Public-Key Encryption, Key Exchange, Digital Signatures

CMSC 23200/33250, Autumn 2018, Lecture 7

David Cash

University of Chicago

Plan

1. Security of RSA
2. Key Exchange, Diffie-Hellman
3. Begin digital signatures

Assignment 1 is Due Wednesday

1. I will hold office hours Tomorrow (Tuesday), 2:30pm-4:30pm.

2. Thanks to everyone who reported server error bugs. I will respond to piazza posts this afternoon.

3. Please ping me on piazza if any more bugs comes up.

RSA "Trapdoor Function"

$PK = (N, e) \quad SK = (N, d)$ where $N = pq, \quad ed = 1 \bmod \phi(N)$

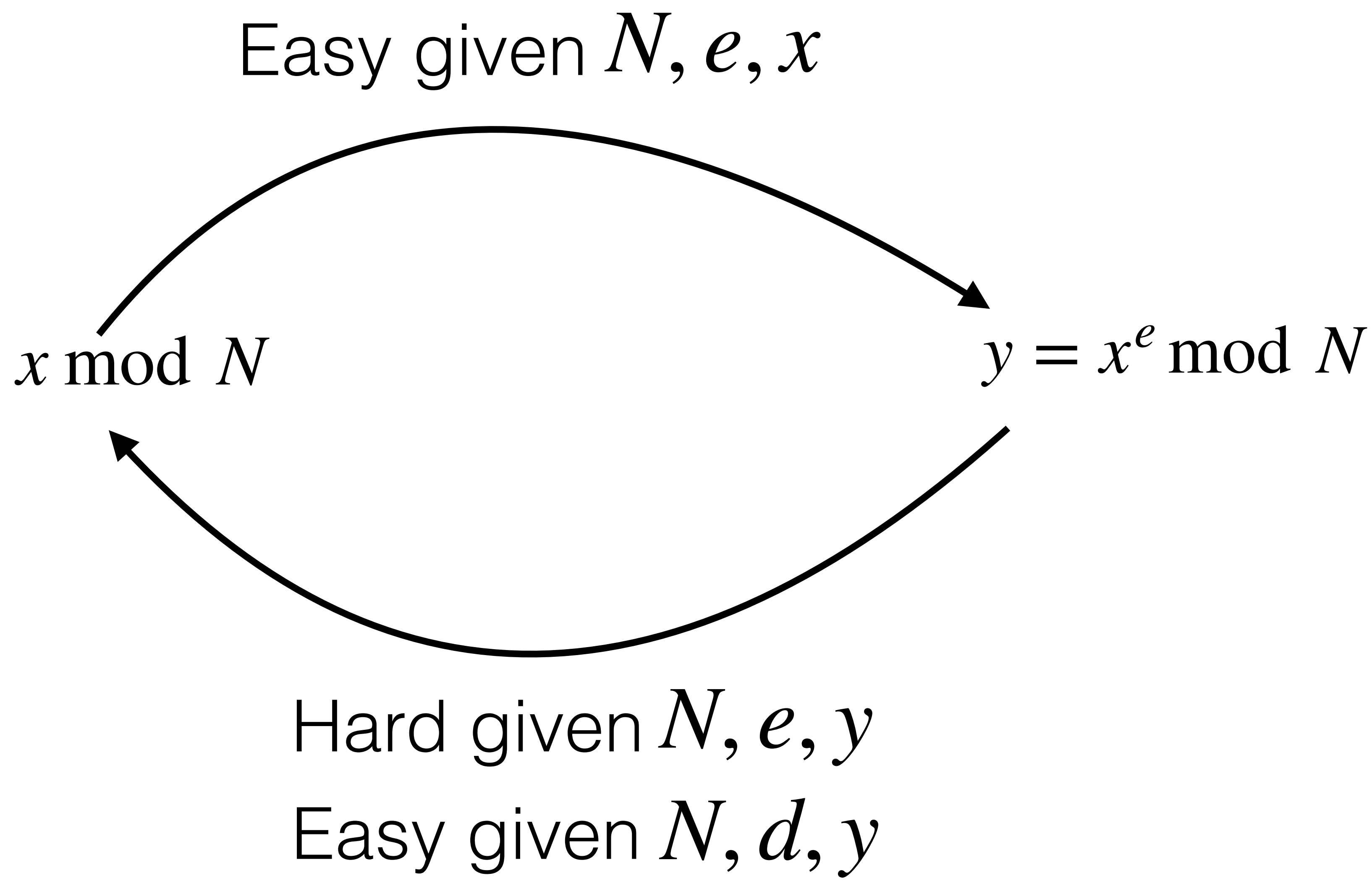
$$\mathrm{Enc}((N, e), M) = M^e \bmod N$$

$$\mathrm{Dec}((N, d), C) = C^d \bmod N$$

Messages and ciphertexts are in $\mathbb{Z}_N^*$

Setting up RSA:
- Pick two large random primes p,q
- Pick e and then find d using p and q
  - Usually e = 3 or e = 65537 = 0b10000000000000001

