by Shamir and Tromer in 2003: Breaking RSA-1024 in one year costs US$10,000,000$

Attacks against TOR part IV

Breaking the crypto

- Main crypto components in TOR:
  - AES-128
  - RSA-1024
- Estimates by Shamir and Tromer in 2003: Breaking RSA-1024 in one year costs US$10,000,000
- RSA-Labs und U.S. government: RSA-1024 only until 2010

Attacks against TOR part IV

Breaking the crypto

- Main crypto components in TOR:
  - AES-128
  - RSA-1024
- Estimates by Shamir and Tromer in 2003: Breaking RSA-1024 in one year costs US$$10,000,000$
- RSA-Labs und U.S. government: RSA-1024 only until 2010
- RSA-768 was broken in 2010, estimate: RSA-1024 is 1000 times harder

Attacks against TOR part IV

Breaking the crypto

- Main crypto components in TOR:
  - AES-128
  - RSA-1024
- Estimates by Shamir and Tromer in 2003: Breaking RSA-1024 in one year costs US$$10,000,000$
- RSA-Labs und U.S. government: RSA-1024 only until 2010
- RSA-768 was broken in 2010, estimate: RSA-1024 is 1000 times harder
- Summary:
  - It is believed that NSA can break RSA-1024
  - It is still hard to do on big scale

Attacks against TOR part IV

Breaking the crypto

- Main crypto components in TOR:
  - AES-128
  - RSA-1024
- Estimates by Shamir and Tromer in 2003: Breaking RSA-1024 in one year costs US$10,000,000$
- RSA-Labs und U.S. government: RSA-1024 only until 2010
- RSA-768 was broken in 2010, estimate: RSA-1024 is 1000 times harder
- Summary:
  - It is believed that NSA can break RSA-1024
  - It is still hard to do on big scale
- Good news: TOR is updating to 128-bit secure Curve25519

Wild speculation part I

Hardware trojans

- Your laptop is running on an Intel or AMD CPU
- Your smartphone has a Freescale, Qualcomm, Apple, TI, ... CPU
- Do you trust these US American companies to not have a trojan in the hardware?

Wild speculation part I

Hardware trojans

- Your laptop is running on an Intel or AMD CPU
- Your smartphone has a Freescale, Qualcomm, Apple, TI, ... CPU
- Do you trust these US American companies to not have a trojan in the hardware?
- Intel CPUs now come with hardware RNG `rdrand`
- Intel "User Manual for the Rdrand Library": Use `rdrand` directly, don't rely on operating system

Wild speculation part I

Hardware trojans

- Your laptop is running on an Intel or AMD CPU
- Your smartphone has a Freescale, Qualcomm, Apple, TI, ... CPU
- Do you trust these US American companies to not have a trojan in the hardware?
- Intel CPUs now come with hardware RNG `rdrand`
- Intel "User Manual for the Rdrand Library": Use `rdrand` directly, don't rely on operating system
- David Johnston (Intel):

> *"...eliminate software PRNGs. Just use the output of the RDRAND instruction wherever you need a random number."*
>
> *"...we did RdRand the way we did, to bypass the OS, libraries, APIs, VMs, caches and memory and feed entropy directly to the register space of the running application."*

Wild speculation part I

Hardware trojans

- Your laptop is running on an Intel or AMD CPU
- Your smartphone has a Freescale, Qualcomm, Apple, TI, ... CPU
- Do you trust these US American companies to not have a trojan in the hardware?
- Intel CPUs now come with hardware RNG `rdrand`
- Intel "User Manual for the Rdrand Library": Use `rdrand` directly, don't rely on operating system
- Becker, Regazzoni, Paar, and Burleson in 2013: Describe almost undetectable hardware trojan that can be used to create a backdoor in `rdrand`

Wild speculation part II

Quantum Computers

- Can't quantum computers break all crypto?

Wild speculation part II

Quantum Computers

- Can't quantum computers break all crypto?
- No.

Wild speculation part II

Quantum Computers

- Can't quantum computers break all crypto?
- No.
- Currently used *asymmetric crypto* (RSA, ECC, DSA,. . .) will be broken
- Symmetric crypto (AES, Salsa20,. . .) needs to double key sizes

Wild speculation part II

Quantum Computers

- Can't quantum computers break all crypto?
- No.
- Currently used *asymmetric crypto* (RSA, ECC, DSA,...) will be broken
- Symmetric crypto (AES, Salsa20,...) needs to double key sizes
- There is no quantum computer, yet (also not for NSA)

Wild speculation part II

Quantum Computers

- Can't quantum computers break all crypto?
- No.
- Currently used *asymmetric crypto* (RSA, ECC, DSA,. . .) will be broken
- Symmetric crypto (AES, Salsa20,. . .) needs to double key sizes
- There is no quantum computer, yet (also not for NSA)
- Mark B. Ketchen (IBM) in 2012:

> *"Now I'm thinking like it's 15 [years] or a little more. It's within reach. It's within our lifetime. It's going to happen."*

Wild speculation part II

Quantum Computers

- Can't quantum computers break all crypto?
- No.
- Currently used *asymmetric crypto* (RSA, ECC, DSA,...) will be broken
- Symmetric crypto (AES, Salsa20,...) needs to double key sizes
- There is no quantum computer, yet (also not for NSA)
- Mark B. Ketchen (IBM) in 2012:

  > *"Now I'm thinking like it's 15 [years] or a little more. It's within reach. It's within our lifetime. It's going to happen."*

- What then? :

Wild speculation part II

Quantum Computers

- Can't quantum computers break all crypto?
- No.
- Currently used *asymmetric crypto* (RSA, ECC, DSA,...) will be broken
- Symmetric crypto (AES, Salsa20,...) needs to double key sizes
- There is no quantum computer, yet (also not for NSA)
- Mark B. Ketchen (IBM) in 2012:

  > *"Now I'm thinking like it's 15 [years] or a little more. It's within reach. It's within our lifetime. It's going to happen."*

- What then? *Post-Quantum Cryptography* to the rescue:
  - Asymmetric cryptography that survives quantum attacks
  - Ongoing research effort to make it practical

Summary

Biggest challenges (increasing hardness (?)):

Summary

Biggest challenges (increasing hardness (?)):

- Make post-quantum cryptography practical

Summary

Biggest challenges (increasing hardness (?)):

- Make post-quantum cryptography practical
- Eliminate side-channel leakages

Summary

Biggest challenges (increasing hardness (?)):

- Make post-quantum cryptography practical
- Eliminate side-channel leakages
- Better anonymity (at acceptable performance) than TOR

Summary

Biggest challenges (increasing hardness (?)):

- Make post-quantum cryptography practical
- Eliminate side-channel leakages
- Better anonymity (at acceptable performance) than TOR
- Obtain good randomness wherever and whenever it's needed

Summary

Biggest challenges (increasing hardness (?)):

- Make post-quantum cryptography practical
- Eliminate side-channel leakages
- Better anonymity (at acceptable performance) than TOR
- Obtain good randomness wherever and whenever it's needed
- Find a way to get rid of bad algorithms fast

Summary

Biggest challenges (increasing hardness (?)):

- Make post-quantum cryptography practical
- Eliminate side-channel leakages
- Better anonymity (at acceptable performance) than TOR
- Obtain good randomness wherever and whenever it's needed
- Find a way to get rid of bad algorithms fast
- Make high-security crypto easy to use for everybody