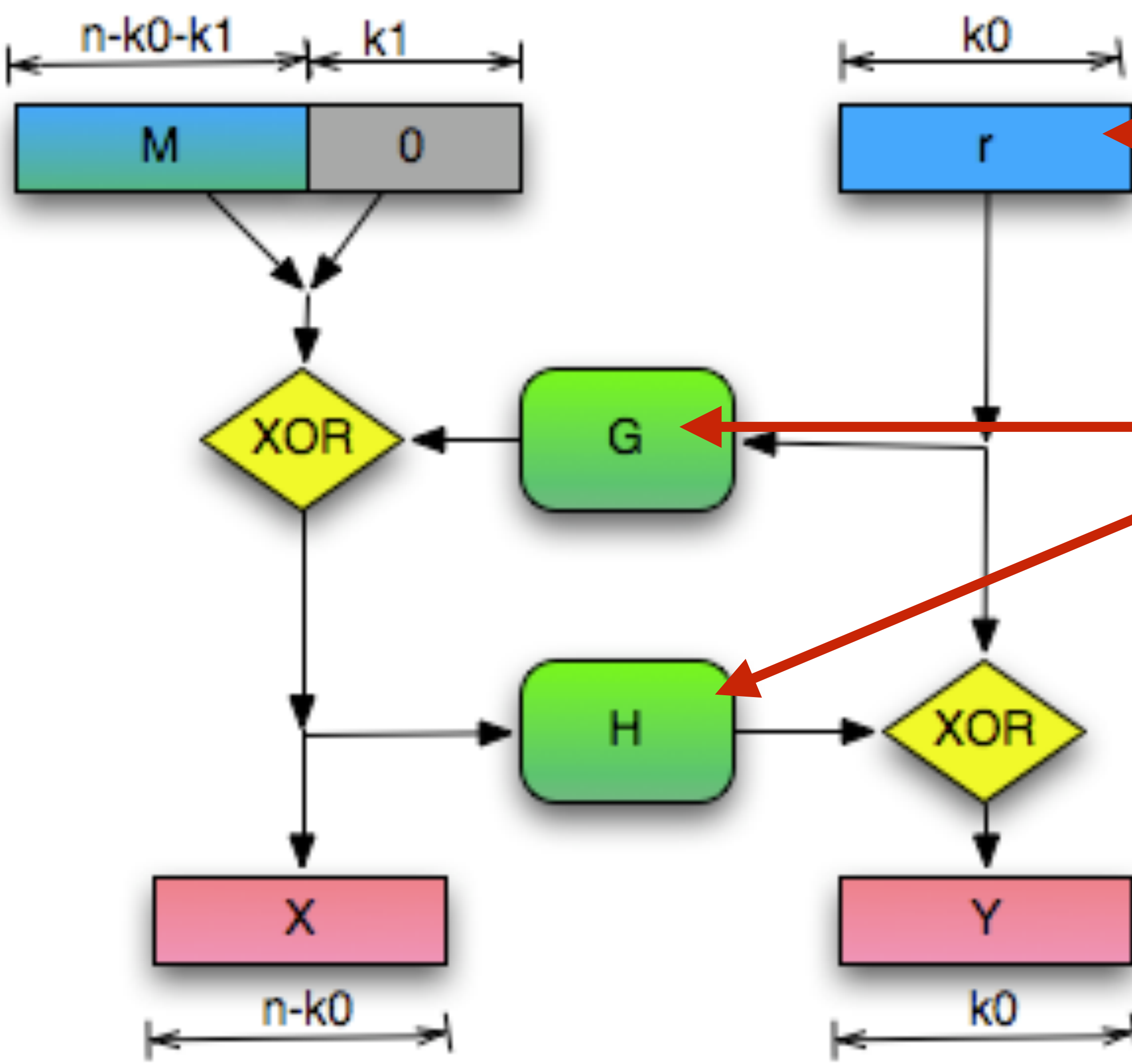
RSA "Trapdoor Function"



Finding "e-th roots modulo N" is hard.
Contrast is usual arithmetic, where finding roots is easy.

Better Padding: RSA-OAEP

RSA-OAEP [Bellare and Rogaway, ‘94] prevents padding-oracle attacks with better padding using a hash function.



random bytes

functions based on hash functions

Uses “Feistel Network” (!)

(Then apply RSA trapdoor function.)

Security of RSA Trapdoor Function Against Inversion

Inverting RSA Trapdoor Function — Given N, e, y find x such that $x^e = y \bmod N$

↑

If we know d… — Compute $x = y^d \bmod N$

↑

If we know $\varphi(N)$… — Compute $d = e^{-1} \bmod \varphi(N)$

↑

If we know p, q… — Compute $\varphi(N) = (p-1)(q-1)$

↑

But if we only know N… — Learning p and q from N is called the *factoring problem*.

- In principle one may invert RSA without factoring N, but it is the only approach known.

Naive Factoring Algorithm

- Given input `N=901`, what are `p`,`q`?

```
NaiveFactor(N):
  1. For i=2…sqrt(N):
          If i divides N:
            Output p=i, q=N/i
```

- Runtime is `sqrt(N)`≪`N`
- But `sqrt(N)` is still huge (e.g. $\mathtt{sqrt}(2^{2048})=2^{1024}$)

Factoring Algorithms

- If we can factor **`N`**, we can find **`d`** and break any version of RSA.

Algorithm	Time to Factor N
Naive: Try dividing by 1,2,3,…	$O(N^{.5}) = O(e^{.5\ln(N)})$
Quadratic Sieve	$O(e^c)$ $c = (\ln N)^{1/2}(\ln\ln N)^{1/2}$
Number Field Sieve	$O(e^c)$ $c = 1.9(\ln N)^{1/3}(\ln\ln N)^{2/3}$

- Total break requires $c = O(\ln\ln N)$

Factoring Records

- Challenges posted publicly by RSA Laboratories

Bit-length of N	Year
400	1993
478	1994
515	1999
768	2009

- Recommended bit-length today: 2048
- Note that fast algorithms force such a large key.
  - 512-bit N defeats naive factoring

Bad Randomness, Bad Primes, Bad Security

Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices

Nadia Heninger[†*] Zakir Durumeric[‡*] Eric Wustrow[‡] J. Alex Halderman[‡]

[†] *University of California, San Diego*
nadiah@cs.ucsd.edu

[‡] *The University of Michigan*
{zakir, ewust, jhalderm}@umich.edu

- Gathered moduli N from 10 million hosts (used in TLS and SSH)
- Factored ≈1% of all N… how?
- Many pairs of moduli shared **exactly one** prime factor
  - Find it fast using: $\gcd(N_1,N_2) = p$
  - … why?

- Bad randomness for entire execution is actually better
  - Can define **`q = H(p)`**

<u>KeyGen():</u>

1. Pick p — Might not be random at startup
2. Pick q — Slightly later, might be random
3. Pick e
4. Compute d
5. Output (N,e) and (N,d)

Public-Key Encryption in Practice: Hybrid Encryption

- RSA runs reasonably fast but is orders of magnitude slower than symmetric encryption with AES.
  - My laptop…
    - Can encrypt 800 MB per second using AES-CBC
    - Can only evaluate RSA 1000 times per second

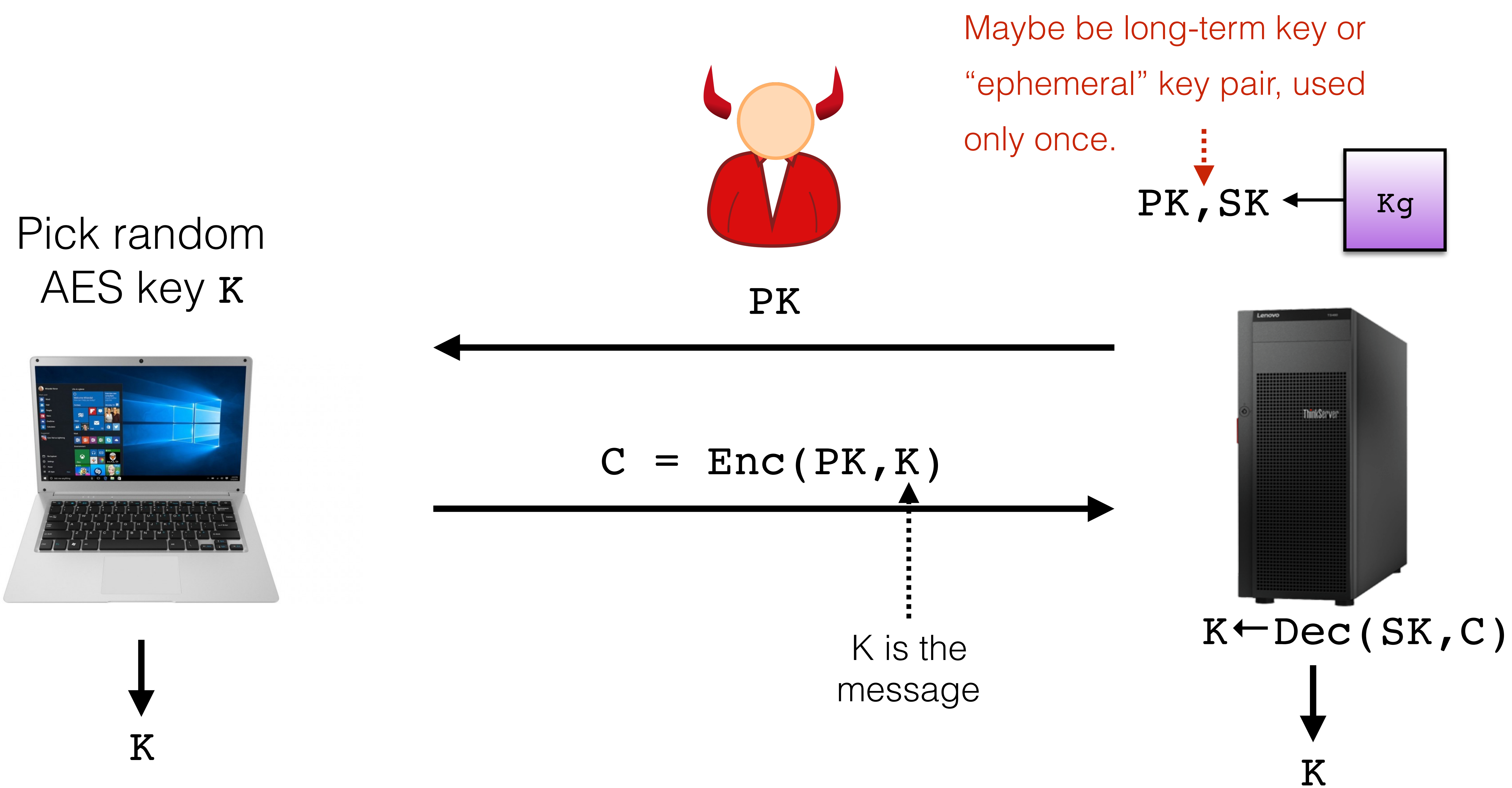
Solution: Use public-key encryption to send a 16-byte key K for AES. Then encrypt rest of traffic using authenticated encryption.

- Called “hybrid encryption”

Key Exchange and Hybrid Encryption

`(Kg, Enc, Dec)` is a public-key encryption scheme.

Goal: Establish secret key `K` to use with Authenticated Encryption.



Key Exchange and Hybrid Encryption

Key Exchange

AES-GCM(K,M_1)

AES-GCM(K,M_2)

AES-GCM(K,M_3)

...

- After up-front cost, bulk encryption is very cheap
- TLS/SSH Terminology:
  - "Handshake" = key exchange
  - "Record protocol" = symmetric encryption phase

An alternative approach to key exchange

- They modulus N for RSA is relatively large
  - Mostly important because it slows down encryption/decryption

- Now: A totally different, faster approach based on different math
  - Invented in 1970s, but new ideas have recently made it the standard choice
  - Strictly speaking, not public-key encryption, but can adapted into it if needed

The Setting: Discrete Logarithm Problem

Discrete Logarithm Problem:
Input: Prime $\mathtt{p}$, integers $\mathtt{g}$, $\mathtt{X}$.
Output: integer $\mathtt{r}$ such that $\mathtt{g}^{\mathtt{r}} = \mathtt{X} \bmod \mathtt{p}$.

- Different from factoring: Only one prime.
- Contrast with logarithms with real numbers, which are easy to compute. *Discrete* logarithms appear to be hard to compute
- Largest solved instances: 768-bit prime p (2016)

Diffie-Hellman Key Exchange

<u>Parameters: (fixed in standards, used by everyone)</u>:
Prime p (1024 bit usually)
Number $g \in \mathbb{Z}_p^*$ (usually 2)

$$(p, g)$$

Diffie-Hellman Key Exchange

<u>Parameters: (fixed in standards, used by everyone)</u>:

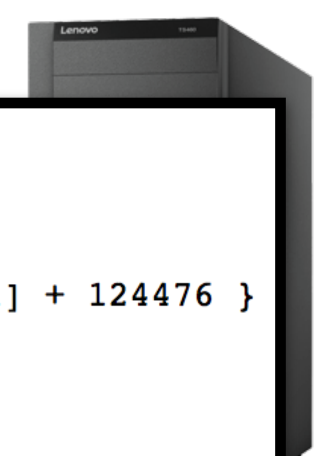
Prime p (1024 bit usually)

Number $g \in \mathbb{Z}_p^*$ (usually 2)

$$(p, g)$$

```
Network Working Group                                        M. Lepinski
Request for Comments: 5114                                       S. Kent
Category: Informational                                  BBN Technologies
                                                             January 2008


     Additional Diffie-Hellman Groups for Use with IETF Standards

Status of This Memo

   This memo provides information for
   not specify an Internet standard o
   memo is unlimited.

Abstract

   This document describes eight Diff
   in conjunction with IETF protocols
   communications.  The groups allow
   with a variety of security protoco
   (SSH), Transport Layer Security (T
   (IKE).
```

```
3.  2048-bit MODP Group

   This group is assigned id 14.

   This prime is: 2^2048 - 2^1984 - 1 + 2^64 * { [2^1918 pi] + 124476 }

   Its hexadecimal value is:

      FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1
      29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD
      EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245
      E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED
      EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE45B3D
      C2007CB8 A163BF05 98DA4836 1C55D39A 69163FA8 FD24CF5F
      83655D23 DCA3AD96 1C62F356 208552BB 9ED52907 7096966D
      670C354E 4ABC9804 F1746C08 CA18217C 32905E46 2E36CE3B
      E39E772C 180E8603 9B2783A2 EC07A28F B5C55DF0 6F4C52C9
      DE2BCBF6 95581718 3995497C EA956AE5 15D22618 98FA0510
      15728E5A 8AACAA68 FFFFFFFF FFFFFFFF

   The generator is: 2.
```



Diffie-Hellman Key Exchange

<u>Parameters: (fixed in standards, used by everyone)</u>:

Prime p (1024 bit usually)

Number $g \in \mathbb{Z}_p^*$ (usually 2)

$$(p, g)$$

Pick $r_A \in \{1,\ldots,p-1\}$

$X_A \leftarrow g^{r_A} \bmod p$

Pick $r_B \in \{1,\ldots,p-1\}$

$X_B \leftarrow g^{r_B} \bmod p$

X_A →

← X_B

$K \leftarrow X_B^{r_A} \bmod p$

$K \leftarrow X_A^{r_B} \bmod p$

Correctness: $X_B^{r_A} = (g^{r_B})^{r_A} = g^{r_A r_B} = (g^{r_A})^{r_B} = X_A^{r_B} \bmod p$

Security of Diffie-Hellman

$r_A \in \{1, \ldots, p-1\}$

$X_A \leftarrow g^{r_A} \bmod p$

Pick $r_B \in \{1, \ldots, p-1\}$

$X_B \leftarrow g^{r_B} \bmod p$

$\xrightarrow{X_A}$

$\xleftarrow{X_B}$

$K \leftarrow X_B^{r_A} \bmod p$

Knows: p, g, X_A, X_B

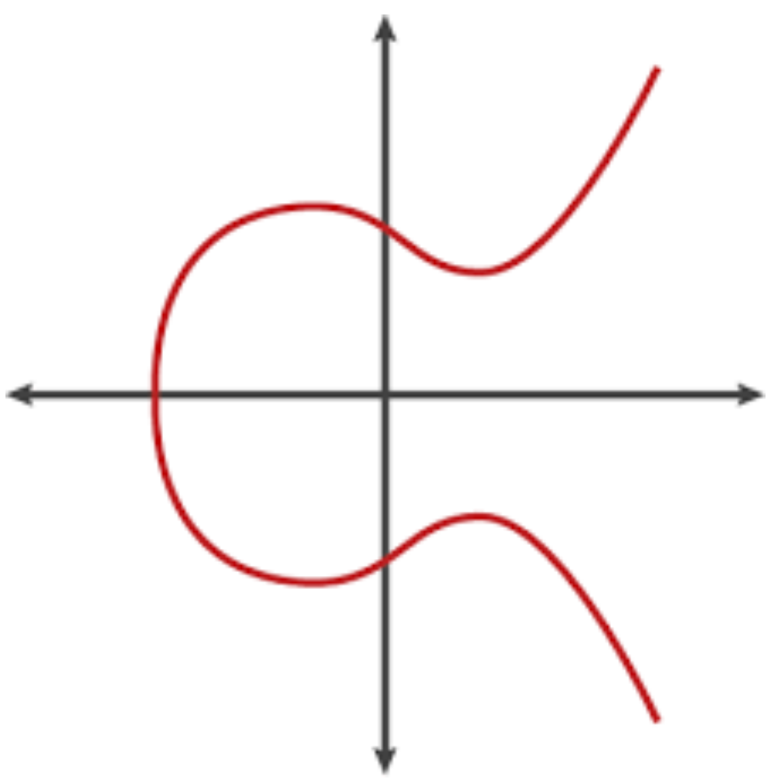
Compute K ?

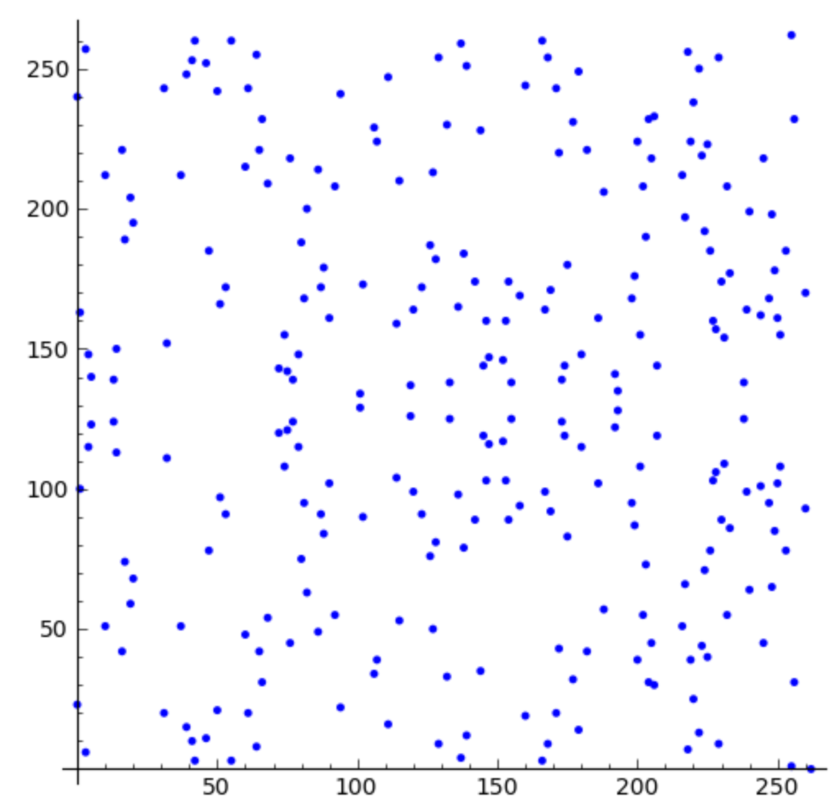
$K \leftarrow X_A^{r_B} \bmod p$

<u>Best attack known:</u> Compute discrete log of X_A, X_B

Key Exchange in the Future: Elliptic Curve Diffie-Hellman

- Diffie-Hellman works in any algebraic setting called a "finite cyclic group"
- Instead of multiplication modulo a prime, other settings have been suggested called "elliptic curve groups over finite fields"
- Advantage: Bandwidth and computation
  - 256 bit vs 2048-bit messages.





Public-Key Encryption/Key Exchange Wrap-Up

- RSA-OAEP and Diffie-Hellman (either mod a prime or in an elliptic curve) are unbroken and run fine in TLS/SSH/etc.
- Elliptic-Curve Diffie-Hellman is likely to be preferred choice going forward.

Shor's algorithm, 1994

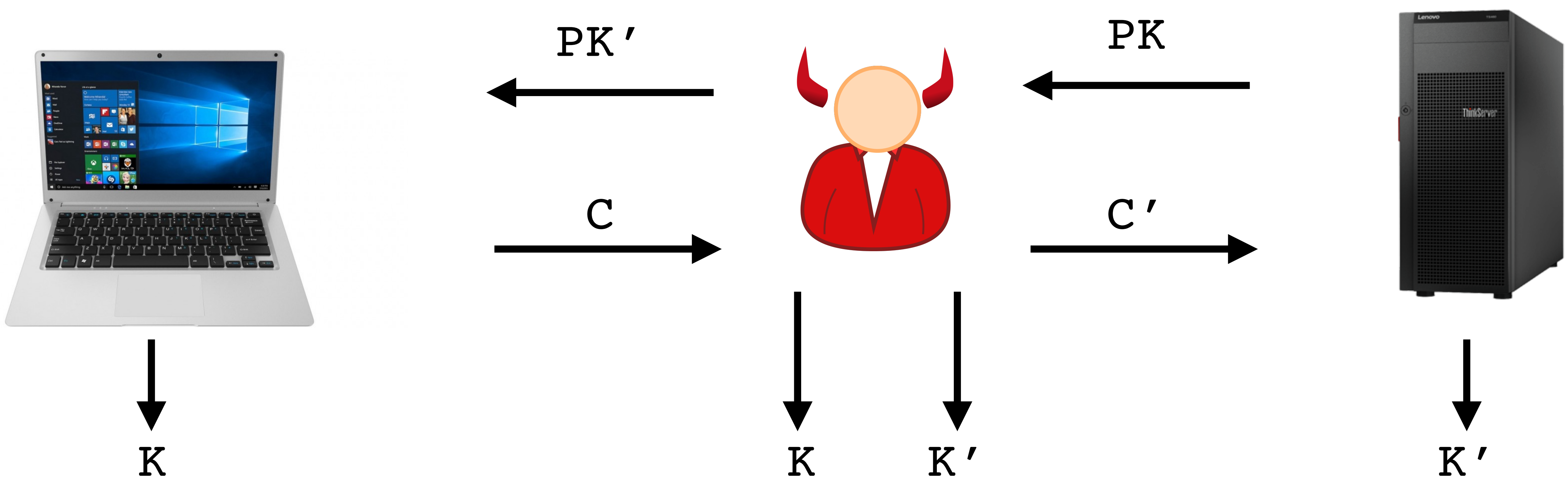
Peter Shor

Huge quantum computers will break:
- RSA (any padding)
- Diffie-Hellman (any finite cyclic group)

- First gen quantum computers will be far from this large
- "Post-quantum" crypto = crypto not known to be broken by quantum computers (i.e. not RSA or DH)
- On-going research on post-quantum cryptography from hard problems on lattices, with first beta deployments in recent years

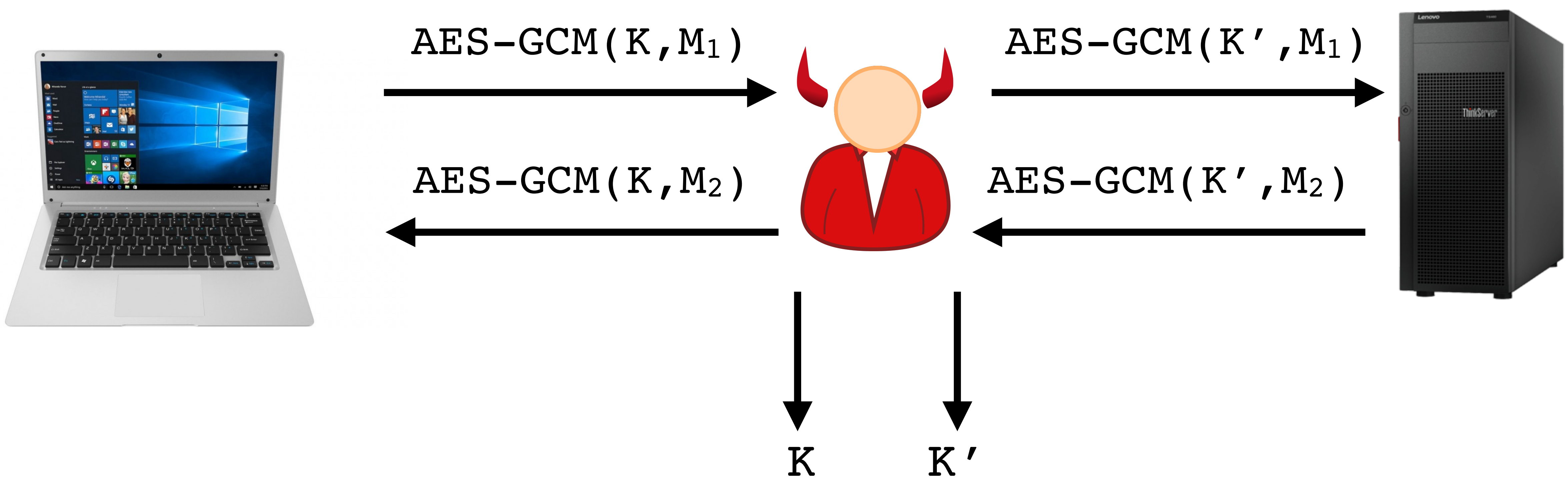
Key Exchange with a Person-in-the-Middle

Adversary may silently sit between parties and modify messages.



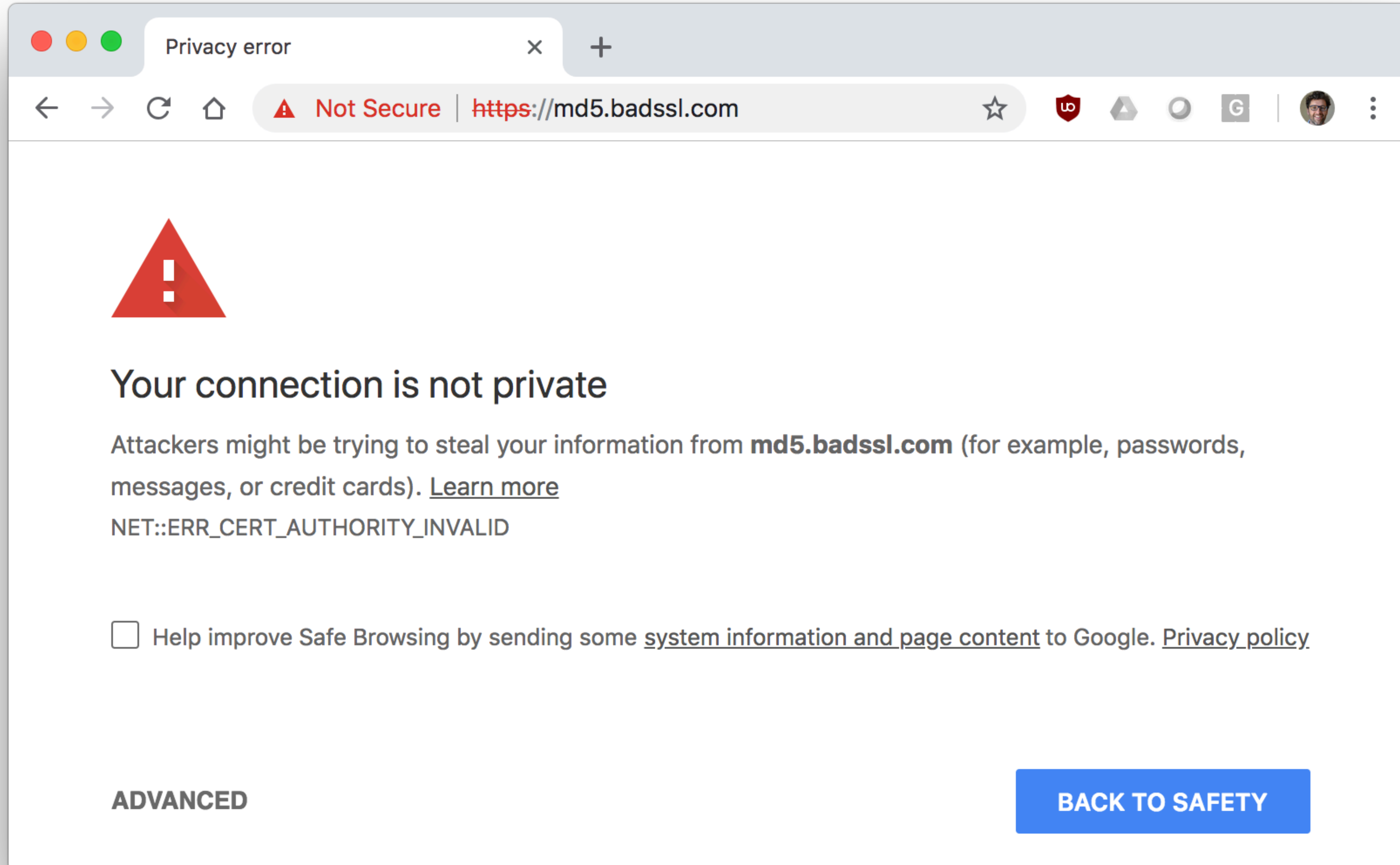
Parties agree on different keys, both known to adversary…

Key Exchange with a Person-in-the-Middle



Connection is totally transparent to adversary.

Translation is invisible to parties.


Privacy error
Not Secure
https://md5.badssl.com
Your connection is not private
Attackers might be trying to steal your information from md5.badssl.com (for example, passwords, messages, or credit cards). Learn more
NET::ERR_CERT_AUTHORITY_INVALID
Help improve Safe Browsing by sending some system information and page content to Google. Privacy policy
ADVANCED
BACK TO SAFETY

Next up: Stopping the Person-in-the-Middle

- Public-Key Infrastructure (PKI)
- Digital Signatures
- Certificates and chains of trust

Public Keys on the Internet

- Anyone can set up a server and generate their own keys.
- When you connect, how do you know you got the correct key?

(PK_1, SK_1)

google.com

(PK_2, SK_2)

amazon.com

(PK_3, SK_3)

facebook.com

(PK_4, SK_4)

twitter.com

Naive Solution

- Just distribute all the keys ahead of time, and store them locally!

```
keys.txt
google.com:PK1
amazon.com:PK2
facebook.com:PK3
twitter.com:PK4
...
```

Problems:

- List will be huge
- List will need to be updated often
- Who sends the list?
- Can adversaries tamper with list?

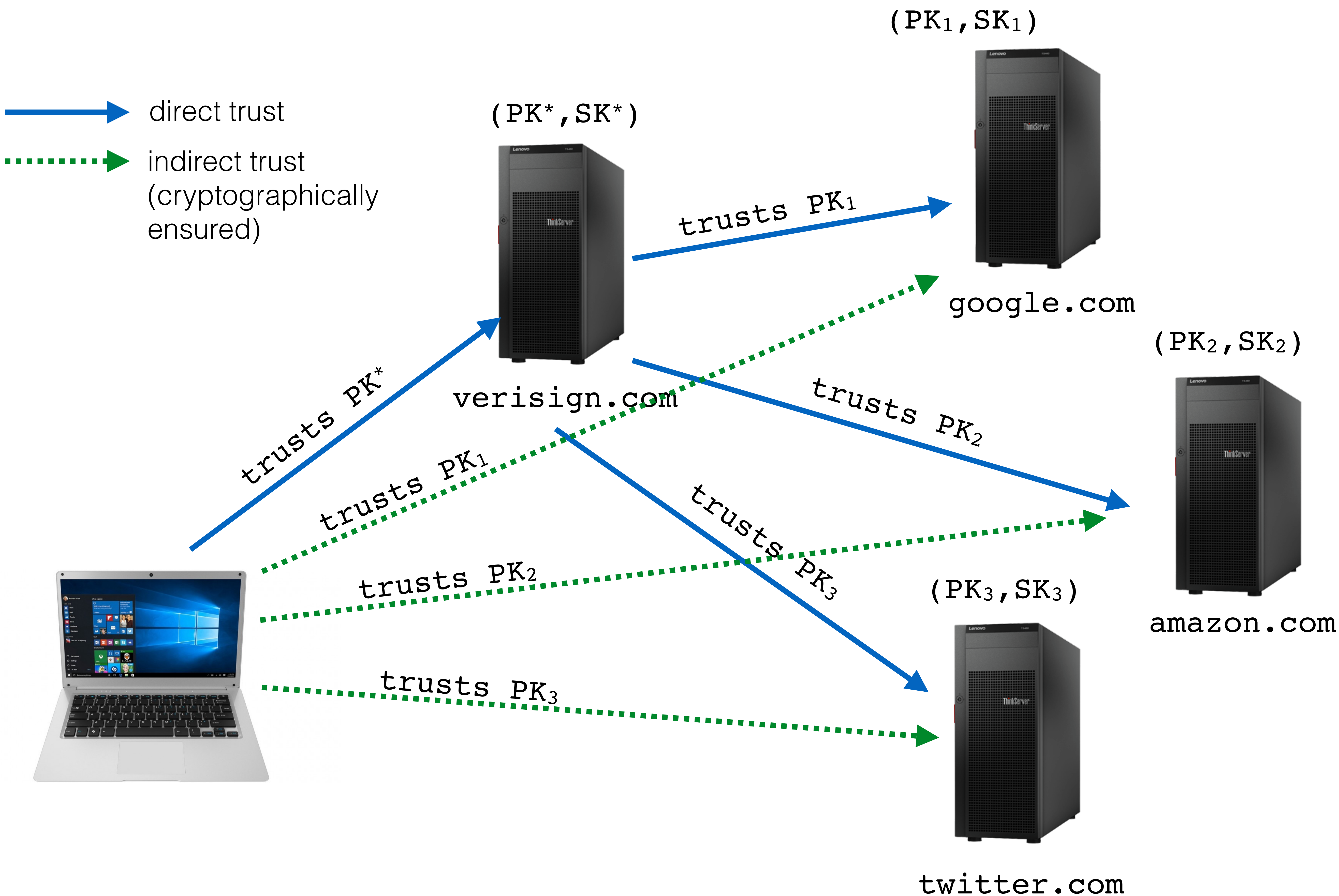
Distributing keys via “Transferring Trust”

- We will “transfer trust” from one key to another.

> If A knows that PK_B belongs to a trusted (in the eyes of A) entity B, and B knows that PK_C belongs to a trusted (in the eyes of B) entity C, then A should also trust C and PK_C.

- Initial “root” of trust established out-of-band via physical interaction.

Distributing keys via “Transferring Trust”



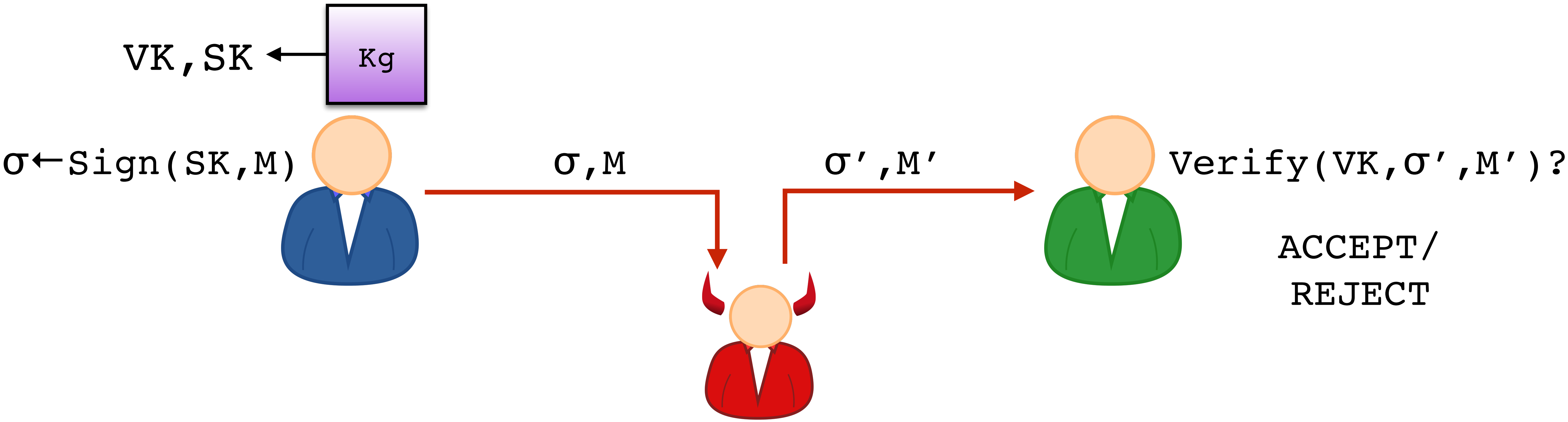
Crypto Tool: Digital Signatures

Definition. A <u>digital signature scheme</u> consists of three algorithms **`Kg`**, **`Sign`**, and **`Verify`**

- <u>Key generation algorithm **`Kg`**</u>, takes no input and outputs a (random) public-verification-key/secret-signing key pair **`(VK,SK)`**
- <u>Signing algorithm **`Sign`**</u>, takes input the secret key **`SK`** and a message **`M`**, outputs "signature" **σ←`Sign(SK,M)`**
- <u>Verification algorithm **`Verify`**</u>, takes input the public key V`K`, a message `M`, a signature **σ**, and outputs `ACCEPT/REJECT`

`Verify(VK,M,σ)=ACCEPT/REJECT`

Digital Signature Security Goal: Unforgeability



Scheme satisfies **unforgeability** if it is unfeasible for Adversary <u>(who knows `VK`)</u> to fool Bob into accepting **`M′`** not previously sent by Alice.

Industry Standard: RSA Signatures

$VK = (N, e)$ $SK = (N, d)$ where $N = pq,\ ed = 1 \bmod \phi(N)$

$$\text{Sign}((N, d), M) = H(M)^d \bmod N$$

$$\text{Verify}((N, e), M, \sigma) : \sigma^e = H(M) \bmod N?$$

Messages & sigs are in $\mathbb{Z}_N^*$

H is cryptographic hash function mapping strings to $\mathbb{Z}_N^*$

```
Totally broken if H is
not used, or weak H is used.
```

Broken

The End